Dizerție – Automatizarea decizilor agentilor intr-un mediu virtual

Cuprins

Sinteza

*Cap1 - Introducere*1.1 - Motivatie  
1.2 - Obiective

*Parte generală*Cap2 - Agenți software  
2.1 - Domenii de aplicabilitate  
2.2 - Soluții software

Cap3 - Învățare automată  
3.1 - Introducere în domeniu (ML clasic vs Rețele cu învățare profundă)  
3.2 - Arhitecturi specifice problemei abordate  
3.3 - Platforme sotfware

*Parte specială*

Cap4 - Proiectarea experimentului  
4.1 - Obiectivele experimentului  
4.2 - Arhitectura

* Circuitul din Unity
* Colectarea datelor din C#
* Scriptul în Python (API)
* Modelul de ML (random forrest & decission tree)
* Diagrama se face folosind limbajul mermaid
* Automatizarea începerii jocului și repornirea în momentul în care jocul este încheiat, fie că agentul a câștigat sau a pierdut

4.3 - Comparație sisteme de decizie

*5. Concluzii*  
5.1 - Contribuții   
5.2 - Dezvoltări viitoare

Capitolul 1 – Introducere

1.1 Motivație

Scopul lucrării este de a învăța un agent șă concureze într-un mediu virtual, în cazul de față agentul fiind representant de un pilot de kart iar mediul virtual este un circuit de karting. În acest experiment se vor compara rezultatele obținute de agent în urma folosirii atât a unor algoritmi supervizați (Decision Tree și Random Forrest) și nesupervizați (Reinforcement Learning).

În vederea realizării experimentului s-au folosit mai multe intrumente digitale:

* Unity Hub pentru mediul virtual
* Anaconda – pentru realizarea API-ului care comunică cu mediul din Unity și în care se realizează partea de ML
* Visual Studio Code – editor text pentru Python si Mermeid

Limbaje de programare utilizate:

* C# - în mediul unity
* Python – pentru API și Machine Learning

1.2 – Obiectivele

Un prim obiectiv este învățarea unui agent să concureze într-un mediu virtual și să își îmbunătățească rezultatele progresiv.

Având în vedere că agentul are nevoie ca date de intrare de informații precum poziția pe harta și senzori, acesta poate fi mai apoi replicat și îmbunătățit într-un automobil autonom.

Capitolul 3 - Sinteza lucrării

Această lucrare este dezvoltată în cadrul Facultății de Electronică, Telecomunicații și Tehnologi Informaționale, parte al Universității Politehnica din Timișoara. Lucrarea reprezintă munca depusă pe parcursul a doi ani de zile 2020-2022.

Pentru partea teoretică, sau parcurs mai multe articole științifice și ale resurse de documentare care ating subiecte precum Machine Learning, Reinforcement Learning și altele.

Pentru partea practică s-a realizat un experiment prin intermediul căreia se compară mai mulți algoritmi de Machine Learning, atât supervizați cât și nesupervizați.

Capitolul 2 Agenti Sotfware

Un agent software este un obiect software autonom, care are access la informațiile din mediul virtual în care acesta se găsește.[1] În cazul acestui experiment, agentul software este reprezentat de un kart care are misiunea să obțină cel mai repid timp pe traseul respectiv.

Agentul software are access la o sumedenie de informații din mediu, cum ar fi distanța până la obiectele din apropiere prin intermediul celor 5 senzori frontali, poziția pe cele trei axe: 0X, 0Y și 0Z, etc.

Capitolul 3 – Învățare automata

Machine Learning

Machine learning este un domeniu foarte de actualitate în zilele noastre, fie că vorbim de domenii precum Automotive, Cyber Security, Data Sience, detecții de imagini, etc. Din punct de vedere al definiților, am expus mai jos două dintre ele:

O definiție mai generală a machine learning este următoarea[2]:

* Machine learning-ul este domeniul de studiu care oferă abilitatea calculatoarelor să învețe fară ca acestea să fie programate în mod explicit (Arthur Samuel, 1958)

Există și o definiție orietată spre tehnică care zice că[2]:

* Unui program software îi este spus să învețe din experiența E respectând niște cerințe C și având niște unități de măsură a performanței P. Dacă performanțele respectă cerințele C, măsurate în P, atunci programul software își îmbunătățește experiența E. (Tom Mitchel, 1997)

Clasificarea sistemelor de Machine Learning

Există mai multe clasificări a sistemelor de ML (Machine Learning) avem următoarele clasificări[2]:

1. Modul de antrenare a sistemelor:

* Sisteme supravegheate
* Sisteme nesupravegheate
* Sisteme semi- supravegheate
* Reinforcement Learning

1. Modul de învățare

* Online prediction
* Batch prediction

Unsupervised ML

Metodele de învățare automată nesupravegheată sunt deosebit de utile în sarcinile de descriere, deoarece urmăresc să găsească relații într-o structură de date fără a avea un rezultat măsurat. Această categorie de învățare automata este denumită nesupravegheată deoarece îi lipsește o variabilă de răspuns care să poată supraveghea analiza.[3]

Supervised ML

Metodele de învățare automată supraveghetă se utilizează pentru a descrie sarcini de predicție, deoarece aceastea au scopul de prognoza și clasifica un anumit rezultat de interes (dacă o anumită persona este predispusă anumitor boli pe baza informațiilor medicale). Învățarea supravegheată a fost aplicată structurilor mari de date, deoarece în vederea obținerii unei acuratețe bune este nevoie să “hrănim” modelul de ML cu un set mare de date în ciclul de învățare.[3]

În această categorie de ML, datele sunt împărțite in mai multe seturi. Un prim set este cel de antrenare care sunt introduse in model. Al doilea set de date este cel de testare, prin intermediul căruia se determină performanța modelului, prin calcularea mai multor indicatori, cum ar fi accuratețea. De obicei, raportul dintre cele două seturi este de 9 la 1.

Setul de date de antrenare cu care este “hrănit” algoritmul de ML poartă numele de etichete (labels).

Reinforcement Learning

Capitolul 4 – Proiectarea experimentului

4.1 Obiectivele experimentului

4.2 Arhitectura experimentului

4.2.1 Circuitul din Unity

În vederea învățării agentului să piloteze kartul, se va folosii un mediu virtual deja existent în Unity care conține kartul și mai multe scenele. Contribuția autorului este să adapteze mediul și codul din Unity pentru a putea prealua datele necesare din acel mediu, dar și ca să poată sa controleze kart-ul programabil. De asemenea s-au mai adaugat și 5 senzori în fața kartului.

4.2.2 Colectarea datelor în C#

Din punct de vedere al colectării datelor avem mai mulți pași care au fost parcurși:

1. Adaptarea codului pentru prelucrarea datelor

Pentru învățarea agentului cum să conducă este nevoie de mai multe date de intrare și se ieșire.

Avem următoarele date de intrare (unity -> api):

* Timpul din Unity
* Poziția kart-ului pe axa X
* Poziția kart-ului pe axa Y
* Poziția kart-ului pe axa Z
* Senzorul din partea din stânga
* Senzorul din partea din stânga față
* Senzorul din partea din fată
* Senzorul din partea din dreapta față
* Senzorul din partea din dreapta
* Distanța senzorului din stânga
* Distanța senzorului din stânga față
* Distanța senzorului din față
* Distanța senzorului din dreapta față
* Distanța senzorului din dreapta
* Deplasarea în fată
* Zona

Avem următoarele date in ieșire(api -> unity):

* În față
* În stânga
* În dreapta
* În spate

1. Salvarea datelor într-un fișier csv pentru fiecare sesiune
2. Trimiterea datelor din fișierele .csv în google cloud -> query
3. Antrenarea modelului
4. Exemplificarea modelului (supervizat)

4.2.3 Scriptul în Python

Scriptul în python a fost implementat folosind pachetul software anaconda, prin intermediul căruia s-a creat un environment care va conține pachetele necesare pentru implementarea soluției.

4.2.3.1 Flask api

O primă componentă a sciptului python este reprezentată de Flask API, prin intermediul căreai s-a realizat un endpoint de tip POST, prin intermediul căreia datele de intrare sunt trimise de către componenta Unity, iar raspunsul este dat de decizia luată de machine learning.

Acest endpoint este apelat de către joc la fiecare frame, din interiorul metodei Update din Unity. Atât pentru intrare, cât și pentru ieșire, date sunt sub format de JSON.

4.2.4 Sistemele de decizie implementate

4.2.4.1 Algoritmi supervizați

O primă categorie de algoritmi pentru învățare automată folosiți în acest experiment sunt cei de tip supervizați. Aceștia se bazează pe un set de date deja colectate din joc, asupra cărora se aplică procedeul de antrenare și validare.

Ulterior, după ce modelul a fost învățat, agentul va lua comanda bazându-se pe datele folosite pentru învățare.

4.2.4.2 Algorimit nesupervizați

4.2.5 Flow-ul datelor

4.2.6 Automatizarea mediului

* 1. Comparație a sistemelor de decizie

Capitolul 5 – Concluzii

5.1 – Contribuții propria

5.2 – Dezvoltări viitoare