Rapport: De beste vier op een rij strategie

Vince Ammerlaan, Tommi Lander, Nima Ghafar, Isa Dijkstra

Connect Four is een strategisch bordspel waarin twee spelers om de beurt schijven in een bord laten vallen. Het doel is om als eerste vier schijven op een rij te krijgen, horizontaal, verticaal of diagonaal.Het kan erg lastig zijn voor spelers om telkens de beste zet te maken. Omdat het spel meerdere mogelijke zetten en strategische varianten heeft, zien spelers niet altijd alle kansen of dreigingen. Dit kan leiden tot fouten: spelers missen soms een winnende zet of laten de tegenstander een winnende kans liggen.

# Doelstelling

**Onze doelstelling:**

We willen een systeem ontwikkelen dat automatisch de beste zet bepaalt zodat:

* Het risico op fouten wordt verkleind.
* Het spel strategisch wordt benaderd, zowel om te winnen als om te voorkomen dat de tegenstander wint.

**Waarom rule-based?**

Een rule-based systeem is geschikt omdat Connect Four duidelijke regels heeft, zoals:

* Spelers doen om de beurt een zet.
* Een schijf valt naar de laagste vrije plek in een kolom.
* Het doel is om vier op een rij te krijgen.

Met een rule-based systeem kun je deze regels programmeren zonder dat het systeem zelf hoeft te leren via data. Je maakt gewoon een set regels die het systeem volgt, zoals:

* Blokkeer de tegenstander als die bijna wint.
* Maak een winnende zet als dit mogelijk is.
* Neem het midden in beslag

**Rule based versus machine learning**

Voordelen van een rule-based system:

* Eenvoudig te begrijpen: De regels van Connect Four zijn helder en kunnen direct als logische voorwaarden worden geprogrammeerd.
* Snelle implementatie: Je hoeft geen grote dataset te verzamelen en te trainen zoals bij machine learning.
* Controleerbaarheid: Omdat het regels volgt die zelf zijn gemaakt, weet je waarom het systeem bepaalde beslissingen neemt.

Machine learning zou hier minder effectief zijn omdat:

* Veel data nodig: Machine learning vereist grote hoeveelheden data om effectief te zijn.
* Onvoorspelbaar: Het model mogelijk willekeurige beslissingen neemt zonder dat je weet waarom.
* Minder uitlegbaar: Het is lastiger om te begrijpen waarom het model een bepaalde zet kiest.

# Methodologie

**Ontwerp van de regels**

We hebben drie rule-based agents gemaakt en een random agent gemaakt. het rule-based systeem past regels in een vaste volgorde toe om de meest optimale zet te bepalen.

Hier onderleggen wij kort uit wat de vier verschillende "agents" doen en wat hun strategie is. Ook leggen we per "agent" uit wat de voor- en nadelen zijn.

**1. RandomAgent**

De RandomAgent is onze meest eenvoudige agent. Deze agent kiest willekeurig uit alle kolommen waar nog een zet mogelijk is. Deze agent vormt een handige baseline om andere agents mee te vergelijken. Dit doen we door elke andere "agents" tegen deze te laten spelen om te bepalen welke het beste werkt.

Hoe werkt het?

* De agent controleert welke kolommen nog beschikbaar zijn.
* Uit deze kolommen kiest hij willekeurig één zet.

Deze random agent is snel en eenvoudig te implementeren. Het doel is vooral om te testen of het spelmechanisme werkt en of agents correct reageren. Ook wordt dit gebruikt als baseline.

Voordelen:

* Simpel en robuust.
* Werkt altijd, ongeacht de situatie.
* Snelle beslissingen en weinig rekentijd nodig.

Nadelen:

* Geen strategie.
* Makkelijk te verslaan door elke andere agent met strategie.
* Geen aanpassingsvermogen.

**2. MinMaxAgent**

De MinMaxAgent gebruikt een bijzondere zoekstrategie: Het heet de MinMaxAgent omdat het gebruik maakt van een MinMax-algoritme. Dit zoekt de best mogelijke zet door vooruit te denken en het spel te simuleren. Het kiest de zet die de kansen Maximaliseert en de tegenstander minimaliseert. Het gebruikt 'alpha-beta pruning' wat onnodige taken overslaat.

Hoe werkt het?

* De agent speelt het spel al meerdere zetten vooruit.
* Hij kiest de zet die zijn kansen maximaliseert en de tegenstander minimaliseert.
* Dankzij alpha-beta pruning worden onnodige takken in de overgeslagen.
* Iteratieve verdieping laat de agent eerst snel schattingen maken en hierna zoekt hij steeds dieper naar betere zetten.
* Belangrijke zetten hebben een hogere prioriteit zoals zetten in het midden van het bord.

Voordelen:

* Sterke agent, moeilijk te verslaan.
* Past zich aan op basis van tijd.
* Slimme strategie en doet aan blokkades.
* Efficiënt dankzij alpha-beta pruning.

Nadelen:

* Verbruikt veel rekentijd bij een voller bord.
* In situaties met veel goede zetten kan de agent fouten maken.

**3. DoubleMoveAgent**

Deze DoubleMoveAgent probeert in Vier op een rij strategisch dubbele zetten te creëren en tegelijkertijd de tegenstander te blokkeren. Dit is hoe de agent werkt:

Hoe werkt het?

* De agent probeert eerst een dubbele zet te maken door in dezelfde kolom twee opeenvolgende stenen te plaatsen, zodat er direct winst ontstaat.
* Als het creëren van een dubbele zet niet direct mogelijk is, probeert de agent de tegenstander te blokkeren
* Als het blokkeren niet nodig is, probeert de agent een andere dubbele zet te creëren door:

Dubbele dreigingen is een goede tactiek omdat ze de tegenstander onder druk zetten. Deze strategie is eenvoudiger te implementeren dan MinMax en is beter tegen zwakke agents.

Voordelen:

* De agent simuleert dubbele zetten in dezelfde kolom en kan zo snel winnen als de situatie dat toelaat.
* Door te controleren op mogelijke winnende zetten van de tegenstander en deze direct te blokkeren, verkleint de agent de kans op verrassende overwinningen door de tegenstander.

Nadelen:

* Minder effectief als er weinig ruimte is voor dubbele dreigingen.
* Voorspelbaar door vaste regels.
* Werkt slecht tegen slimme agents.

**4. GreedyAgent**

De GreedyAgent is een makkelijke en logsiche agent die alleen kijkt naar directe resultaten. In plaats van vooruit te denken kiest hij tijdens de zet de beste zet.

Hoe werkt het?

* De agent bekijkt alle mogelijke zetten.
* Voor elke zet bekijkt hij de volgende bord toestand.
* De zet met de hoogste directe score wordt gekozen.

Deze strategie is snel en beter dan puur willekeurig kiezen. De agent neemt beslissingen gebaseerd op een simpele evaluatie zonder verdere voorspellingen.

Voordelen:

* Sneller dan MinMax dus minder rekenkracht nodig.
* Werkt beter dan de RandomAgent, omdat het logische keuzes maakt.

Nadelen:

* Denkt niet vooruit, kan in een valstrik lopen tegen bijvoorbeeld de MinMax.
* Herkent geen dubbele dreigingen of blokkades.

Deze vier agents zijn erg verschillend van simpel naar slimme agents. Door deze te laten spelen tegen elkaar en een baseline kunnen we bekijken welke agents het beste werken.

**Implementatiedetails**

De implementatie is opgebouwd uit de volgende stappen:

1. Agents bouwen: We hebben 4 agents gebouwd; een random agent, double move agent, min max agent en een greedy agent.

* Random agent: Voert willekeurige zetten uit zonder strategie.
* Double move agent: Probeert twee schijven in dezelfde kolom te plaatsen
* Min-max agent: Zoekt de best mogelijke zet door vooruit te denken en het spel te simuleren.
* Greedy agent: Kiest de zet die onmiddellijk het meeste voordeel oplevert, zonder verder vooruit te kijken.

1. Spelbord representatie: Er wordt een bord gemaakt waarin lege vakjes, schijven van speler 1 en schijven van speler 2 worden weergegeven.
2. Regelcontrole: Elke regel wordt in volgorde toegepast. Zodra een regel een geldige zet vindt, wordt deze uitgevoerd.
3. Prioriteit: De regels worden volgens een vaste volgorde afgehandeld. Zodra een regel met een hogere prioriteit van toepassing is, wordt deze meteen uitgevoerd en komen regels met een lagere prioriteit niet meer aan bod.

# Resultaten

A graph with a green rectangular and red rectangular and black rectangular bars

AI-generated content may be incorrect.

In de staafdiagram hierboven is te zien hoe de MinMaxAgent heeft gepresteerd tegen de RandomAgent in 20 wedstrijden. De MinMaxAgent won 15 keer, terwijl de RandomAgent slechts 1 overwinning behaalde. Daarnaast eindigden 4 wedstrijden in een gelijkspel.

Deze resultaten suggereren dat de MinMaxAgent aanzienlijk beter is in het maken van strategische zetten dan de RandomAgent, die willekeurige beslissingen neemt. Het aantal gelijke spelen toont echter aan dat de MinMaxAgent niet in alle situaties een winnende zet weet te vinden.

A graph of a graph with blue squares

AI-generated content may be incorrect.

In de staafdiagram hierboven laten we zien hoe vaak de vier agents in totaal hebben gewonnen. De GreedyAgent steekt er duidelijk bovenuit met 238 overwinningen. De RandomAgent heeft daarentegen slechts 29 keer gewonnen. Tussen de MinMaxAgent en de DoubleMoveAgent is het verschil klein: de MinMaxAgent behaalde 127 overwinningen, terwijl de DoubleMoveAgent 111 keer won.

De sterke prestaties van de GreedyAgent zijn waarschijnlijk te verklaren door zijn aanpak: deze agent kiest per beurt de zet die direct het meeste voordeel oplevert, zonder diep vooruit te kijken. Hierdoor kan hij snel inspelen op dreigingen van de tegenstander en kansen op directe winst benutten.

De RandomAgent presteert het slechtst omdat deze willekeurige zetten maakt, zonder strategisch vooruit te denken of winstkansen te herkennen. Dit verklaart waarom deze agent aanzienlijk minder wedstrijden heeft gewonnen.

# Reflectie

Uit onze resultaten kunnen wij concluderen dat de GreedyAgent het beste presteert. Omdat de RandomAgent als onze baseline fungeerde, kunnen we stellen dat de GreedyAgent aanzienlijk beter is dan deze willekeurige aanpak. Dit komt doordat de GreedyAgent per beurt de meest voordelige zet kiest, waardoor hij snel kan reageren op directe kansen en bedreigingen.

**Beperkingen van het systeem**

Hoewel de prestaties van de GreedyAgent goed zijn, hebben we enkele beperkingen van ons rule-based systeem geïdentificeerd:

1. Gebonden aan vaste regels:  
   Het model kan alleen zetten uitvoeren die voldoen aan de vooraf gedefinieerde regels. Complexe strategieën die buiten deze regels vallen, worden niet herkend of toegepast.
2. Gebrek aan leervermogen:  
   Omdat ons systeem rule-based is, leert het niet van eerdere fouten of successen. Hierdoor past het zich niet aan nieuwe strategieën of ongebruikelijke spelpatronen aan.

Door deze beperkingen kan het systeem goed presteren in standaard situaties, maar zal het moeite hebben om te winnen van adaptieve tegenstanders.

# Bronnenlijst

* *ChatGPT. (2025).\_* <https://chatgpt.com/c/67e6c0d2-b430-8013-9c07-88528297189c>
* *OpenAI. (2025, maart 28). Connect Four greedy agent explanation [ChatGPT-gesprek].\_* <https://chatgpt.com/share/67e6c187-3a50-800d-a21d-4b0a7b065553>
* *Adada. (2003, december 15). Connect Four playing AI agent.\_* <https://0xadada.pub/2003/12/15/connect-four-playing-ai-agent/>
* BotPenguin. (2024, October 26). *Rule-Based System: Pros and Cons | BotPenguin*. <https://botpenguin.com/glossary/rule-based-system>