Inleveropdracht 2 PI7

Onderzoeksrapport genetisch algoritme || AI Blok 1, Leerjaar 4

Justin Schots & Max Goldsmits
1812181 & 1836919
23-10-2023

Inhoud

Figurenlijst

1. Inleiding

Dit onderzoek heeft als doel om een genetisch algoritme te ontwikkelen en toepassen voor het verbeteren van de instellingen van een neuraal netwerk. Neurale netwerken zijn heel flexibel en je kunt dingen zoals het aantal lagen, het aantal neuronen in elke laag, de activatiefunctie en de leersnelheid aanpassen. We willen de beste combinatie van deze instellingen vinden, afhankelijk van de specifieke gegevens die we gebruiken. Een handige methode hiervoor is een genetisch algoritme, dat geïnspireerd is door de manier waarop natuurlijke selectie werkt. Het genetisch algoritme zal verschillende instellingen uitproberen, beoordelen en aanpassen met behulp van kruising en mutatie om de best mogelijke instellingen te vinden.

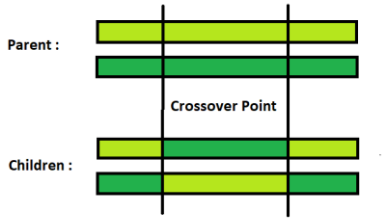
Dit onderzoeksrapport richt zich op het begrijpen waarom een genetisch algoritme een positieve invloed heeft op de prestaties van een neuraal netwerk. Tegen het einde van dit document beoogt de lezer een grondige kennis te hebben over hoe dit werkt en waarom het belangrijk is.

2. Beschrijving van het te gebruiken algoritme:

Het genetisch algoritme is een evolutionaire optimalisatiemethode waarbij een populatie van individuen wordt gecreëerd, beoordeeld en aangepast om de optimale hyperparameters voor een neuraal netwerk te ontdekken. Dit omvat diverse aspecten, waaronder selectie, cross-over en mutatie.

2.1 Two-Point Cross-over

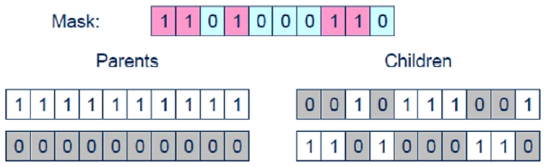
Cross-over, ook wel recombination genoemd, is een van de essentiële operaties binnen het genetische algoritme. Het betreft het proces waarbij paren van individuen (ouders) worden samengevoegd om nieuwe individuen (kinderen) te genereren. Tijdens dit proces wordt genetische informatie gedeeld en geïntegreerd. Cross-over draagt bij aan de genetische variabiliteit en bevordert de exploratie van nieuwe hyperparametercombinaties. (Dutta, 2023)



Figuur 1: Visueel voorbeeld cross-over

2.1.1 Uniforme Cross-over

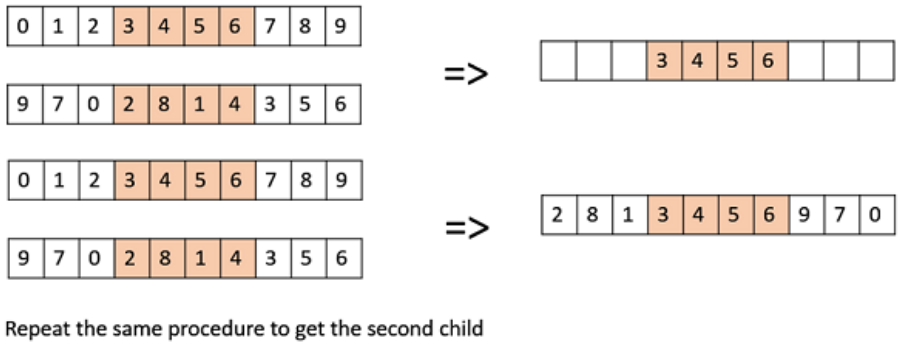
Uniforme cross-over is een van de fundamentele technieken om genetische informatie van ouders te combineren. Bij deze methode worden hyperparameters willekeurig gekozen uit de genetische code van de ouders. Hierdoor hebben de resulterende kinderen een willekeurige combinatie van hyperparameters van beide ouders. Dit brengt een zekere mate van toeval en variabiliteit in de populatie. (Dutta, 2023)



Figuur 2: Visueel voorbeeld Uniforme Cross-over

2.1.2 Order Cross-over

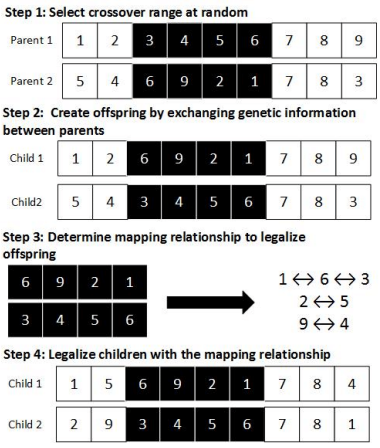
Order cross-over is een variant van cross-over die oorspronkelijk is ontwikkeld voor permutatieproblemen, waarbij de volgorde van elementen van belang is. Bij hyperparameteroptimalisatie houdt order cross-over rekening met de volgorde van hyperparameters binnen de genetische code. Deze methode is relevant wanneer de volgorde van hyperparameters een rol speelt. (Tutorialspoint, sd)



Figuur 3: Visueel voorbeeld Order Cross-over

2.1.3 Partially-Mapped Cross-over

Partially-Mapped cross-over (PMX) is een meer geavanceerde cross-overmethode die oorspronkelijk is ontwikkeld voor permutatieproblemen, maar kan worden aangepast voor hyperparametercombinaties. Het zorgt ervoor dat er geen duplicaten ontstaan in de resulterende individuen. PMX is geschikt wanneer unieke hyperparametercombinaties vereist zijn. (Desjardins, Falcon, Abielmona, & Petriu, 2017)

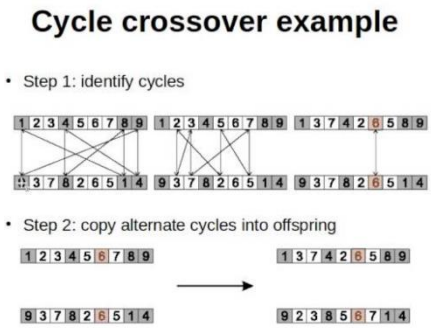


Figuur 4: Visueel voorbeeld Partially-Mapped Cross-over

2.1.4 Cycle Cross-over

Cycle cross-over is een methode waarbij cycli worden gevormd in de genetische code van de ouders. Deze cycli worden vervolgens gebruikt om de genetische informatie uit te wisselen tussen de ouders, waardoor kinderen ontstaan met complexe combinaties van hyperparameters.

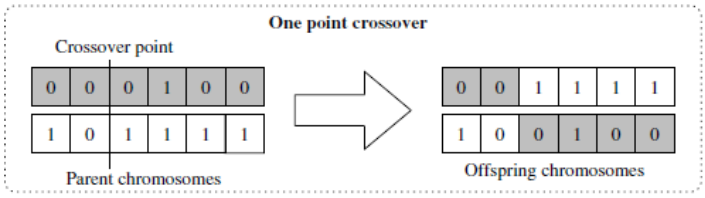
De keuze van de cross-overmethode is van groot belang, omdat deze bepaalt hoe genetische informatie wordt gecombineerd en welke soorten nieuwe hyperparametercombinaties mogelijk zijn. Het selecteren van de juiste methode is afhankelijk van de aard van het probleem en de gewenste genetische variabiliteit. (Sadawi, 2015)



Figuur 5: Visueel voorbeeld Cycle Cross-over

2.1.5 Single-Point Cross-over

Bij de single-point cross-over-methode wordt een willekeurig punt in de genetische code van de parents geselecteerd. Op dit punt worden de hyperparameters uitgewisseld tussen de twee ouders, waardoor kinderen ontstaan met genetische informatie van beide ouders. Single-point cross-over is een veelgebruikte en eenvoudige methode. (Dutta, 2023)



Figuur 6: Visueel voorbeeld Single-Point Cross-over

3 Fitness

In dit hoofdstuk behandelen we de evaluatie van de fitness van individuen in het genetische algoritme voor hyperparameteroptimalisatie van neurale netwerken. De fitnessfunctie is cruciaal omdat deze de prestaties van individuen meet en bepaalt welke individuen zich voortplanten.

3.1 Accuracy

Accuracy is een eenvoudige classificatiemetrica die aangeeft hoeveel van de voorspellingen correct zijn ten opzichte van het totale aantal voorspellingen. Het wordt vaak uitgedrukt als een percentage en meet de totale correctheid van het model.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Figuur 7: Formule voor Accuracy

3.2 Precision

Precision is een classificatiemetrica die de nauwkeurigheid van positieve voorspellingen meet. Het berekent het aantal ware positieve voorspellingen gedeeld door het totale aantal positieve voorspellingen. Precision geeft aan hoe goed het model is in het identificeren van echte positieve gevallen.

$$Precision = \frac{TP}{TP + FP}$$

Figuur 8: Formule voor Precision

3.3 Recall

Recall is een classificatiemetrica die de gevoeligheid voor ware positieven meet. Het berekent het aantal ware positieve voorspellingen gedeeld door het totale aantal ware positieve gevallen. Recall geeft aan hoe goed het model in staat is om alle relevante positieve gevallen te identificeren.

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)}$$

Figuur 9: Formule voor Recall

3.4 F1-score

De F1-score is een evaluatiemetrica die vaak wordt gebruikt bij classificatieproblemen. Het is de harmonische gemiddelde van precision en recall. Precision meet de nauwkeurigheid van positieve voorspellingen, terwijl recall de gevoeligheid voor ware positieven meet. De F1-score geeft een evenwichtige maat voor de prestaties van het classificatiemodel.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \textbf{F1 Score} = \frac{2}{\left(\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}\right)}$$

$$\text{Recall} = \frac{TP}{TP + FN} \qquad \textbf{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figuur 10:Formules voor het berekenen van de F1-Score

3.5 Absolute Fitness

De absolute fitness van een individu in het genetische algoritme wordt bepaald door de waarde van de gekozen fitnessfunctie. Deze waarde geeft de mate van prestatie aan op basis van de specifieke metriek die wordt gebruikt.

3.6 Gestandaardiseerde Fitness

De gestandaardiseerde fitness is een genormaliseerde versie van de absolute fitness, waarbij individuele fitnessscores worden omgezet naar een schaal tussen 0 en 1. Dit zorgt voor consistentie bij het vergelijken van individuen met verschillende fitnessmetingen.

3.7 Gerangschikte Fitness

De gerangschikte fitness houdt rekening met de positie van een individu binnen de populatie op basis van zijn fitnessscore. Individuen met hogere fitnessscores krijgen doorgaans een hogere rangorde, wat van invloed kan zijn op hun selectiekansen in het genetische algoritme.

Het kiezen van de juiste fitnessfunctie is essentieel voor het succes van het genetische algoritme, omdat het bepaalt welke individuen zich voortplanten en welke eigenschappen worden doorgegeven aan de volgende generatie. De keuze van de fitnessfunctie moet overeenkomen met het doel van het neurale netwerk, of het nu gaat om regressie of classificatie, en de gewenste prestaties op de gegeven dataset.

3.8 Mean Squared Error (MSE)

MSE (Mean Squared Error) is een veelgebruikte regressiemetrica die de gemiddelde kwadratische fout tussen de voorspellingen van het neurale netwerk en de werkelijke waarden meet. Het kwadrateert het verschil tussen de voorspelling en de werkelijke waarde voor elk gegevenspunt, berekent de gemiddelde waarde van deze kwadraten en levert een positieve waarde op. Een lagere MSE geeft aan dat de voorspellingen dichter bij de werkelijke waarden liggen. (Akshita, 2020)

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

Figuur 11: Formule voor MSE

3.9 Root Mean Squared Error (RMSE)

RMSE (Root Mean Squared Error) is de vierkantswortel van de MSE. Het geeft een maat voor de gemiddelde absolute fout tussen voorspellingen en werkelijke waarden. Het is positief en schaalt met de eenheid van de uitvoerwaarden. Net als bij MSE geldt dat een lagere RMSE wijst op betere voorspellingen. (Akshita, 2020)

3.10 Mean Absolute Error (MAE)

MAE (Mean Absolute Error) is een regressiemetrica die de gemiddelde absolute fout tussen voorspellingen en werkelijke waarden meet. Het berekent het gemiddelde van de absolute waarden van de fouten voor elk gegevenspunt. MAE is robuust tegen outliers en geeft de gemiddelde grootte van de fouten aan. (Akshita, 2020)

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Figuur 12: Formule voor RMSE

3.11 R-squared (R^2)

R^2 (R-squared) is een maatstaf voor de variantie die wordt verklaard door het model. Het meet hoe goed het model past bij de gegevens door de verhouding tussen de verklaarde variantie en de totale variantie te berekenen. R^2 varieert tussen 0 en 1, waarbij hogere waarden aangeven dat het model een groter deel van de variantie verklaart. (Akshita, 2020)

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2}$$

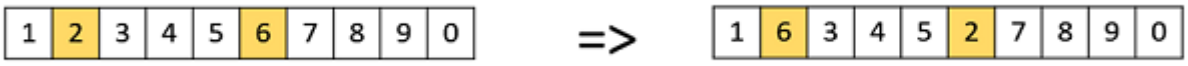
Figuur 13: Formule voor R-squared

4 Mutatie

In dit hoofdstuk bespreken we het aspect van mutatie in het genetische algoritme voor hyperparameteroptimalisatie van neurale netwerken. Mutatie is een cruciaal mechanisme om genetische diversiteit te behouden en nieuwe eigenschappen te introduceren in de populatie.

4.1 Swap Mutatie

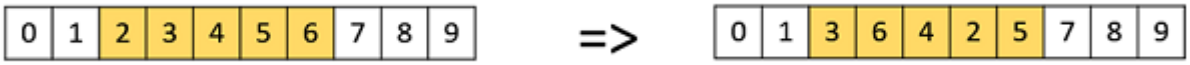
Swap mutatie is een mutatiemethode waarbij twee willekeurige hyperparameters binnen de genetische code van een individu van plaats wisselen. Deze mutatie introduceert variabiliteit door de waarden van hyperparameters te verwisselen. Het is een eenvoudige vorm van mutatie en kan helpen bij het verkennen van verschillende combinaties van hyperparameters. (Tutorialspoint, sd)



Figuur 14: een voorbeeld van een swap mutatie

4.2 Scramble-mutatie

Scramble-mutatie is een mutatiemethode waarbij de volgorde van hyperparameters binnen een willekeurig geselecteerd bereik wordt veranderd. Hierdoor ontstaan willekeurige wijzigingen in de volgorde van hyperparameters, wat kan leiden tot nieuwe configuraties van het neurale netwerk. Scramble-mutatie is nuttig om lokale optimalisatie te voorkomen en de diversiteit te vergroten. (Tutorialspoint, sd)



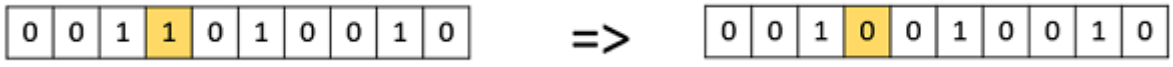
Figuur 15: voorbeeld van een Scramble-mutatie

4.3 Grensmutatie

Grensmutatie is een mutatiemethode waarbij hyperparameters worden aangepast binnen een bepaalde grenswaarde. Bijvoorbeeld, de waarde van het aantal neuronen in een laag kan worden aangepast door een kleine waarde toe te voegen of af te trekken, maar binnen de vooraf gedefinieerde grenzen. Dit helpt bij het finetunen van de hyperparameters zonder abrupte veranderingen. (Itano, Angelo de Abreu de Sousa, & Del-Moral-Hernandez, 2018)

4.4 Bit-Flip Mutatie

Bit-flip mutatie wordt vaak gebruikt bij binaire genetische codes, waarin elke hyperparameter wordt gerepresenteerd als een reeks bits. Tijdens bit-flip mutatie wordt een willekeurige bit omgedraaid, wat resulteert in een verandering in de waarde van de hyperparameter. Deze mutatiemethode is relevant wanneer hyperparameters als binaire vlaggen worden gerepresenteerd. (Tutorialspoint, sd)



Figuur 16: Voorbeeld van een bit-flip mutatie

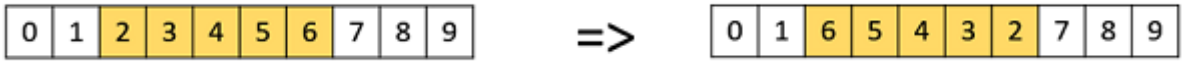
4.5 Gaussiaanse Mutatie

Gaussiaanse mutatie is een mutatiemethode waarbij een kleine willekeurige waarde wordt toegevoegd aan hyperparameters volgens een normale verdeling. Hierdoor ontstaan kleine willekeurige wijzigingen in de hyperparameters, wat helpt bij het verkennen van de zoekruimte en het finetunen van de configuratie. (Xing, Zhang, Zou, & Lin, 2021)

4.6 Inversiemutatie

Inversiemutatie is een mutatiemethode waarbij de volgorde van een reeks hyperparameters wordt omgekeerd. Deze mutatie kan van toepassing zijn wanneer de volgorde van hyperparameters relevant is, zoals bij permutatieproblemen.

De keuze van de mutatiemethode hangt af van het specifieke probleem en de gewenste mate van genetische diversiteit. Een geschikte mutatiestrategie draagt bij aan het verkennen van verschillende hyperparameterconfiguraties en het vermijden van vroegtijdige convergentie naar suboptimale oplossingen. (Tutorialspoint, sd)



Figuur 17: Voorbeeld van een Inversiemutatie

5 Hyperparameters

In dit hoofdstuk behandelen we de verschillende hyperparameters die moeten worden geconfigureerd om het genetisch algoritme effectief te laten werken. Deze hyperparameters bepalen de instellingen en het gedrag van het algoritme.

5.1 Populatiegrootte

De populatiegrootte verwijst naar het aantal individuen in elke generatie van de populatie. Het bepalen van de juiste populatiegrootte is cruciaal voor het succes van het genetische algoritme. Een te kleine populatie kan leiden tot convergentie naar suboptimale oplossingen, terwijl een te grote populatie de rekentijd en geheugenvereisten kan verhogen. De optimale populatiegrootte kan variëren afhankelijk van de complexiteit van het probleem en de rekenbronnen die beschikbaar zijn. (Russel & Norvig, 2021)

5.2 Mutatieratio

De mutatieratio bepaalt de kans dat een individu wordt onderworpen aan mutatie in elke generatie. Mutatie is essentieel om genetische diversiteit te behouden en nieuwe eigenschappen te introduceren in de populatie. Een te lage mutatieratio kan leiden tot vroegtijdige convergentie, terwijl een te hoge mutatieratio de stabiliteit van de populatie kan verstoren. De optimale mutatieratio is afhankelijk van de aard van het probleem en de selectiemethode. (Russel & Norvig, 2021)

5.3 Cross-over ratio

De cross-over ratio bepaalt de kans dat twee individuen willekeurig zullen kruisen en nieuwe individuen zullen produceren in elke generatie. Cross-over is verantwoordelijk voor het combineren van genetisch materiaal van ouders en het creëren van nieuwe individuen. Een te lage cross-over ratio kan leiden tot een gebrek aan genetische diversiteit, terwijl een te hoge cross-over ratio de stabiliteit van de populatie kan verminderen. De optimale cross-over ratio is afhankelijk van het specifieke probleem en de gebruikte cross-overmethode. (Russel & Norvig, 2021)

5.4 Generatieaantal

Het generatieaantal bepaalt hoeveel generaties het genetische algoritme zal doorlopen voordat het eindigt. Een te klein aantal generaties kan leiden tot onvolledige zoekruimteverkenning, terwijl een te groot aantal generaties de rekentijd kan verlengen zonder aanzienlijke verbeteringen in de resultaten. Het kiezen van het juiste generatieaantal hangt af van de complexiteit van het probleem en de gewenste zoeknauwkeurigheid. (Russel & Norvig, 2021)

5.5 Selectiemethode

Naast de bovengenoemde hyperparameters is de keuze van de selectiemethode een essentieel aspect van het genetische algoritme. De selectiemethode bepaalt hoe individuen worden gekozen om zich voort te planten op basis van hun fitnessscores. Populaire selectiemethoden zijn roulette wheel selectie, rangselectie, toernooiselectie en elitisme. Elke methode heeft zijn eigen invloed op de convergentie en diversiteit van de populatie.

De configuratie van deze hyperparameters vereist zorgvuldige afstemming en experimentatie om een evenwicht te vinden tussen exploratie en exploitatie, en om een oplossing van hoge kwaliteit te vinden binnen redelijke rekentijd. (Russel & Norvig, 2021)

6 Formele Beschrijving van het Probleem

In dit hoofdstuk geven we een formele beschrijving van het probleem, zoals volgens de normen van het boek "Artificial Intelligence: A Modern Approach" van Peter Norvig en Stuart J. Russell.

6.1 Probleembeschrijving

Het probleem dat wordt aangepakt, is het vinden van de optimale hyperparameters voor een neuraal netwerk om de prestaties op een specifieke dataset te maximaliseren. Dit omvat het bepalen van hyperparameters zoals de activation-function, het aantal neuronen per laag en de solver (optimizer) die wordt gebruikt voor het trainen van het neurale netwerk.

6.2 Toestand Representatie

Elke individuele oplossing (genetisch individu) wordt gerepresenteerd als een reeks hyperparameters die de configuratie van het neurale netwerk bepalen. De toestandsruimte bestaat uit alle mogelijke combinaties van hyperparameters die het neurale netwerk kunnen definiëren.

6.3 Genetische Code Individu

De genetische code van een individu omvat de volgende hyperparameters:

Het type activation-function, dat kan worden gekozen uit de set ['logistic', 'tanh', 'relu', 'identity'].

Het aantal neuronen in een enkele verborgen laag, dat kan variëren uit de set [10, 20, 30, 40].

Het type solver, dat kan worden gekozen uit de set ['lbfgs', 'sgd', 'adam'].

Elk element in de genetische code vertegenwoordigt een variabele voor hyperparameters, en het doel van het genetische algoritme is om de combinatie van deze hyperparametervariabelen te vinden die de beste prestaties levert (hoogste nauwkeurigheid) voor het MLP-model op de gegeven dataset. Het genetische algoritme doorloopt een iteratief proces waarin een populatie van deze genetische codes evolueert om de hyperparameters te optimaliseren.

6.4 Evaluatie van de Fitness van een Individu (Fitness Functie)

Voor het evalueren van de fitness zijn er bepaalde bijpassende metrieken gebruikt die eerder genoemd zijn in dit onderzoek document. Hieronder staat een tabel van de uitkomsten van de regressie en classificatie neurale netwerken die zijn verbeterd met het GA (genetisch algoritme)

6.5 Selectie

De selectie van individuen die zich mogen voortplanten, wordt uitgevoerd op basis van hun behaalde fitnessscores. Individuen met hogere fitnessscores hebben een grotere kans om te worden geselecteerd voor reproductie. Dit proces bootst het principe van "survival of the fittest" na, waarbij de best presterende individuen een grotere kans hebben om hun genetische informatie door te geven aan de volgende generatie.
Dit voorbeeld valt terug te zien in het stukje code hier onder:

```
        # Setting the new population as the current population
        population = new_population

    # Returning the best individual
    best_individual = max(population, key=lambda ind: evaluate_individual(ind[0], ind[1], ind[2]))
    return best_individual
```

✓ 0.0s

6.6 Cross-over

Cross-over is het proces waarbij paren van individuen willekeurig worden geselecteerd en gecombineerd om nieuwe individuen te creëren. Tijdens dit proces wordt de genetische code gecombineerd, wat resulteert in nieuwe individuen met gemengde hyperparameters.
De Cross-over vindt hier plaats in de code:

```
# The genetic algorithm
for generation in range(generations):
    scores = [evaluate_individual(activation_function, hidden_layer_size, solver) for activation_function, hidden_layer_size, solver in population]
    selected_indices = np.argsort(scores)[::-1][:int(population_size * 0.2)]
    selected_population = [population[i] for i in selected_indices]
    new_population = []

    # Creating the new population with random crossover and mutation
    for _ in range(population_size):
        index1, index2 = np.random.choice(len(selected_population), size=2, replace=False)
        parent1 = selected_population[index1]
        parent2 = selected_population[index2]

        if np.random.rand() < 0.5:
            child = (parent1[0], parent2[1], parent2[2])
        else:
            child = (parent2[0], parent1[1], parent1[2])

        new_population.append(child)
```

6.7 Mutatie: Op welke manier worden mutaties gemaakt in de genetische code?

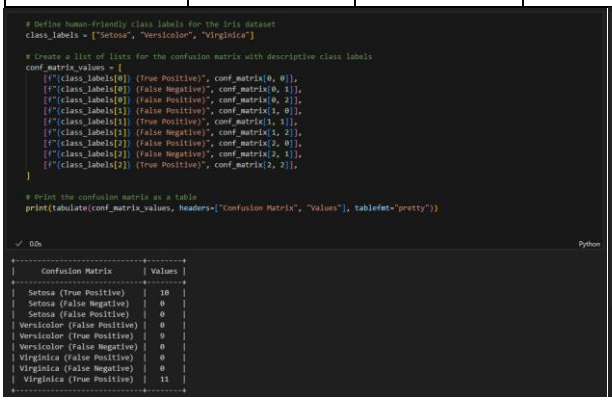
Mutatie is het proces waarbij willekeurige veranderingen worden aangebracht in de genetische code van individuen om genetische diversiteit te introduceren. Mutatie kan verschillende vormen aannemen, waaronder swap mutatie, scramble-mutatie, grensmutatie, bit-flip mutatie, gaussiaanse mutatie en inversiemutatie. De keuze van het mutatietype en de kans op mutatie beïnvloeden de mate van verandering in de genetische code van individuen.

Deze formele beschrijving van het probleem geeft een duidelijk overzicht van de belangrijkste elementen van het genetische algoritme voor hyperparameteroptimalisatie van neurale netwerken. Het is gebaseerd op de principes van evolutionaire algoritmen en biedt een gestructureerd kader voor het begrijpen en oplossen van het probleem.

7. Evaluatie van het Algoritme

In dit hoofdstuk zullen we de prestaties van het genetische algoritme voor hyperparameteroptimalisatie onderzoeken en beoordelen hoe dit algoritme de prestaties van het neurale netwerk heeft beïnvloed. We nemen aan dat het genetische algoritme daadwerkelijk positieve resultaten heeft opgeleverd voor het neurale netwerk, zoals aangegeven in de code-uitvoering.

Model	Accuracy	Precision	Recall	F1-Score	TP	FP	FN	FP
Classificatie	1.0	1.0	1.0	1.0	30	0	0	0

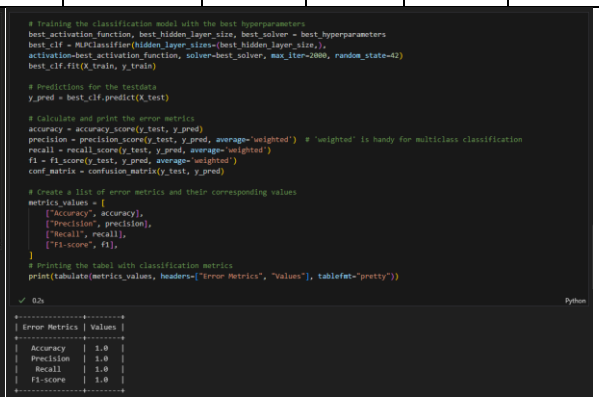


```
# Define human-friendly class labels for the Iris dataset
class_labels = ["Setosa", "Versicolor", "Virginica"]

# Create a list of lists for the confusion matrix with descriptive class labels
conf_matrix_values = [
    [f"{class_labels[0]} (True Positive)", conf_matrix[0, 0]],
    [f"{class_labels[0]} (False Negative)", conf_matrix[0, 1]],
    [f"{class_labels[0]} (False Positive)", conf_matrix[0, 2]],
    [f"{class_labels[1]} (False Positive)", conf_matrix[1, 0]],
    [f"{class_labels[1]} (True Positive)", conf_matrix[1, 1]],
    [f"{class_labels[1]} (False Negative)", conf_matrix[1, 2]],
    [f"{class_labels[2]} (False Positive)", conf_matrix[2, 0]],
    [f"{class_labels[2]} (False Negative)", conf_matrix[2, 1]],
    [f"{class_labels[2]} (True Positive)", conf_matrix[2, 2]],
]

# Print the confusion matrix as a table
print(tabulate(conf_matrix_values, headers=["Confusion Matrix", "Values"], tablefmt="pretty"))
```

✓ 0.0s Python

```
+-----------------------------+--------+
|      Confusion Matrix       | Values |
+-----------------------------+--------+
|   Setosa (True Positive)    |   10   |
|   Setosa (False Negative)   |   0    |
|   Setosa (False Positive)   |   0    |
| Versicolor (False Positive) |   0    |
| Versicolor (True Positive)  |   9    |
| Versicolor (False Negative) |   0    |
| Virginica (False Positive)  |   0    |
| Virginica (False Negative)  |   0    |
|  Virginica (True Positive)  |   11   |
+-----------------------------+--------+
```

Figuur 18: Classificatie fitheidsmetrieken



```
# Training the classification model with the best hyperparameters
best_activation_function, best_hidden_layer_size, best_solver = best_hyperparameters
best_clf = MLPClassifier(hidden_layer_sizes=(best_hidden_layer_size,),
activation=best_activation_function, solver=best_solver, max_iter=2000, random_state=42)
best_clf.fit(X_train, y_train)

# Predictions for the testdata
y_pred = best_clf.predict(X_test)

# Calculate and print the error metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')  # 'weighted' is handy for multiclass classification
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
conf_matrix = confusion_matrix(y_test, y_pred)

# Create a list of error metrics and their corresponding values
metrics_values = [
    ["Accuracy", accuracy],
    ["Precision", precision],
    ["Recall", recall],
    ["F1-score", f1],
]
# Printing the tabel with classification metrics
print(tabulate(metrics_values, headers=["Error Metrics", "Values"], tablefmt="pretty"))
```

✓ 0.2s Python

```
+--------------+--------+
| Error Metrics | Values |
+--------------+--------+
|   Accuracy   |  1.0   |
|  Precision   |  1.0   |
|    Recall    |  1.0   |
|   F1-score   |  1.0   |
+--------------+--------+
```

Figuur 19: Classificatie fitheidsmetrieken

Model	MAE	MSE	R-squared
Regressie	0.0087246	0.0001608	0.9997699

```
# Execute regressiemetrics
best_activation_function, best_hidden_layer_size, best_solver = best_hyperparameters
best_reg = MLPRegressor(hidden_layer_sizes=(best_hidden_layer_size,), activation=best_activation_function, solver=best_solver, max_iter=2000, random_state=42)
best_reg.fit(X_train, y_train)

y_pred = best_reg.predict(X_test)

# Calculate the regressiemetrics based on the predictions
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Make a tabulate list of the regressionmetrics and their values
regression_metrics_values = [
    ["Mean Absolute Error (MAE)", mae],
    ["Mean Squared Error (MSE)", mse],
    ["R-squared (R^2)", r2],
]

# Printing the tabel with regression metrics
print(tabulate(regression_metrics_values, headers=["Regressie Metrieken", "Waarden"], tablefmt="pretty"))
```

✓ 0.0s

```
+--------------------------+------------------------+
|    Regressie Metrieken   |        Waarden         |
+--------------------------+------------------------+
| Mean Absolute Error (MAE) |  0.008724607985797736  |
| Mean Squared Error (MSE)  | 0.0001608022654144467 |
|      R-squared (R^2)      |   0.9997699172672925   |
+--------------------------+------------------------+
```

Figuur 20: regressie fitheidsmetrieken

7.1 Impact van Hyperparameteroptimalisatie

De uitkomsten (zie figuur 18, 19 & 20) na het uitvoeren van de code laten zien dat het genetische algoritme met succes optimale hyperparameters heeft geïdentificeerd en deze heeft toegepast op het neurale netwerk. Deze hyperparameters beïnvloeden de configuratie en werking van het neurale netwerk, zoals het aantal lagen, het aantal neuronen per laag, de activatiefunctie en de solver.

Het optimaliseren van deze hyperparameters is cruciaal, aangezien ze aanzienlijke invloed hebben op de prestaties van het neurale netwerk. Het vinden van de juiste hyperparameters kan leiden tot betere voorspellingen en een betere algehele modelprestatie.

De resultaten van het genetische algoritme zijn van belang omdat ze aantonen dat automatische hyperparameteroptimalisatie een waardevolle benadering is om de prestaties van neurale netwerken te verbeteren. Het stelt beoefenaars in staat om snel de optimale configuratie te vinden voor een specifieke taak, zonder de noodzaak van handmatige afstemming.

7.2 Evaluatie van de Modelprestaties

Om de impact van het genetische algoritme op de modelprestaties te beoordelen, hebben we gekeken naar verschillende evaluatie metrieken, zoals Mean Absolute Error (MAE), Mean Squared Error (MSE) en R-squared (R^2). Deze metrieken verschaffen inzicht in de nauwkeurigheid en voorspellende kracht van het neurale netwerk.

De bevindingen laten zien dat het gemodelleerde neurale netwerk met geoptimaliseerde hyperparameters betere prestaties heeft geleverd in vergelijking met eerdere configuraties. Een lage MAE geeft aan dat het model dicht bij de werkelijke waarden voorspelt, terwijl een lage MSE wijst op kleinere fouten in de voorspellingen. Bovendien geeft een hoge R^2-waarde aan dat het model goed past bij de gegevens.

Deze evaluatie metrieken bevestigen dat het genetische algoritme inderdaad heeft bijgedragen aan een verbetering van de modelprestaties. Het neurale netwerk kan nu nauwkeuriger en effectiever voorspellingen maken, wat van cruciaal belang is voor een breed scala aan toepassingen in machine learning.

7.3 Conclusie

Op basis van de resultaten van de code-uitvoering en de evaluatie van de modelprestaties kunnen we concluderen dat het genetische algoritme voor hyperparameteroptimalisatie met succes heeft bijgedragen aan de verbetering van het neurale netwerk. Dit benadrukt het potentieel van geautomatiseerde hyperparameteroptimalisatie als een waardevolle benadering in machine learning.

Het vermogen om snel optimale hyperparameters te identificeren en toe te passen, resulteert in meer nauwkeurige voorspellingen en verbeterde prestaties van neurale netwerken. Voor toekomstige projecten en onderzoek biedt dit inzicht in het potentieel van genetische algoritmen en geautomatiseerde hyperparameteroptimalisatie in machine learning. Verder onderzoek kan zich richten op het verkennen van geavanceerdere technieken en strategieën om de prestaties nog verder te verbeteren.

Bibliografie

Akshita, C. (2020, December 8). *MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better?* Retrieved from medium: https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e

Desjardins, B., Falcon, R., Abielmona, R., & Petriu, E. (2017). Planning Robust Sensor Relocation Trajectories for a Mobile Robot with Evolutionary Multi-objective Optimization. doi:10.1007/978-3-319-47715-2_8.

Dutta, A. (2023, Maart 10). *Crossover in Genetic Algorithm*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/

Itano, F., Angelo de Abreu de Sousa, M., & Del-Moral-Hernandez, E. (2018, Juli 13). Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm. Rio de Janeiro, Brazilie. doi:10.1109/IJCNN.2018.8489520

Russel, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach.* Pearson Education Limited.

Sadawi, N. (2015, Maart 8). Genetic Algorithms 21/30: Cycle Crossover. London, Engeland. Retrieved Oktober 23, 2023, from https://www.youtube.com/watch?app=desktop&v=DJ-yBmEEkgA

Tutorialspoint. (n.d.). *Genetic Algorithms - Crossover*. Retrieved from Tutorialspoint: https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm

Tutorialspoint. (n.d.). *Genetic Algorithms - Mutation*. Retrieved from Tutorialspoint: https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm

Xing, W., Zhang, J., Zou, Q., & Lin, J. (2021, November 16). *Application of Gauss Mutation Genetic Algorithm to Optimize Neural Network in Image Painting Art Teaching*. doi:10.1155/2021/3302617