# Samenvatting feedback van ML-Agents opdracht deel III

## De opdracht

De Unity opdracht dat door de studenten gemaakt moest worden, ging over het leren bouwen van een Reïnforcement Learning agent in een virtuele 3D omgeving die over een bepaalde object moest springen zonder dat object te raken.

De agent in de opdracht wordt voorgesteld als een groene kubus met als doel over blauwe blaken te springen door gebruik te maken van Ray perceptron sensors in een 360 graden omgeving om vanuit elke kant een object te kunnen anticiperen. In de opdracht moest er een uitbreiding gekozen worden door de studenten om meer uitdaging te creëren in dit project. In dit project wordt er gebruik van een kruispunt systeem waar er vanuit 4 verschillende wegen blauwe balken kunnen komen en richting de agent kunnen bewegen die in het midden gesitueerd staat.

## Tutorial

De eerste stap zou zijn om de git repo te clonen naar je eigen omgeving. Hieronder tonen we de bepaalde parameters die wij hebben gebruikt om het origineel model werkend te doen runnen:  
Behaviour parameters:  
Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving  
Ray perception sensor:  
Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

## Training van het algoritme

Voor de training van het Neuraal netwerk hebben wij 3 runs uitgevoerd om de ideale resultaat te vinden. We kozen maar 3 runs omdat wij toch de effecten van bepaalde parameters wouden zien of bepaalde aanpassingen een betere resultaten zouden geven. Omdat de leerproces vrij lang en intensief was, werd er gekozen voor 3 runs maar. We hebben als eerst de originele werkende run die wij uiteindelijk zouden vergelijken met de wijziging in lengte van de Ray sensor length en als andere run de wijziging in hyperparameter learning rate om de prestaties een beetje te vergelijken.

|  |  |  |  |
| --- | --- | --- | --- |
| **Run id** | **Ray sensor length** | **Learning rate** | **Geaccumuleerde beloningen** |
| **1**  (origineel) | 63 | 0.0003 | 0.975 |

Afbeelding met tekst, schermopname, Perceel, diagram

Automatisch gegenereerde beschrijving

Hier in run 2 proberen wij door de lengte van de ray sensors te zien of er beter gepresteerd wordt of niet vergeleken met run 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Run id** | **Ray sensor length** | **Learning rate** | **Geaccumuleerde beloningen** |
| **2** | 15 | 0.0003 | 0.975 |

Afbeelding met tekst, schermopname, Perceel, diagram

Automatisch gegenereerde beschrijving

Hier in run 3 proberen wij door de hyperparameter Learning rate aan te passen te zien of het leerproces beter presteert dan run 2.

Afbeelding met tekst, schermopname, Perceel, diagram

Automatisch gegenereerde beschrijving

|  |  |  |  |
| --- | --- | --- | --- |
| **Run id** | **Ray sensor length** | **Learning rate** | **Geaccumuleerde beloningen** |
| **3** | 15 | 0.001 | 0.6367 |

## Resultaten van de training

Hieronder zijn wij een grafiek die de trainingsproces van 3 runs voorstelt en de geaccumuleerde beloningen laat zien.

Afbeelding met tekst, schermopname, diagram, lijn

Automatisch gegenereerde beschrijving

We zien in het algemeen toch een verschil tussen de 3 runs in prestatie. Als wij run 1 met run 2 vergelijken zien wij dat de geaccumuleerde beloning hetzelfde is, maar toch run 2 sneller tot een stabieler value is terecht gekomen. Het verschil voornamelijk tussen deze runs is dat bij run 2 de lengte in ray sensoren kleiner is gezet wat een succesvollere resultaat gaf als run 1.

Als wij run 3 bekijken , kunnen wij zien dat in het algemeen dit het slechtste beloning heeft. We zien dat het vanaf een bepaald sneller accurater wordt dan run 1, maar nog steeds niet beter presterend.

## Conclusie

De run met in het algemeen het beste resultaten is Run 2. Het heeft het snelst tot een betere beloning gekomen en presteert het hoogst. Door vervolgens in Run 3 met dezelfde Ray sensor lengte de hyperparameter voor de learning rate aan te passen zien wij dat de learning niet beter wordt.