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

Toen we begonnen na te denken over een onderwerp voor ons profielwerkstuk, wilden we graag een thema kiezen dat zowel uitdagend als actueel was. Kunstmatige intelligentie fascineert ons al enige tijd, vooral vanwege de invloed die het heeft op onze toekomst en de vele toepassingen die het nu al kent. Het idee om ons te verdiepen in reinforcement learning ontstond omdat deze tak van KI niet alleen theoretisch interessant is, maar ook praktisch ontzettend krachtig is.

Reinforcement learning staat aan de basis van indrukwekkende prestaties, zoals zelflerende spelprogramma's, geavanceerde robotsystemen en zelfrijdende auto's. De manier waarop een computer 'leert' door beloningen en straffen sprak ons aan, omdat het lijkt op hoe wij als mensen leren. Het leek ons daarom een perfecte uitdaging om dit complexe onderwerp te onderzoeken en te begrijpen hoe het precies werkt.

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

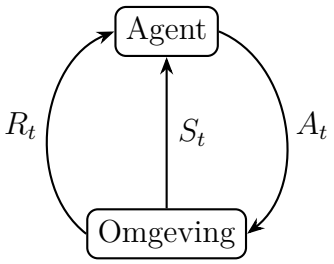
Inhoudsopgave

Hoofdstuk 1

Inleiding

Reinforcement Learning (RL) is een subdiscipline binnen de kunstmatige intelligentie (KI) die zich richt op het trainen van een agent om optimale acties te ondernemen binnen een specifieke 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.



Figuur 1.1: RL model tussen agent en omgeving.

Dit proces vindt plaats door middel van een vallen en opstaan aanpak, waarbij de agent beloningen ontvangt voor correcte acties en straffen voor incorrecte acties(negatieve beloningen). Het uiteindelijke doel is het maximaliseren van de cumulatieve beloning over tijd.

Computerspellen vormen een ideaal testplatform voor RL vanwege de veelzijdige uitdagingen die ze bieden, zoals dynamische omgevingen, complexe regels en onvoorspelbare scenario's. RL heeft bewezen effectief te zijn in spellen met zowel discrete als continue actie- en toestandruimtes, variërend van actiespellen zoals Snake tot strategische spellen zoals Schaken. Het toepassen van RL in gaming vereist een diep begrip van zowel de kenmerken van de spellen als de algoritmes die worden ingezet.

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 een-

voudige 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 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.

1.4 Relevantie van het Onderzoek

Dit onderzoek laat effectiviteit van reinforcement learning algoritmes in verschillende omgevingen laat zien, wat bijdraagt aan het beter gebruik van KI-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

Reinforcement Learning (RL) opereert binnen het kader van Markov Decision Processes (MDP's), die een wiskundige basis bieden voor het modelleren van sequentiële beslissingsproblemen. Dit hoofdstuk bespreekt de fundamentele concepten en wiskundige formuleringen die ten grondslag liggen aan RL, gestructureerd rond vier kernonderdelen: de basiselementen van MDP's, de Markov-eigenschap en overgangsdynamiek, beleid en waarde-functies, en de Bellman-vergelijkingen.

2.1 Fundamentele Elementen van MDP's

2.1.1 Toestandsruimte

Laat ($\mathcal{X}$) de toestandsruimte zijn, waarbij elke toestand ($x \in \mathcal{X}$) de huidige situatie of staat is van de omgeving waarin de agent opereert. Op de aanvangsstap ($t = 0$) begint de agent in een initiële toestand (x_0). Naarmate het proces vordert, bevindt de agent zich in nieuwe toestanden gebaseerd op zijn acties.

2.1.2 Actieruimte

Laat ($\mathcal{A}$) de actieruimte zijn, waarbij elke actie ($a \in \mathcal{A}$) een mogelijke beslissing van de agent vertegenwoordigt. De interactie tussen de agent en de omgeving verloopt in discrete tijdstappen ($t = 0, 1, 2, \ldots, T$), waarbij de horizon (T) eindig of oneindig kan zijn.

2.1.3 Beloningsfunctie

De beloningsfunctie ($r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$) koppelt toestand-actieparen aan beloningen, waarbij ($r(x, a)$) de directe beloning vertegenwoordigt die wordt ontvangen na het uitvoeren van actie (a) in toestand (x).

2.2 Markov-eigenschap en Overgangsdynamiek

2.2.1 De Markov-eigenschap

Het onderscheidende kenmerk van MDP's is de Markov-eigenschap, die stelt dat de toekomstige toestand alleen afhankelijk is van de huidige toestand en actie, onafhankelijk van de geschiedenis:

$$p(x_{t+1}|x_t, a_t, x_{t-1}, a_{t-1}, \ldots, x_0, a_0) = p(x_{t+1}|x_t, a_t)$$

Voorbeeld van het Markov-eigenschap:

- **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 Markov-eigenschap:

- **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.

2.2.2 Overgangswaarschijnlijkheidsfunctie

Voor eindige toestands- en actieruimten ($|\mathcal{X}|, |\mathcal{A}| < \infty$) worden de overgangsdynamieken beschreven door een waarschijnlijkheidsfunctie ($p : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$), waarbij ($p(x'|x, a)$) de waarschijnlijkheid vertegenwoordigt om over te gaan naar toestand (x') gegeven de huidige toestand (x) en actie (a).

2.3 Beleid en Verwachte Waarden

2.3.1 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. Een beleid in reinforcement learning kan op twee manieren worden gedefinieerd:

- **Deterministisch Beleid:**
  $\pi : \mathcal{X} \to \mathcal{A}$, waarbij $a_t = \pi(x_t)$
  Voor elke toestand x_t schrijft het beleid exact één actie a_t voor.
- **Stochastisch Beleid:**
  $\pi : \mathcal{X} \times \mathcal{A} \to [0, 1]$, waarbij $\pi(a|x)$ de waarschijnlijkheid geeft van het kiezen van actie a in toestand x
  Voor een gegeven toestand x definieert het beleid een waarschijnlijkheidsverdeling over mogelijke acties.

2.3.2 Verwachte Waarden

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. De conditionele verwachting ($\mathbb{E}[X|Y]$) geeft de verwachte waarde van (X) gegeven (Y).

2.3.3 Kortingsfactor

De kortingsfactor ($\gamma \in [0, 1]$) bepaalt het gewicht bepaalt van toekomstige beloningen ten opzichte van onmiddellijke beloningen, waarbij:

- ($\gamma = 0$) alleen directe beloningen hebben invloed
- ($\gamma = 1$) alle toekomstige beloningen zijn even invloedrijk
- ($0 < \gamma < 1$) een korting toepast op toekomstige beloningen

2.4 Waarde-functies en Bellman-vergelijkingen

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

2.4.1 Toestandswaarde-functie

De toestandswaarde-functie ($V^\pi : \mathcal{X} \to \mathbb{R}$) onder beleid ($\pi$) wordt gedefinieerd als:

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

Deze functie geeft de verwachte waarde van de totale beloning die een agent zal ontvangen vanaf de toestand x.

2.4.2 Q-functie

De Q-functie ($Q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$) onder beleid ($\pi$) wordt gedefinieerd als:

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

Deze functie geeft de verwachte waarde van de totale beloning die een agent zal ontvangen vanaf de toestand x en na het nemen van actie a.

2.4.3 Bellman-vergelijkingen

De Bellman-vergelijkingen drukken de recursieve relatie uit tussen waarden van opeenvolgende toestanden:

- Toestandswaarde Bellman-vergelijking:

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

- Actiewaarde Bellman-vergelijking:

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

De relatie tussen (V^π) en (Q^π) wordt gegeven door:

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

2.5 Leerparadigma's

2.5.1 Model-Based vs Model-Free Learning

Model-Based Learning

Bij model-based learning construeert de agent expliciet een model ($\hat{p}$) van de overgangsdynamiek en beloningsfunctie van de omgeving ($\hat{r}$). Het geleerde model benadert:

- Overgangsfunctie: $\hat{p}(x'|x, a) \approx p(x'|x, a)$
- Beloningsfunctie: $\hat{r}(x, a) \approx r(x, a)$

De agent kan dit model gebruiken voor:

- *Planning*: Het simuleren van trajecten zonder interactie met de omgeving.
- *Counterfactual reasoning*: Het evalueren van acties die niet daadwerkelijk zijn uitgevoerd.

Model-Free Learning

Model-free methoden leren waardefuncties of beleidsregels direct uit ervaring, zonder een expliciet model te construeren. Deze methoden werken door:

- Directe updates van waarde-schattingen.
- Beleidsverbetering op basis van gesamplede trajecten.
- Geen expliciete representatie van overgangsdynamiek.

2.5.2 On-Policy vs Off-Policy Learning

On-Policy Learning

On-policy methoden leren over het beleid dat momenteel wordt uitgevoerd. De update van de waardefunctie volgt:

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

waarbij π zowel:

- Het gedragsbeleid (dat ervaringen genereert) als
- Het doelbeleid (dat wordt geleerd) is.

Off-Policy Learning

Off-policy methoden leren over één beleid (π) terwijl een ander beleid (μ) wordt gevolgd. Dit omvat:

- *Gedragsbeleid* (μ): Gebruikt voor exploratie.
- *Doelbeleid* (π): Het beleid dat wordt geleerd.

De importance sampling ratio (ρ) corrigeert voor het verschil tussen de]beleidsregels:

$$\rho_t = \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}$$

Dit maakt het mogelijk om te leren over optimaal gedrag terwijl een exploratief beleid wordt gevolgd.

Theoretisch Kader

Definitie

Een actie is 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.

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

Hoofdstuk 3

Kenmerken van specifieke Algoritmes

3.1 Q-Learning

Q-learning, geïntroduceerd door Chris Watkins in 1989, is een van de belangrijkste vooruitgangen binnen reinforcement learning. Dit model-vrije, off-policy algoritme is een van de meest gebruikte algoritmen binnen reinforcement learning vanwege zijn eenvoud en effectiviteit.

3.1.1 Proces

Het proces van het Q-learning-algoritme, zoals weergegeven in **Algoritme 1** en de flowchart in **Figuur 3.1**, begint met het opstellen van een Q-tabel. Deze tabel bevat de Q-waarden voor alle combinaties van toestanden en acties. Aan het begin zijn alle waarden ingesteld op nul.

Vervolgens start het spel, waarbij de agent de volgende actie bepaalt. Hierbij heeft de agent twee mogelijkheden:

- Exploratie: De agent voert een willekeurige actie uit om nieuwe informatie te verkennen.
- Exploitatie: De agent selecteert een actie op basis van de bestaande Q-tabel, waarbij de actie met de hoogste Q-waarde in de huidige toestand wordt gekozen.

De keuze tussen exploratie en exploitatie wordt bepaald door de parameter ϵ. De kans dat de agent een willekeurige actie uitvoert (exploratie) is gelijk aan ϵ. Aan het begin van

de training is ϵ gelijk aan 1, en deze waarde neemt exponentieel af naarmate de training vordert. Tegen het einde van de training is ϵ vrijwel 0.

Nadat een actie is uitgevoerd, ontvangt de agent een beloning. Op basis van deze beloning wordt de Q-waarde voor de combinatie van de uitgevoerde actie en de huidige toestand bijgewerkt in de Q-tabel. Dit proces wordt herhaald totdat de training is voltooid.

3.1.2 Convergentie en Optimaliteit

3.1.3 Voordelen en Beperkingen

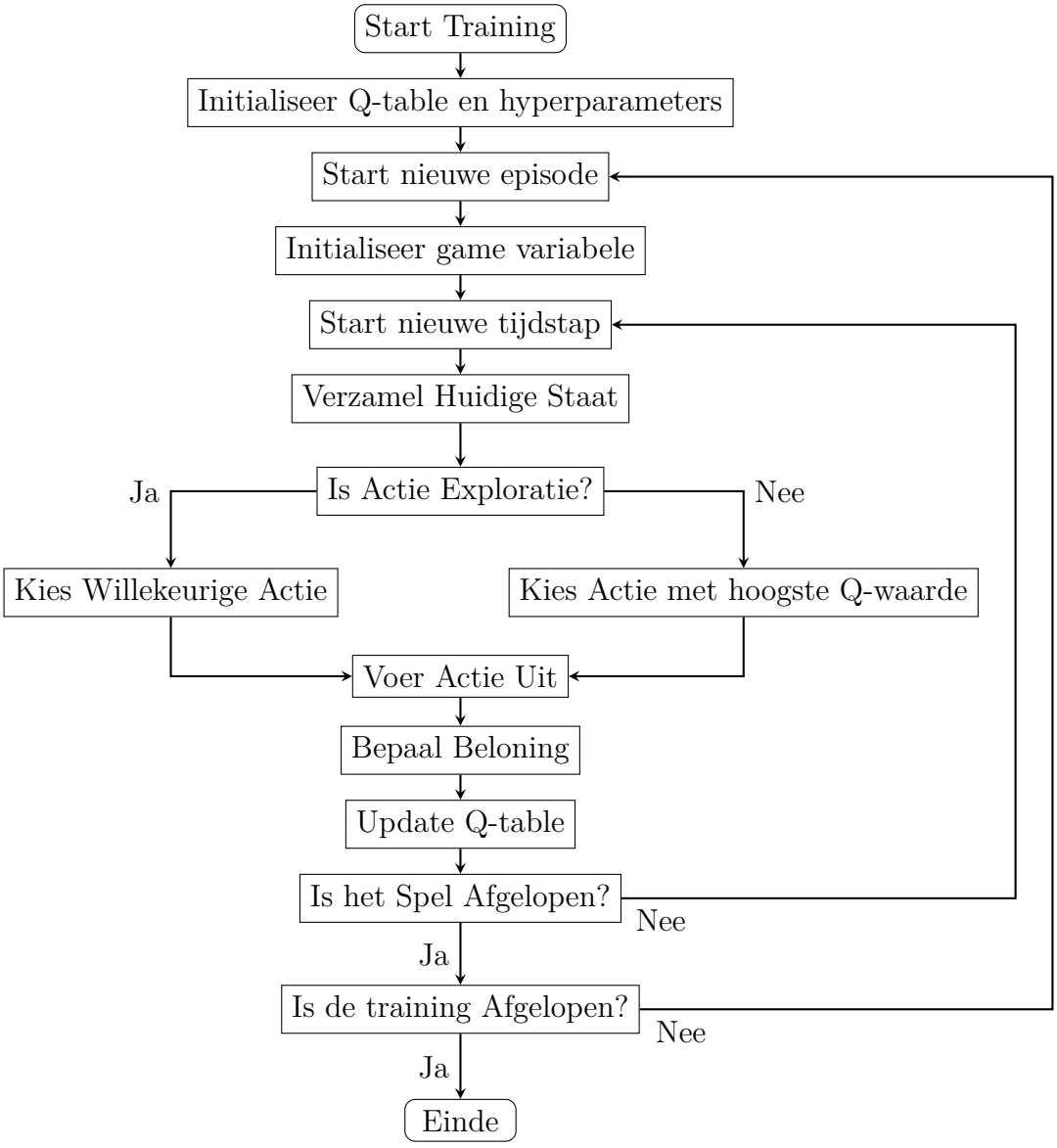
3.1.4 Toepassingen van Q-learning

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



Figuur 3.1: Flowchart van het Q-Learning Algoritme

3.2 Deep Q-Network

3.3 AplhaZero

Hoofdstuk 4

Kenmerken van specifieke Computerspellen

Er zijn talloze computerspellen met diverse en uitdagende omgevingen voor de toepassing van reinforcement learning (RL)-algoritmes. Spellen onderling kunnen erg van elkaar variëren onder andere in structuur, dynamiek, complexiteit, en tijdsduur. Al deze aspecten kunnen dergelijke invloed hebben op de effectiviteit van RL-algoritmes. Elk type spel stelt specifieke eisen aan een RL-algoritme, afhankelijk van aspecten zoals de omvang van de toestandsruimte, de regels en beperkingen, en de vereiste strategische vaardigheden.

In dit hoofdstuk worden de specifieke kenmerken van vier computerspellen met elkaar vergeleken: een *auto-racespel* waar de agent het autobesturingssysteem is, *Snake* waar de agent de slang is, *Schaken* waar de agent de schaker is en *Super Mario Bros* waar de agent Mario is. Een overzicht van alle speleigenschappen is te zien in Tabel ??.

4.1 Indeling en Strategische Diepgang van Spellen

Spellen kunnen worden ingedeeld op basis van hun genre en de mate van strategische diepgang die nodig is om succesvol te zijn. Actie- en platformspellen, zoals *Super Mario Bros*, hebben een gemiddelde strategische diepgang. Het doel is obstakels te overwinnen, vijanden te ontwijken of verslaan en tegelijkertijd gouden munten te verzamelen. Dit soort spellen vereist doorgaans korte termijn optimalisatie en directe reacties.

Strategiespellen, zoals *Schaken*, vragen daarentegen om diepgaande planning en vooruitdenken. Hier moet een speler of agent een reeks mogelijke toekomstige toestanden analyseren en anticiperen op de acties van een tegenstander. De strategische diepgang maakt leren complex, omdat beloningen vaak cumulatief en pas aan het einde van het spel duidelijk worden. Dergelijke spellen vereisen geavanceerde reinforcement learning (RL)-algoritmes die langetermijnplanning ondersteunen.

Puzzel-/actiespellen, zoals *Snake*, zijn minder afhankelijk van strategie. Hier draait het om patroonherkenning en korte termijn optimalisatie, waarbij eenvoudige RL-algoritmes voldoende zijn om succesvol te leren. Het doel is bijvoorbeeld appels te verzamelen zonder jezelf te raken, waarbij de beloningsstructuur rechtlijnig is.

Simulatie- en racegames, zoals een racespel met een zelfrijdende auto, richten zich op efficiënt en veilig navigeren over een parcours. Hier ligt de nadruk op het optimaliseren van gedrag in een gesimuleerde omgeving, vaak zonder de noodzaak van complexe planningsstrategieën.

Deze variatie in genres en strategische eisen bepaalt welk type RL-algoritme het meest geschikt is voor een spel. Complexere spellen met hogere strategische diepgang vereisen geavanceerdere algoritmes, terwijl eenvoudigere spellen vaak volstaan met directe responsmechanismen.

4.2 Indeling van Spellen

Super Mario Bros valt binnen het genre van actie- en platformspellen. Het doel van het spel is over hindernissen springen en vijanden te ontwijken en verslaan en tegelijkertijd gouden munten te verzamelen. *Schaken* daarentegen is een strategiespel, dat volledig turn-based is en gericht op denkvermogen, vooruitdenken en strategische planning. *Snake* wordt vaak als een puzzel-/actiespel beschouwd, waarbij het doel is een appel te eten terwijl je niet jezelf raakt; hier is patroonherkenning belangrijk. Een zelfrijdende auto in een racespel valt binnen het genre van simulatie en racegames. Het draait om het efficiënt en veilig navigeren over een parcours. Dit wordt vaak gebruikt bij offline racespellen waar je tegen de computer speelt.

4.3 Strategische Diepgang

De mate van strategische diepgang in een spel is een van de belangrijkste factoren die bepalen welk type RL-algoritme geschikt is. Strategie verwijst naar het vermogen om vooruit te denken en acties te plannen die op lange termijn voordelig zijn. Dit varieert sterk tussen spellen.

Strategische spellen, zoals *Schaken*, vereisen vereisen dat een agent ver vooruit denkt en een reeks mogelijke toekomstige toestanden analyseert. Hier is langetermijnplanning essentieel. De agent moet niet alleen rekening houden met de huidige toestand, maar ook anticiperen op de mogelijke acties van een tegenstander en de daaropvolgende uitkomsten. Bij strategische spellen is het leren complex, omdat beloningen vaak cumulatief en pas aan het einde van het spel duidelijk worden.

Aan de andere kant zijn er spellen zoals *Snake*, waarin strategie een veel minder be-

langrijke rol speelt. In deze spellen zijn acties vaak gebaseerd op eenvoudige regels en is de beste keuze meestal direct duidelijk. Het succes van een speler hangt hier voornamelijk af van korte termijn optimalisatie. Dergelijke spellen vereisen relatief eenvoudige RL-algoritmes, die zijn ontworpen om direct te reageren op beloningen of straffen zonder complexe planningsstrategieën. De eenvoudige structuur en beloningsmechanismen maken het leerproces rechtlijnig en efficiënt.

4.4 Beslissingsdynamiek en Tijdgevoeligheid

De snelheid waarmee de omgeving van een spel verandert, bepaalt in grote mate hoe moeilijk het is voor een RL-agent om effectief te leren en te reageren.

De beslissingsdynamiek verschilt sterk tussen de spellen. *Super Mario Bros* vereist snelle real-time beslissingen. De agent moet op het juiste moment springen of een vijand ontwijken, en timing is hierbij cruciaal. Bij *Schaken* is er juist geen tijdsdruk; de agent kan lang "nadenken" over elke zet. *Snake* zit er tussenin: hoewel het spel niet zo snel is als *Mario*, zit er wel een kleine tijdsdruk achter, maar dit is meestal verwaarloosbaar. Timing en patroonherkenning worden belangrijker naarmate het spel vordert. Bij een zelfrijdende auto in een racespel ligt de beslissingsdynamiek real-time. Hier zijn snelheid en precisie van beslissingen belangrijk, omdat milliseconden het verschil kunnen maken tussen een succesvolle race en een botsing.

4.5 Complexiteit

De complexiteit van de regels en doelen varieert sterk. *Super Mario Bros* heeft relatief eenvoudige regels: de agent moet vijanden vermijden en verslaan, munten verzamelen, en het einde van het level bereiken. De complexiteit van de vijanden en het terrein neemt echter toe naarmate de levels moeilijker worden. *Schaken* heeft relatief eenvoudige basisregels: zes verschillende stukken met elk unieke bewegingsmogelijkheden. Het doel, het schaakmat zetten van de tegenstander, vereist echter strategisch inzicht en planning. Dit maakt *Schaken* bijzonder uitdagend voor een RL-agent vanwege de enorme toestandsruimte en de langetermijnplanning die nodig is. *Snake* heeft zeer eenvoudige regels: de agent moet voedsel verzamelen en mag niet botsen met de muur of zichzelf. De uitdaging ligt in de toenemende snelheid en lengte van de slang. Een zelfrijdende auto in een racespel heeft daarentegen te maken met complexe regels die gebaseerd zijn op realistische fysica. Het doel is simpel: zo snel mogelijk de finish bereiken.

4.5.1 Regels en Beperkingen

De complexiteit en het aantal regels in een spel spelen een cruciale rol in de uitdaging die een RL-agent tegenkomt.

Spellen met veel regels en vaste patronen

Spellen zoals 4-op-een-rij of boter-kaas-en-eieren hebben een voorspelbare structuur en strikte regels. De mogelijke zetten en uitkomsten zijn beperkt, wat het spel eenvoudiger maakt om te modelleren. RL-algoritmes kunnen hier profiteren van waarschijnlijkheidsmodellen en gestructureerde planning. De voorspelbaarheid van deze spellen vermindert de onzekerheid in het leerproces. Een RL-agent kan relatief eenvoudig een optimale strategie leren door alle mogelijke acties te analyseren en te kiezen voor de meest belonende uitkomst.

Spellen met weinig regels en veel vrijheid

Spellen zoals GTA of Call of Duty bieden een grote mate van keuzevrijheid. De speler kan vrij bewegen in een open wereld, interacties aangaan en talloze acties uitvoeren. Deze spellen hebben een enorme toestandsruimte, die driedimensionaal en dynamisch is. Dergelijke spellen vereisen een flexibel en adaptief RL-algoritme. Het is onrealistisch voor een RL-agent om alle mogelijke acties en toestanden volledig te doorzoeken. Algoritmes zoals Proximal Policy Optimization (PPO) zijn hier geschikt. PPO gebruikt stochastische beleidsmodellen en leert door te experimenteren met acties, waarbij het snel aanpassingen kan maken op basis van feedback.

4.6 Dynamiek en Tijdgevoeligheid

De snelheid waarmee de omgeving van een spel verandert, bepaalt in grote mate hoe moeilijk het is voor een RL-agent om effectief te leren en te reageren.

4.6.1 Turn-based spellen

Spellen zoals *Schaken* of Monopoly bieden de speler voldoende tijd om de optimale actie te berekenen. Omdat de omgeving niet continu verandert, kan een RL-algoritme worden ingezet om een uitgebreide analyse te maken van alle mogelijke uitkomsten van een actie. Dit type algoritme is bijzonder effectief in spellen waar de agent kan profiteren van gestructureerde planning en voorspelbare omgevingen.

4.6.2 Realtime spellen

In spellen zoals *Mario Bros* of Tetris veranderen de omstandigheden continu. Obstakels bewegen, vijanden verschijnen en de tijdsdruk vereist snelle besluitvorming. Voor deze spellen zijn algoritmes nodig die snel leren en direct reageren, met neurale netwerken, waardoor het algoritme in real-time beslissingen kan nemen op basis van eerdere ervaringen.

4.7 Beloningsstructuur

Beloningen vormen de kern van RL en bepalen hoe een agent leert. De manier waarop beloningen worden toegekend, varieert sterk tussen spellen.

4.7.1 Directe beloningen

Spellen zoals *Snake* bieden onmiddellijke feedback. Elke actie resulteert direct in een beloning (zoals punten voor het eten van voedsel) of een straf (zoals botsingen). RL-algoritmes die afhankelijk zijn van directe beloningen werken goed in deze context, omdat ze snel leren welke acties voordelig zijn.

4.7.2 Cumulatieve beloningen

In strategische spellen zoals *Schaken* worden beloningen vaak pas aan het einde van het spel toegekend. Dit vereist dat de agent leert om acties te nemen die op lange termijn voordelig zijn. Het leren wordt complexer omdat de agent beloningen moet toeschrijven aan acties die mogelijk vele stappen eerder werden ondernomen.

Hoofdstuk 5

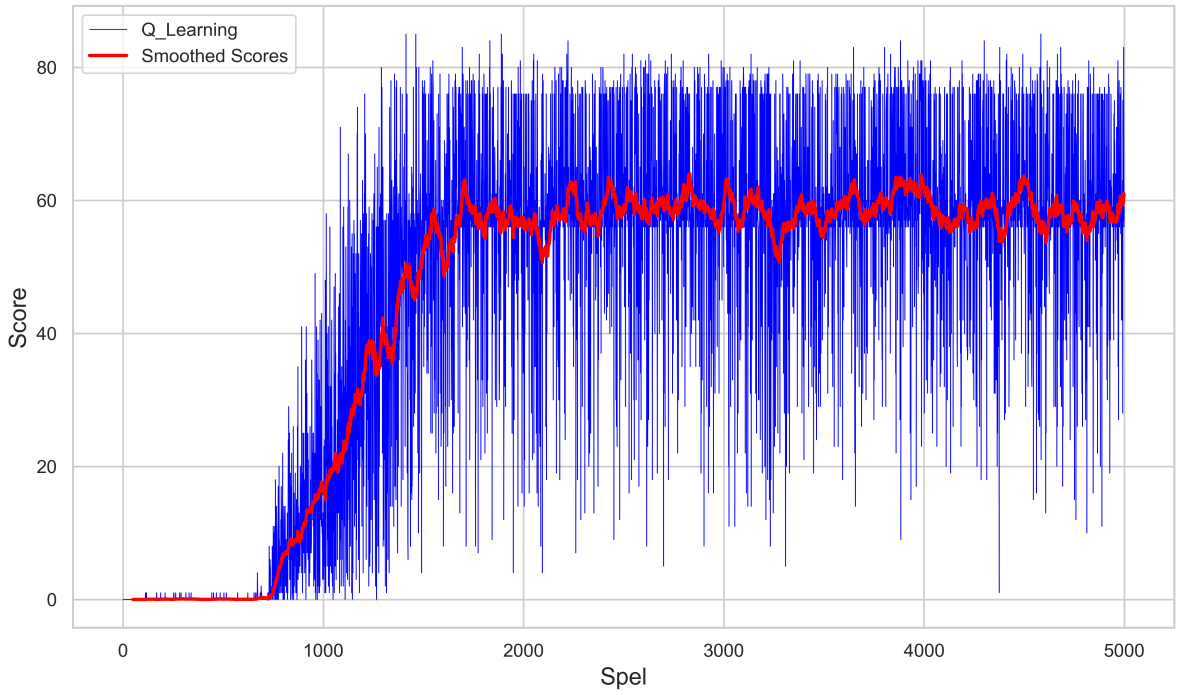
Onderzoeksmethoden

5.1 Q-Learning

Hoofdstuk 6

Analyse en Resultaten

6.1 Snake



Hoofdstuk 7

Conclusie

Hoofdstuk 8

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.).