Reinforcement Learning en Computerspellen

Hoe beïnvloeden de specifieke kenmerken van computerspellen de effectiviteit van specifieke reinforcement learning-algoritmes?

Matthijs Gorter
Thom Brinkhorst
Pepijn van Iperen

Profielwerkstuk
onder begeleiding van
S. Rook
Christelijk Lyceum Zeist
Natuur en Techniek
Februari 2025

Voorwoord

Voorwoord inhoud hier. Bedank mensen die geholpen hebben, beschrijf het doel van het profielwerkstuk en eventuele persoonlijke motieven of ervaringen.

Matthijs Gorter, Thom Brinkhorst, Pepijn van Iperen
Christelijk Lyceum Zeist
Februari 2025

Notatie

Variabele	**Definitie**
t	Tijdstap
T	Laatste tijdstap van een episode (horizon)
x	Toestand (state)
x_t	Toestand op tijdstip t
x'	Toestand een tijdstap na x
$\mathcal{X}$	Set van alle toestanden
a	Actie
$\mathcal{A}$	Alle mogelijke acties
a_t	Actie op tijdstip t
r	Beloning (reward)
$\mathcal{R}$	Set van mogelijke beloningen
r_t	Beloning op tijdstip t
$r(x, a)$	Beloningsfunctie
μ	Deterministisch beleid
π	Stochastisch beleid
π^*	Optimale stochastisch beleid
γ	Kortingsfactor tussen 0 en 1
$p(x'\|x, a)$	Overgangswaarschijnlijkheidsfunctie
$\mathcal{P}$	Overgangswaarschijnlijkheidsmatrix
$V(x)$	Waardefunctie
$Q(x, a)$	Q-functie
$Q^*(x, a)$	Q-functie met het optimale beleid
$\mathbb{E}[X]$	Verwachtingswaarde van variabele X
$\mathbb{E}[a\|b]$	Geconditioneerde verwachtingswaarde
$\mathbb{E}_\pi[X]$	Verwachtingswaarde als beleid π wordt gevolgd

Tabel 1: Notatie

Inhoudsopgave

Hoofdstuk 1

Inleiding

Introductie Onderwerp
Onderwerpkeuze verantwoorden
Onderzoeksvraag/Hoofdvraag met eventuele hypothese
Deelvragen
Wat zijn de specifieke kenmerken van verschillende soorten computerspellen?
Welke reinforcement learning-algoritmes zijn beschikbaar en wat zijn hun kenmerken?
Hoe beïnvloeden de spelkenmerken de prestatie van reinforcement learning-algoritmes?
klein stukje theorie als inleiding op het theorie-onderdeel
werkplan in grote lijnen, opbouw van verslag

1.1 Doel van het onderzoek

Het doel van dit onderzoek is om te begrijpen hoe de kenmerken van verschillende computerspellen de effectiviteit van verschillende reinforcement learning (RL) algoritmes beïnvloeden bij het verbeteren van spelprestaties. Dit onderzoek richt zich op het identificeren van de eigenschappen van verschillende soorten spellen en de kenmerken van RL-algoritmes.

Door verschillende RL-algoritmes toe te passen op een reeks spellen met verschillende kenmerken, willen we ontdekken welke algoritmes het beste presteren in welke soorten spellen. Dit kan variëren van strategische spellen die planning vereisen tot actiespellen die snelle beslissingen vragen.

1.2 Onderzoeksvragen

Hoofdvraag

Hoe beïnvloeden de specifieke kenmerken van computerspellen de effectiviteit van verschillende reinforcement learning-algoritmes in het optimaliseren van spelprestaties?

Deelvragen

Om beter te begrijpen hoe de kenmerken van computerspellen de prestaties van verschillende reinforcement learning (RL) algoritmes beïnvloeden, hebben we drie belangrijke deelvragen opgesteld

1. **Wat zijn de specifieke kenmerken van verschillende soorten computerspellen?**

   Deze vraag richt zich de eigenschappen van verschillende soorten computerspellen. Spellen kunnen sterk verschillen in hoe ze zijn opgebouwd, hoe snel spelers beslissingen moeten nemen en hoe complex de spelregels zijn. Door deze kenmerken te onderzoeken, kunnen we inzicht krijgen in welke aspecten van een spel een uitdaging vormen voor RL-algoritmes.

2. **Welke reinforcement learning-algoritmes zijn beschikbaar en wat zijn hun kenmerken?**

   Hier willen we kijken naar de verschillende soorten RL-algoritmes die beschikbaar zijn en wat hen uniek maakt. Sommige algoritmes zijn beter in het leren van eenvoudige taken, terwijl andere juist goed zijn in het omgaan met complexe situaties.

3. **Hoe beïnvloeden de spelkenmerken de prestatie van reinforcement learning-algoritmes?**

   Deze vraag gaat in op het belangrijkste deel van het onderzoek: het verband tussen de kenmerken van een spel en hoe goed een RL-algoritme presteert. We willen weten hoe bepaalde eigenschappen van een spel, zoals de noodzaak voor snelle beslissingen of lange-termijnplanning, invloed hebben op de effectiviteit van een algoritme. Door de prestaties van verschillende algoritmes in verschillende spellen te vergelijken, kunnen we ontdekken welke het beste werken voor bepaalde soorten spellen en waarom dat zo is.

1.3 Experimenten

In dit onderzoek willen we kijken hoe verschillende reinforcement learning (RL) algoritmes presteren in verschillende computerspellen. We hebben drie spellen gekozen: Snake, Schaken, en Mario Super Bros. Elk spel heeft zijn eigen kenmerken en uitdagingen, en we zullen drie RL-algoritmes testen: Deep Q-Network (DQN), Proximal Policy Optimization (PPO), en AlphaZero (of Deep Deterministic Policy Gradient (DDPG) als AlphaZero niet lukt). Het doel is om te ontdekken hoe goed elk algoritme werkt in elk spel en hoe de kenmerken van het spel de prestaties van de algoritmes beïnvloeden.

1. **Snake**

   Snake is een simpel actiespel waar je een slang bestuurt die appels eet om langer te worden, terwijl je ervoor zorgt dat hij zichzelf niet raakt. Het spel vereist snelle beslissingen en het vooruitzicht zodat je je zelf niet opsluit. Het doel van het experiment met Snake is om te onderzoeken hoe de RL-algoritmes presteren in een omgeving met beperkte ruimte en snel veranderende situaties. We zullen kijken hoe snel elk algoritme leert, hoe stabiel de prestaties zijn en wat de uiteindelijke score is.

2. **Schaken**

   Schaken is een complex bordspel met een grote hoeveelheid mogelijke zetten en uitkomsten. Dit experiment is bedoeld om te bepalen hoe de RL-algoritmes omgaan met de grote hoeveelheid mogelijkheden en de noodzaak om lange-termijnstrategieën te ontwikkelen. We zullen meten hoe snel de algoritmes leren, hoe goed de strategieën zijn die ze ontwikkelen (gecontroleerd door een schaakprogramma), en hoe consistent de prestaties zijn in verschillende schaaksituaties.

3. **Mario Super Bros**

   Mario Super Bros, is een platformspel waarin de speler een personage bestuurt dat door verschillende levels moet navigeren, obstakels moet vermijden en vijanden moet verslaan. Dit spel combineert elementen van actie en planning, met veel variatie in speelomgevingen en uitdagingen. Het doel van het experiment met Mario Super Bros is om te onderzoeken hoe de RL-algoritmes presteren in een dynamische omgeving waar zowel snelheid als strategie nodig zijn. We zullen beoordelen hoe snel de algoritmes leren, hoe stabiel de prestaties zijn, en hoeveel levels succesvol worden voltooid.

1.4 Hypothese

We verwachten dat:

1. Deep Q-Network het beste zal presteren in Snake omdat het algoritme snel kan leren in omgevingen met beperkte ruimte en snel veranderende situaties, waar directe beloningen een grote rol spelen.
2. Proximal Policy Optimization zal beter presteren in Mario Super Bros, omdat dit algoritme geschikt is voor dynamische omgevingen en situaties waar zowel snelheid en planning belangrijk zijn.
3. AlphaZero zal beter zijn in Schaken, vanwege het planning en lange-termijnstrategie die nodig zijn.

1.5 Relevantie van het Onderzoek

Dit onderzoek laat effectiviteit van reinforcement learning algoritmes in verschillende omgevingen laat zien, wat bijdraagt aan het beter gebruik van AI-systemen. Deze kennis kan niet alleen worden toegepast binnen de game-industrie, maar ook in andere sectoren zoals de gezondheidszorg, zelfrijdende auto's en robotica.

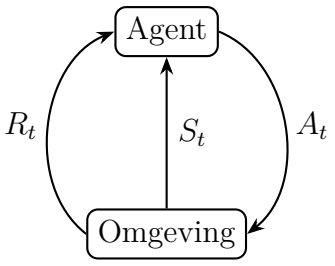
Hoofdstuk 2

Theoretisch Kader

2.1 Definitie

Reinforcement Learning (RL) is een tak binnen kunstmatige intelligentie waarin een agent leert door interactie met zijn omgeving. Een agent is een entiteit die leert en acties onderneemt. Bij een zelfrijdende auto is het besturingssysteem de agent, en bij een schaakspel is de schaker de agent. De omgeving is alles waarmee de agent interageert en die reageert op de acties van de agent. Bij een zelfrijdende auto is dit de weg waar de auto op rijdt en de voertuigen om de auto heen. Bij een schaakspel is dit het schaakbord.

De agent leert door interactie met zijn omgeving. De agent ontvangt beloningen of straffen (negatieve beloningen) als gevolg van zijn acties. Het doel van de agent is om een strategie te ontwikkelen die de cumulatieve beloning maximaliseert over tijd. Bij Super Mario Bros (1985) is Mario de agent en de agent krijgt een beloning als Mario richting de eindslag beweegt (meestal naar rechts) en als Mario een coin of een power up oppakt en Mario krijgt een straf als Mario sterft en hij krijgt elke seconde straf zodat hij zo snel mogelijk het level wilt voltooien.



Figuur 2.1: RL model tussen agent en omgeving.

Een actie naar de beslissing die een agent neemt bij elke stap in een besluitvormingsproces. Acties worden aangeduid met a en worden gekozen uit een reeks mogelijke acties $\mathcal{A}$. Elke door de agent genomen actie beïnvloedt de interactie met de omgeving, wat leidt tot een verandering in de toestand en een daaruit voortvloeiende beloning.

Een toestand x vertegenwoordigt de huidige situatie of staat van de omgeving waarin de agent opereert. Dit wordt aangeduid met x en maakt deel uit van de toestandsruimte $\mathcal{X}$. Bij de aanvangsstap $t = 0$, begint de agent in een initiële toestand x_0 die willekeurig wordt bepaald door een verdeling p. Naarmate het proces vordert, bevindt de agent zich in nieuwe toestanden gebaseerd op zijn acties.

Een beloning r is een feedbackwaarde die wordt ontvangen nadat de agent een actie heeft uitgevoerd in een bepaalde toestand. Deze beloning wordt bepaald door de beloningsfunctie $r(x, a)$. De beloningsmatrix R bevat de onmiddellijke beloningen voor elke combinatie van toestand en actie.

Een overgang beschrijft de verandering van de huidige toestand naar de volgende toestand als gevolg van een actie die door de agent wordt genomen. De waarschijnlijkheid van overgang wordt bepaald door de overgangswaarschijnlijkheidsfunctie $p(x'|x, a)$, die afhangt van de huidige toestand x, de genomen actie a en leidt tot een nieuwe toestand x'. De overgangswaarschijnlijkheidsmatrix P bevat de waarschijnlijkheden van het overgaan van de ene toestand naar de volgende toestand, gegeven een bepaalde actie.

Markov-eigenschap

MDP werkt onder de Markov-aanname, wat betekent dat de volgende toestand en beloning alleen afhangen van het huidige toestand-actiepaar en niet van enige eerdere geschiedenis. Deze eigenschap vereenvoudigt het besluitvormingsmodel door zich alleen te concentreren op de huidige situatie.

Voorbeeld van een MDP:

- **Snake**: De toekomstige toestand (positie van de slang en voedsel) is volledig bepaald door de huidige toestand (huidige positie en locatie van het voedsel) en de actie (richting van beweging) zonder afhankelijk te zijn van de geschiedenis van eerdere bewegingen.

Voorbeeld van geen MDP:

- **Poker**: De beslissingen in poker zijn afhankelijk van niet alleen de huidige hand, maar ook van de geschiedenis van inzetten en het gedrag van andere spelers in vorige rondes.

Voorbeeld van een twijfelgeval:

- **Schaak**: Elke positie op het bord (toestand) en mogelijke zetten (acties) bepalen de volgende positie, maar strategieën kunnen afhankelijk zijn van eerdere zetten.

In een MDP gaat een agent verder in tijdstappen $t = 0, 1, 2, \ldots, T$ waar de horizon T zowel eindig als oneindig kan zijn. Hierin is T oneindig tenzij anders aangegeven.

Overgangswaarschijnlijkheidsmatrix

Wanneer de set van alle toestanden $\mathcal{X}$ en de set van alle acties $\mathcal{A}$ eindig zijn, d.w.z. $|\mathcal{X}|, |\mathcal{A}| < \infty$, kan de overgangswaarschijnlijkheidsfunctie $p(x'|x, a)$ worden weergegeven als een overgangsmatrix.

De grootte van de overgangswaarschijnlijkheidsmatrix is $|\mathcal{X}| \times |\mathcal{X}| \times |\mathcal{A}|$. Dit is een driedimensionale matrix. De dimensies zijn als volgt:

- De huidige toestand x.
- De genomen actie a.
- De volgende toestand x'.

Beleid

In RL is een beleid de strategie die een agent volgt om beslissingen te nemen. Het bepaalt welke actie een agent moet uitvoeren, gegeven de huidige toestand van de omgeving. Er zijn twee hoofdtypen beleid:

- **Deterministisch Beleid**:
  - **Formule:** $a_t = \pi(s_t)$
  - **Beschrijving:** Voor elke toestand s_t kiest het beleid altijd dezelfde actie a_t. Het resultaat is volledig voorspelbaar zolang de toestand bekend is.
- **Stochastisch Beleid**:
  - **Formule:** $a_t \sim \pi(\cdot|s_t)$
  - **Beschrijving:** Voor een gegeven toestand s_t kiest het beleid een actie a_t volgens een waarschijnlijkheidsverdeling. Dit betekent dat de actie die wordt gekozen afhankelijk is van kans, wat leidt tot variabiliteit in het gedrag van de agent.

Verwachtingswaarde

De verwachtingswaarde $\mathbb{E}[X]$ (Expected value), of het gemiddelde, van een willekeurige variabele X is een manier om het gemiddelde resultaat te berekenen dat je zou verwachten als je een groot aantal experimenten uitvoert. Bijvoorbeeld, als X een dobbelsteenworp vertegenwoordigt, dan is $\mathbb{E}[X]$ het gemiddelde van de uitkomsten 1, 2, 3, 4, 5, en 6, wat gelijk is aan 3,5.

Geconditioneerde verwachtingswaarde $\mathbb{E}[a|b]$: het gemiddelde van a berekenen, gegeven de voorwaarde b.

Kortingsfactor

De kortingsfactor γ is een getal tussen 0 en 1 dat het gewicht bepaalt van toekomstige beloningen ten opzichte van onmiddellijke beloningen. Bij een lage kortingsfactor hebben toekomstige beloningen weinig invloed. Bij een kortingsfactor van 1 hebben alle beloningen evenveel invloed.

Waardefunctie

De waardefunctie $V(x)$ geeft de verwachte waarde van de totale beloning die een agent zal ontvangen vanaf de toestand x als hij het beleid π volgt.

$$V(x) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = x \right]$$

Hierin is $\sum_{t=0}^{\infty} \gamma^t r_t$ de som van alle beloningen waarbij elke beloning wordt vermenigvuldigd met γ^t als de kortingsfactor ($\gamma \in [0, 1]$). Dan wordt elke beloning met een groter tijdstapje (verder in de toekomst) kleiner en heeft dus minder invloed op de verwachte waarde.

Q-functie

De Q-functie $Q(x, a)$ geeft de verwachte waarde van de totale beloning die een agent zal ontvangen vanaf de toestand x en na het nemen van actie a, als hij daarna het beleid π volgt.

$$Q(x, a) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = x, a_0 = a \right]$$

waarin $a_0 = a$ betekent dat de eerste actie a is.

De waardefunctie geeft aan hoe goed een bepaalde toestand is, terwijl de Q-functie aangeeft hoe goed een actie in een bepaalde toestand is.

Waarde functie en Q-functie zijn gerelateerd aan elkaar op deze manier:

$$V(x) = \mathbb{E}_{a \sim \pi(\cdot|x)}[Q(x, a)]$$

Bellman vergelijking

De Q-functie en de waardefunctie in RL moeten voldoen aan consistentievoorwaarden, bekend als de Bellman-vergelijkingen. Voor een gegeven beleid π, kunnen de waardefunctie $V(x)$ en de Q-functie $Q(x, a)$ als volgt worden gedefinieerd:

$$V(x) = \mathbb{E}_{a\sim\pi(\cdot|x),x'\sim p(\cdot|x,a)}[r(x, a) + \gamma V(x')]$$

$$Q(x, a) = \mathbb{E}_{x'\sim p(\cdot|x,a),a'\sim\pi(\cdot|x')}[r(x, a) + \gamma Q(x', a')]$$

waarin:

- $a \sim \pi(\cdot|x)$ betekent dat actie a de actie is volgens beleid π in toestand x.
- $x' \sim p(\cdot|x, a)$ betekent dat toestand x' volgt uit toestand x en actie a volgens de overgangswaarschijnlijkheidsfunctie p.
- $a' \sim \pi(\cdot|x')$ betekent dat de volgende actie a' de actie is volgens beleid π in de nieuwe toestand x'.
- $r(x, a)$ is de beloning van actie a in toestand x.

Evaluatie en Controle

Er zijn twee problemen in RL:

- **Evaluatie:** Het eerste probleem in RL is evaluatie. Het doel is om te berekenen hoe goed een bepaald beleid π presteert. Dit wordt gedaan door te kijken naar de waarde functie $V(x)$ of de Q-functie $Q(x, a)$.

  Waarom is evaluatie een probleem?

  - **Complexiteit van de omgeving:** In een dynamische omgeving kan het moeilijk zijn om te voorspellen welke beloningen een agent zal ontvangen vanuit een bepaalde toestand, vooral als de omgeving verandert zonder dat de agent verandert.
  - **Variabiliteit van beloningen:** Beloningen kunnen stochastisch zijn, wat betekent dat dezelfde actie in dezelfde toestand verschillende uitkomsten kan hebben.
  - **Lange termijn effecten:** Het effect van een bepaalde actie wordt pas later zichtbaar.

Het evaluatieprobleem wordt meestal beschouwd als een subroutine van het tweede probleem: controle.

- **Controle:** Bij controle is het doel om het beleid π te vinden dat de waarde functie maximaliseert over de initiële toestanden $x \sim \rho$.

$$\max_{\pi} \mathbb{E}_{x \sim \rho}[V(x)]$$

waarin $x \sim \rho$ betekent dat de toestand x van de agent komt uit een vooraf gedefinieerde verzameling toestanden, waarbij elke toestand een bepaalde kans heeft om gekozen te worden.

Waarom is controle een probleem?

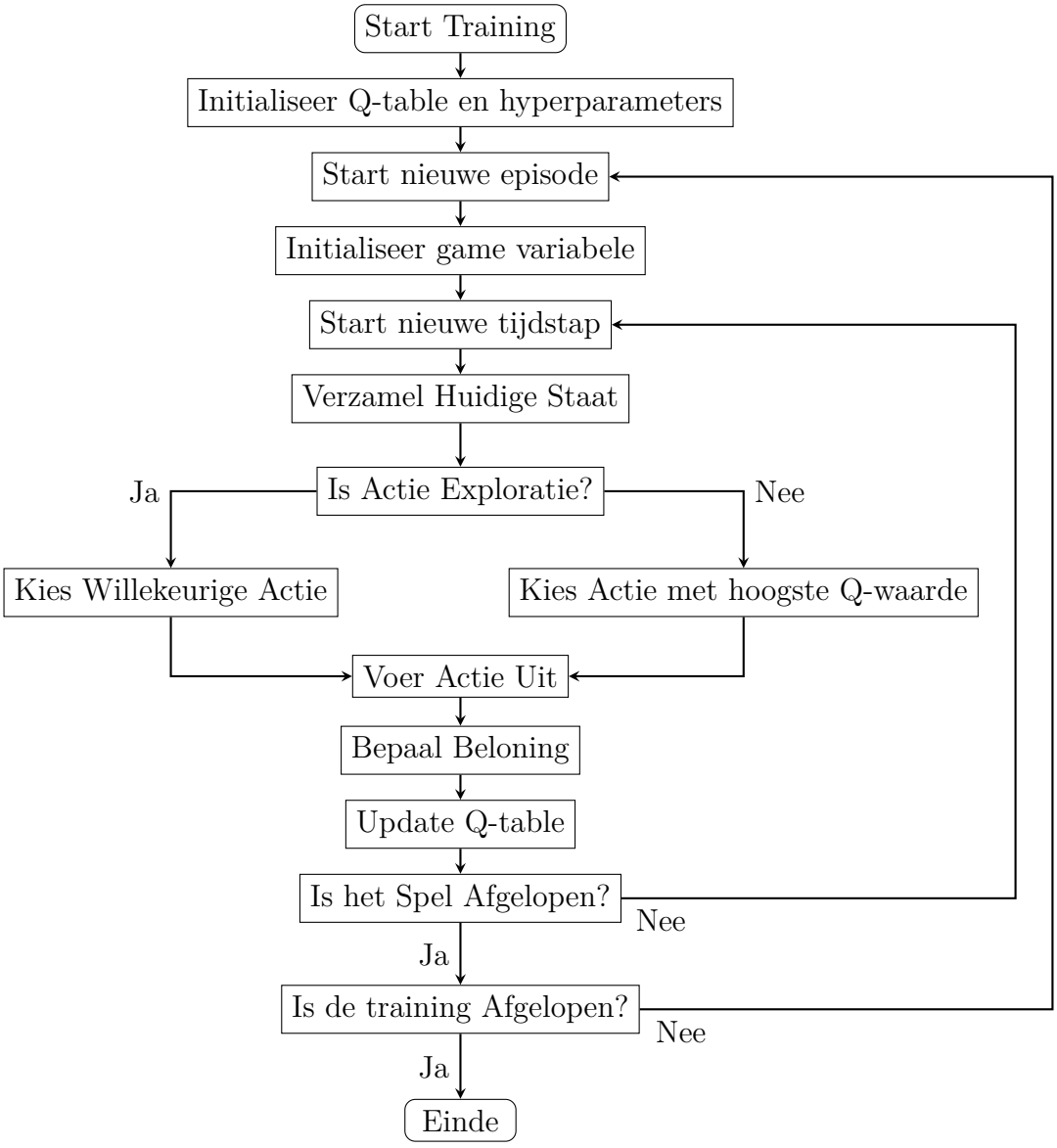
  - **Exploratie vs. exploitatie:** De agent moet een balans vinden tussen het verkennen van nieuwe acties om betere beloningen te ontdekken (exploratie) en het uitbuiten van bekende acties die momenteel de hoogste beloning bieden (exploitatie).
  - **Dimensionale complexiteit:** In veel RL-toepassingen zijn er een groot aantal toestanden en acties, wat het zoeken naar het optimale beleid computationeel duur maakt.

Het optimale beleid π^* is het beleid dat de verwachte cumulatieve beloning over tijd maximaliseert.

$$Q^*(x, a) = \mathbb{E}_{x' \sim p(\cdot|x,a)} \left[r(x, a) + \gamma \max_{a'} Q^*(x', a') \right]$$

2.2 Q-Learning

Het Q-learning algoritme, dat zowel in Algoritme 1 als in de flowchart in Figuur 2.2 is weergegeven, is een reinforcement learning algoritme dat gebruik maakt van een Q-tabel. Hierin de Q-waarden voor alle combinaties van toestanden en actie's. Met behulp van deze tabel kiest agent de volgende actie.



Figuur 2.2: Flowchart van het Q-Learning Algoritme

Algorithm 1 Q-Learning Algoritme

Initialisatie: Stel $Q(s, a)$ willekeurig in voor alle toestanden s en acties a
for elke episode **do**
  Initialiseer begin-toestand s
  while s is niet een terminale toestand **do**
    Kies actie a in toestand s op basis van een beleid π
    Voer actie a uit, observeer beloning r en de volgende toestand s'
    Update de Q-waarde:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

    $s \leftarrow s'$
  end while
end for

2.3 Deep Q-Network

2.4 DDPG

2.5 AplhaZero

,

2.6 Artificiële Leermethodes

2.7 Computerspellen

Hoofdstuk 3

Onderzoeksmethoden

3.1 Q-Learning

Hoofdstuk 4

Analyse en Resultaten

Hoofdstuk 5

Conclusie

Hoofdstuk 6

Discussie

Appendix

Bibliografie

Introduction to RL. (2018). https://spinningup.openai.com/en/latest/user/introduction.html

Millington, I., & Funge, J. (2009). *Artificial Intelligence for Games, Second Edition* (2nd). Morgan Kaufmann Publishers Inc.

Puterman, M. L. (1994, april). *Markov Decision processes*. https://doi.org/10.1002/9780470316887

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. A Bradford Book.

Tang, Y. (2021). *Reinforcement Learning: New Algorithms and An Application for Integer Programming* (tech. rap.).