MAKE IT WORK

Projectbeschrijving Project 1
Haven en Transport

MAKE IT WORK

Inhoudsopgave

Inleiding

Als je aan Rotterdam denkt dan denk je onder andere aan de Rotterdamse Haven. De haven van Rotterdam is het grootste haven- en industriecomplex van Europa. Tussen 1962 en 2004 was de haven van Rotterdam zelfs de grootste haven van de wereld. Rotterdam heeft de ambitie om de 'smartest port' te zijn en hiermee neemt de stad het voortouw in de digitale transformatie van haven en logistiek (https://www.ai-cursus.nl/partners/port-of-rotterdam/). Daarnaast heeft de haven als doel om in 2030 schepen autonoom door de haven te laten varen. Het eerste project van het programma Applied AI richt zich daarom ook op het thema Haven en Transport (https://www.ibm.com/case-studies/port-of-rotterdam-authority).

Opdracht

Het autonoom laten rijden van voertuigen en varen van schepen is een belangrijk doel om het werk in de haven efficiënter te maken. Hiermee kunnen kosten bespaard worden en kan er meer werk worden verzet. Echter moet het autonoom laten rijden van voertuigen en varen van schepen wel veilig zijn. En wie is er verantwoordelijk als er iets misgaat? In dit eerste project krijg je de opdracht een voertuig autonoom te laten rijden in een digitale omgeving. Het autonome voertuig, een auto in dit geval, kan tijdens het rijden data verzamelen door middel van twee typen sensoren, een LIDAR en een SONAR. Je verzamelt de data van de auto om hiermee een neuraal netwerk te trainen zodat de auto zelfstandig en zo veilig mogelijk rond kan rijden in de simulatieomgeving.

Doel

Het doel van dit project is om praktische ervaring op te doen met het gebruik (in Python) van een neuraal netwerk voor het besturen van een realtime systeem, in dit geval een auto. Daarbij gaat het vooral om het fundamentele verschil tussen het imperatief programmeren van een door menselijk vernuft bedachte, situatie-afhankelijke, expliciete strategie enerzijds, tegenover de aspecifieke benadering waarbij een neuraal netwerk impliciet patronen leert herkennen, zonder dat er sprake is van zo'n strategie.

Een bijkomend doel van dit project is ervaring op te doen met het flexibel en ordelijk opzetten van software met behulp van Python. Probeer je programma's zo op te zetten dat voor overstap van SciKitLearn op Keras/Tensorflow slechts geringe wijzigingen nodig zijn en de overall structuur van je programma gehandhaafd blijft. Een dergelijke programma-opzet heet "stabiel" omdat het de tand des tijds kan doorstaan. Het is zogezegd duurzaam en bespaart daarmee mensuren en andere resources.

Context

Specifiek worden twee situaties onderzocht, besturing met behulp van een 2D Scanning LIDAR, waarbij het voertuig een nauwkeurig 2-dimensionaal beeld van z'n omgeving krijgt en besturing door een drietal losse SONAR modules, waarbij het voertuig slechts een grove

indicatie krijgt van de positie van obstakels.

Inzicht in de verschillen in hoekresolutie tussen LIDAR en SONAR technologie leiden bij imperatief programmeren tot verschillende expliciete besturings-algoritmen.

Bij uitvoering van dit project word je geconfronteerd met een aantal concrete punten waarin gebruik van een neuraal netwerk afwijkt van gebruik van een expliciet, imperatief geprogrammeerd algoritme.

Gegeven is een eenvoudige simulatie van een auto, die wel aan de natuurwetten voldoet: Als je te hard rijdt, vlieg je uit de bocht. De gesimuleerde auto kan worden bestuurd via een socket-verbinding. Begrip van de werking van de simulator zelf is voor uitvoering van dit project niet nodig.

Gegeven is tevens een hardcoded besturing in het bestand *hardcoded_client.p*y in de folder *control_clients*. Gebruik deze code tevens als voorbeeld voor communicatie met de gesimuleerde auto, die de rol van server vervuld. Merk op dat het dictionary *sensors* de meetgegevens van de LIDAR of SONAR bevat en dictionary *actuators* de gewenste stuurhoek en snelheid. De gegevens van de drie vaste SONAR units worden, samen met de resulterende stuurhoek, volledig gelogd. De gegevens van de LIDAR worden gereduceerd in dimensionaliteit, door de gegevens van de LIDAR niet bijvoorbeeld per graad door te geven maar de zichthoek van de LIDAR in een aantal (bijvoorbeeld zestien) sectoren te verdelen. Ook met deze opsplitsing heeft de LIDAR gemiddeld een 16/3 keer zo nauwkeurige hoek-resolutie als de SONAR.

Deliverables

- Projectplan - Zie hiervoor practicum ‘Projectplan opstellen’
- Broncode - Doe dit in kleine stappen en zorg dat je steeds iets hebt dat in ieder geval de goedkeuring van de Python interpreter kan wegdragen, al doet het nog niks. Werk niet van begin naar eind maar van belangrijk naar onbelangrijk en houd de planning in de gaten om te zorgen dat je niet te veel in de details verdwaalt (MoSCoW gecombineerd met timeboxing). Alle code moet gepusht naar jouw eigen Github project. Zorg ervoor dat de docenten toegang hebben tot deze omgeving.
- Testrapport – Het testrapport beantwoordt de volgende vragen:
    - Welke scanningsmethode (SONAR/LIDAR) is het meest geschikt en waarom?
    - Welke aansturingsmethode (explicit algoritme /neuraal netwerk) werkt

het beste en waarom?

- Hoe is veiligheid gegarandeerd in alle gevallen?
- Welke aansturingsmethode is het veiligst en waarom?
- Is de opzet van de codebase modulair? Hoe aan te tonen?

- Projectevaluatie – Het doel van de projectevaluatie is tweeledig. Aan de ene kant wordt in dit document aandacht besteed aan de meta-functionele overwegingen die gemaakt worden tijdens het project, daarnaast wordt in dit document aandacht besteed aan de individuele ervaringen gedurende dit project. Meta functionele eisen: het gaat hier om bedrijfsmatige, ethisch-maatschappelijke, juridische, regulerings- en technische kaders die de directe functionele eisen overstijgen. Wat is het verschil tussen een neuraal netwerk en een expliciet algoritme? Dit betreft onder andere de mate waarin:
  - Detailkennis over de technologie van het te besturen systeem nodig is om de besturing te maken
  - Het gedrag van de besturing in specifieke situaties te verklaren is
  - Het gedrag van de besturing in nieuwe situaties te voorspellen is
  - Een mens ethisch gezien verantwoordelijk is, dan wel juridisch verantwoordelijk kan worden gesteld voor ongelukken

  De projectevalutatie is een groeidocument, met andere woorden, hij wordt gaandeweg het project verder uitgewerkt. Vanwege het informele karakter van dit document hoeft hierop geen versiebeheer te worden toegepast.

Tips & Tricks

1. Vergelijk de LIDAR-besturingsstrategie in method *lidarSweep* met de SONAR-besturingsstrategie in functie *sonarSweep*. Probeer in samenspraak met andere cursisten en, indien nodig, de docent, te komen tot een formulering in “gewoon nederlands” van beide strategieen en de verschillen ertussen.
2. Laat het voertuig daarna achtereenvolgens met beide strategieen rijden op de bijpassende baan (*lidar.track* of *sonar.track*) en de bovengenoemde gegevens loggen.
3. Maak vervolgens in Python een fully connected neuraal netwerk met tussen de 2 en de 5 lagen en per laag tussen de 8 en de 256 nodes met behulp van SciKitLearn. Gebruik een *relu* activation function. Er is één analoge uitgangsnode. Gebruik de eerder gegenereerde logfiles als trainingsdata. Houd rekening met het volgende:

- Omdat alleen op de trainingsbaan hoeft te worden gereden, is de trainingset gelijk aan de testset. Er hoeft dus, net als bij de Formule 1, geen rekening gehouden te worden met overfitting, het gaat alleen om de prestaties op *dit* "circuit".
- Begrens als eerste stap alle afstanden op 20 meter, om ill conditioning te voorkomen.
- Pas daarna voorafgaand aan de training per kolom uniforme schaling toe naar het bereik [-1, 1]
- Vergeet niet weer terug te schalen bij het gebruik van het getrainde netwerk om de auto te besturen!

4. Probeer verschillende laagbreedtes en aantallen lagen uit. Je mag ook nog andere zaken proberen, zoals een andere activation function. Kijk wat het beste werkt.
5. Maak, als je een optimum hebt bereikt voor LIDAR en SONAR, precies hetzelfde netwerk met TensorFlow/Keras. Wat zijn de verschillen?
6. Wat heeft meer invloed op het resultaat, de verschillen tussen enerzijds SciKitLearn en anderzijds Tensorflow/Keras of de verschillen in gekozen netwerk-topologie, ongeacht welke library je gebruikt?