Preskriptiv analytik Slutprojekt med Unity

av Ammar Mandawi

# **Bil som navigerar genom en 3D-miljö med Unity ML-agent implementation**

A screenshot of a computer

Description automatically generated

**Den 2023/12**

Contents

[Bil som navigerar genom en 3D-miljö med Unity ML-agent implementation 1](#_Toc152858547)

[Konfigurationsanalys: 3](#_Toc152858548)

[Kodanalys: 3](#_Toc152858549)

[Sammanfattningsvis: 3](#_Toc152858550)

[Två olika metoder Basic.yml och Walker.yml 4](#_Toc152858551)

[Kod/Script beskrivning: 5](#_Toc152858552)

[Terminalkommandon för att använda Anaconda med ML-Agents för träning: 7](#_Toc152858553)

[Länk för projekt: 8](#_Toc152858554)

[Github länk för kode alla material för kursen: 8](#_Toc152858555)

[ml-agents /config /ppo/ länk: 8](#_Toc152858556)

[Bilder om hele projekten: 8](#_Toc152858557)

A video game of a car on a road

Description automatically generated  
Projektbeskrivning:

Projektet involverar en bil som navigerar genom en 3D-miljö med målet att undvika kollision med vägen och nå slutet av den. Unity ML-agent implementeras för att lära bilen undvika kollisioner och belönas när den når målet eller träffar en cylinder. Målet är att träningen leder till att bilen navigerar säkert och effektivt.

# Konfigurationsanalys:

Jämförelse mellan walker.yml och Basic.yml visar skillnader i parametrar som påverkar träningsresultaten. walker.yml presterar snabbare och stabilare med högre batch-storlek, mindre buffer, normalisering, fler dolda enheter, högre gamma, längre tidshorisont och längre träningssteg.

# Kodanalys:

Unity ML-Agents Agent Script används för att styra bilens beteende. Initialisering och parametrar sätts upp, handlingar och observationer hanteras, och olika metoder styr rörelse och logik. Kodstrukturen möjliggör träning av en intelligent agent genom maskininlärning.

Sammanfattningsvis:

Projektet går ut på spelmekanik och maskininlärning för att skapa en autonom bil, och konfigurationsanalysen belyser hur olika inställningar påverkar träningsresultaten. Kodstrukturen ger en översikt över agentens beteende och styrning.Problem lost med två olika metoder:

# Två olika metoder Basic.yml och Walker.yml

A graph with blue and pink lines

Description automatically generated A graph with lines and numbers

Description automatically generated

A screen shot of a computer program

Description automatically generated A screen shot of a computer program

Description automatically generated

**Batch-storlek:**  
*walker.yml:* Använder en större batch-storlek på 2048. Detta betyder att fler exempel används i varje träningsuppdatering.  
*Basic.yml:* Använder en mindre batch-storlek på 512.

**Buffer-storlek:**  
*walker.yml:* Använder en mindre upplevelsebuffer på 20480. Bufferstorleken påverkar minnet för träningsupplevelser.  
*Basic.yml:* Använder en större upplevelsebuffer på 32768.

**Normalisering:**  
*walker.yml:* Normaliserar observationerna (sätter dem på en gemensam skala).  
*Basic.yml:* Använder inte normalisering av observationerna.

**Dolda enheter:**  
*walker.yml:* Har fler dolda enheter (256) i varje lager av det neurala nätverket.  
*Basic.yml:* Har färre dolda enheter (20).

**Gamma (rabattfaktor):**  
*walker.yml:* Använder en högre rabattfaktor (0.995) för den extrinsiska belönningssignalen. Detta ger högre viktning åt framtida belöningar.  
*Basic.yml:* Använder en lägre rabattfaktor (0.9).

**Maximala steg:**  
*walker.yml:* Anger ett högre maximalt antal träningssteg (30 miljoner).  
*Basic.yml:* Använder ett lägre maximalt antal träningssteg (5 miljoner).

**Tidshorisont:**  
*walker.yml:* Använder en längre tidshorisont (1000). Detta påverkar hur långt in i framtiden agenten ser när den tar beslut.  
*Basic.yml:* Använder en kortare tidshorisont (3).

Dessa konfigurationsskillnader har påverkat resultaten så att walker.yml har varit mycket snabbare och uppnått målet på ett mer stabilt sätt och kortare tid jämfört med Basic.yml.

Kod/Script beskrivning:

A screenshot of a computer program

Description automatically generated  
**Unity ML-Agents Agent Script**

**Initialization och Parametrar**

**• StartPosition: Lagrar agentens initiala position.**

**• StartingRotation: Lagrar agentens initiala rotation.**

**• startDistance: Representerar det initiala avståndet till målet.**

**• speed: Agentens hastighet.**

**• turnSpeed: Hastighet för att svänga agenten.**

**• maxSpeed: Maximal hastighet för agenten.**

**• m\_Rigidbody: Rigidbody-komponenten för agenten för fysiska interaktioner.**

**• Olika spelobjekt (GoalDistanse, Waypoint, C\_point, tresure, StartDistance) är offentliga fält för referenser i Unity-editorn.**

**Handlingar och Observationer**

**• OnActionReceived: Denna metod kallas när agenten får en handling. Den tolkar diskreta handlingar (t.ex. röra sig framåt, röra sig bakåt, svänga vänster, svänga höger) och utför motsvarande handlingar.**

**• CollectObservations: Denna metod används för att samla observationer från omgivningen för agenten. Den inkluderar information som agentens position, waypoint-position, checkpoint-position, hastighet, avstånd till waypoint och rotation.**

**Start-metod**

**• Start: Initialiserar startposition, rotation och andra parametrar. Den ställer också in de initiala positionerna för cylindrarna (punkterna) med hjälp av StartAllCylinders.**

**StartAllCylinders-metod**

**• StartAllCylinders: Positionerar cylindrarna (punkterna) på fördefinierade koordinater. Koordinaterna anges i tre arrayer (xCoordinates, yCoordinates, zCoordinates).**

**FixedUpdate-metod**

**• FixedUpdate: Denna metod kallas vid fasta intervall och hanterar agentens huvudlogik. Den beräknar avståndet till målet, belönar eller straffar agenten baserat på dess rörelse och kontrollerar träningsperioden. Den uppdaterar också bilens position och hastighet.**

**OnEpisodeBegin-metod**

**• OnEpisodeBegin: Kallas i början av en ny episod. Den återställer agentens position och rotation.**

**Reset-metod**

**• Reset: Återställer agentens position och rotation till det initiala tillståndet och stoppar dess hastighet.**

**GetGoalDistance-metod**

**• GetGoalDistance: Beräknar avståndet mellan agenten och ett angivet målobjekt.**

**ScoreCloserToGoal-metod**

**• ScoreCloserToGoal: Belönar eller bestraffar agenten baserat på dess närhet till målet.**

**TextInfoOutput-metod**

**• TextInfoOutput: Skickar textinformation till ett UI-element.**

**Heuristic-metod**

**• Heuristic: Ger ett manuellt (heuristiskt) sätt att styra agenten med hjälp av tangentbordsinmatning.**

**Rörelsemetoder**

**• LeftDrive, RightDrive, DriveForward, DriveBackwards: Metoder som styr agentens rörelse i olika riktningar.**

**RandomPosition-metod**

**• RandomPosition: Positionerar cylindrarna (waypoints) slumpmässigt.**

**otherRandomPosition-metod**

**• otherRandomPosition: Positionerar slumpmässigt en specifierad kolliderare.**

**Kollisions- och Trigger-metoder**

**• OnTriggerEnter: Hanterar händelser när agenten kommer in i en triggerkolliderare. Belönar eller bestraffar agenten baserat på kolliderarens tag.**

**• OnCollisionExit: Hanterar händelser när agenten lämnar en kollision. Den stoppar agenten och avslutar episoden om den kolliderar med en tagg "plane".**

# Terminalkommandon för att använda Anaconda med ML-Agents för träning:

# Installera ML Agents toolkit 2.3.0 i Unity

#          Window->Packe Manager

#          Tryck på + och “Install from Git URL”

#          Kopiera in följande länk i URL-rutan:

# git+https://github.com/Unity-Technologies/ml-agents.git?path=com.unity.ml-agents#release\_20

# ML Agents trainer API (installera i Anaconda)

# conda create -n mlagentsUnity python=3.8.13

# conda activate mlagentsUnity

# pip install torch==1.8.0

# pip install mlagents==0.30.0

# # Finale projekt

# cd /d D:\2024\Period 1\Autonoma Agenter\lektion 3\Exemple 4 Egen\Training\_Walker

# # RobotTestBuild18C5 sista test

# # to start conda

# conda activate mlagents2

# # visa visualisering resultat

# tensorboard --logdir results --port 5004

# # Start game in unity

# mlagents-learn config.yml --run-id="RobotTestBuild18C3" --time-scale=5  --resume

# # 10  med skärm

# mlagents-learn config.yml --run-id="MyAi-5" --env="D:\2024\Period 1\Autonoma Agenter\lektion 3\Exemple 4 Egen\Training\newBuild\Exemple 3 auto drive car.exe" --width=800 --height=512 --time-scale=10 --num-envs=1 --resume

# # 10  utan skärm

# mlagents-learn config.yml --run-id="RobotTestBuild18C3" --env="D:\2024\Period 1\Autonoma Agenter\Projekt1\ArmRobot\Training\build4X\ArmRobot.exe" --no-graphics --time-scale=5 --num-envs=5 --resume

# # build

# # 10 X 10 gånger snabbare utan screen

# mlagents-learn config.yml --run-id="MyAi-walker" --env="D:\2024\Period 1\Autonoma Agenter\lektion 3\Exemple 4 Egen\Training\_Walker\newBuild\Exemple 3 auto drive car.exe" --no-graphics --time-scale=10 --num-envs=5

# Länk för projekt:

Github länk för kode alla material för kursen:

https://github.com/Ammar-Man/Preskriptiv\_analytik\_2023\_p2/tree/main/Final\_Project

## ml-agents /config /ppo/ länk:

https://github.com/Unity-Technologies/ml-agents/tree/develop/config/ppo

# Bilder om hele projekten:

A video game of a car on a road

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated