Adviesrapport voor een verkeerssysteem in Utrecht door middel van Reinforcement Learning.

Inhoud

Context

Onder de bestuurs doelen van de gemeente vallen doelen als het reduceren van de uitstoot van broeikasgassen en het verhogen van de bereikbaarheid van binnen de stad. Binnen de gemeente kwam de vraag naar boven of slimme verkeerslichten die gebruikmaken van Reinforcement learning (Hierna vernoemt als RL) een oplossing kan bieden. Binnen dit onderzoek zullen wij de voordelen, nadelen, ethische overwegingen en praktische obstakels toelichten. Wij maken gebruik van persoonlijke ervaring opgedaan binnen het project Adaptive systemes (Hierna vernoemt als het project) en andere beschikbaren bronnen.

Management samenvatting

Door de hoge hoeveelheid parameters die komen kijken bij een verkeerssimulatie heb je een hoop states/situaties. De hoeveelheid is dusdanig hoog dat het maar de vraag is of een netwerk hier ooit op zal trainen (zie Nadelen:Veel input mogelijk en Nadelen:Train tijd). Ook zijn er voor het bepalen van de reward dusdanig veel ethische en praktische obstakels dat de kans realistisch is dat de RL ongewenst gedrag vertoond, ook al is het optimaal getraind. (Zie Nadelen: Reward functie en Ethische Overwegingen: Reward functie).
Ervan uitgaande dat al deze obstakels worden overwonnen moet er rekening mee worden gehouden dat de environment continue veranderd (zie Nadelen: Dynamisch omgeving). Aanpassingen aan de infrastructuur of omgeving kunnen ervoor zorgen dat het systeem niet langer optimaal werkt. Het zal opnieuw getraind moeten worden maar in de tussentijd werken de stoplichten suboptimaal. Potentieel zelfs slechter dan de huidige oplossing.
Om deze redenen en de hoge tijd en geld investering die een RL oplossing vereist raden wij het af om een RL oplossing te bouwen. Wel is het interessant om een RL oplossing te gebruiken om knelpunten in de stad te herkennen en potentieel op te lossen. (zie Extra)

Bestaande oplossingen

In de bestaande verkeerslichten springen de stoplichten op rood wanneer ze zien dat er een gat valt nadat een reeks auto's het kruispunt heeft gepasseerd. De informatie krijgen ze van meetlussen in het wegdek. Wanneer de auto's gepasseerd zijn gaat een ander verkeerslicht op groen. De volgorde waarin dit gebeurt, is van tevoren ingesteld. Hierdoor kan er wanneer er een omleiding is snel een file voorkomen.

Voordelen

Trainen op vele input states.

Een voordeel van RL is dat het vele verschillende states als input kan krijgen. Hierdoor kan het trainen op alle verschillende scenario's. Omdat in het verkeer er oneindig veel verschillende mogelijkheden zijn, is het slim om dit in buckets te stoppen. Dit betekent dat je de states opslaat als kleine groepen en dan kijk je in welke bucket de huidige state past. Hierdoor kan je van een oneindig universum een kleiner discreet universum maken.

In de verkeerssituatie zou je bijvoorbeeld de positie van de auto's in buckets opschrijven. Het maakt niet veel uit of een auto 50 cm van het midden van de weg af staat of niet. Ook zou je de type van de autos in buckets kunnen plaatsen. Bijvoorbeeld in kleine, grote en vrachtwagens. Hierdoor kan je generaliseren waardoor de input states worden verkleind.

Gedrag hoeft niet handmatig te worden uitgeschreven

Als je niet gebruik wilt maken van een RL dan moet je voor elk scenario het gedrag van het algoritme uitschrijven. Omdat er duizenden verschillende scenario's zijn, is dit niet te doen. Door gebruik te maken van een RL kunnen wij dit complexe gedrag van elk scenario versimpelen. Door te trainen kan het zelf ontdekken wat goed en verkeerd is. Dit werkt zo in zijn algemeenheid maar is uiteraard zeer relevant voor deze casus omdat er een gigantische hoeveelheid situaties mogelijk zijn in deze casus (zie ook Nadelen veel input mogelijk).

Innovatief idee

RL heeft een imago dat het innovatief en modern is. Door het toepassen van RL kan het imago dat Utrecht een stad worden waar op een innovatieve manier met moderne technieken problemen worden opgelost en verbeterd. De verbeteringen van het imago maken uiteraard geen deel uit van de primaire doelen van deze casus maar zijn een mogelijke positieve bijwerking. Wel kan het de steun van de bevolking voor dit project verhogen.

Nadelen

Veel input mogelijk

Omdat er zoveel input variabelen mogelijk zijn (de locatie van auto's, de snelheid, waar ze vandaan kwamen, etc) wordt het snel moeilijk om overzicht te houden. Hoe meer parameters hoe meer situaties/state hoe hoger het aantal situaties hoe moeilijker het is om te trainen. Een RL moet een situatie namelijk al eens zijn tegengekomen voordat het weet wat het moet doen.

Blackbox

Een RL blijft tot in zekere maten een blackbox. Het is moeilijk om te zien wat het systeem ziet en hoe hij reageert. Binnen het project hebben wij de input image gecropt om zo de states te verminderen. Na het trainen kwamen wij erachter dat de agent niet verplaatste. Om er achter te komen waarom dit het geval was hebben wij uiteindelijk een gif gemaakt met die input van de Agent. Hier bleek dat de agent zichzelf niet zag. Waarschijnlijk verplaatste hij zich hierdoor niet. In tweede instantie bleek ook dat hij de kogels niet zag. Deze kleine dingen tonen zich alleen na een grote tijd investering in het weergeven van de input van de Agent. Tijdens het trainen is niet te zien of er zulke fouten zitten in het systeem of dat het de train parameters zijn die niet kloppen.

Ook is er voor gezorgd dat de Agent meerdere frames kan waarnemen. Voor zover wij weten hebben wij dit juist geïmplementeerd. Als wij fouten hadden gemaakt zouden wij input en demension errors moeten krijgen. Het gebrek hieraan liet ons concluderen dat de implementatie goed was maar 100% zeker kan je nooit zijn. Debuggen wordt zo erg moeilijk.

Train tijd

Naast dat het systeem tot in zeker mate een black box is duurt het trainen van een netwerk erg lang. Hoelang zal de praktijk uitwijzen maar de train tijd kan zitten tussen een paar uur, een paar dagen of tot weken.
Ongeacht de specifieke train tijden kamp je met hetzelfde probleem. Namelijk dat het maken van een aanpassing en het noteren van het effect erg lang duurt. Minuscule aanpassingen kunnen dagen duren. Verschillende aanpassingen kunnen niet met elkaar gecombineerd worden omdat het dan niet meer duidelijk is welke aanpassing verantwoordelijk is voor welk effect. Zelfs na oneindig lang trainen met de optimalen parameters is het nog steeds niet gegarandeerd dat er een gewenst resultaat uitkomt.

Lastig ontwikkelproces

Hierdoor is het ontwikkelproces ook moeilijk te plannen. Het is erg moeilijk om in te schatten hoeveel progressie er is gemaakt na een bepaalde hoeveelheid tijd. Het is mogelijk dat er weken worden gespendeerd aan aanpassingen die geen effect blijken te hebben. Het kan ook dat de optimalen parameters relatief snel worden gevonden. Dit zagen wij ook terug in het project. Wij dachten dat we na een paar dagen parameters zouden hebben gevonden die een kleine verbetering lieten zien maar dit was niet het geval. Zelfs nu is het onmogelijk te zeggen of als we door hadden gewerkt wij deze wel hadden gevonden.

Reward functie

Het is zeer ingewikkeld om te bepalen hoe het systeem moet worden beloont voor het uitvoeren van acties.
Binnen het project is er geprobeerd om een reward mee te geven voor hoe lang de agent overleefde. Uiteindelijk is de wortel van de timestep opgeteld bij de standaard reward (of hij een vijand heeft gedood). Na 1.5 uur trainen bleek dit ook weinig effect te hebben. Wij hebben achteraf niet kunnen bepalen of dit kwam omdat de toevoeging aan de reward functie geen nut heeft of dat dit komt omdat de train parameters niet exact goed zijn. Verder komen er nog ethische overwegingen bij kijken (zie Ethische Overwegingen).

Dynamische omgeving

Utrecht is een dynamische omgeving, dit betekent dat het constant veranderd. Denk bijvoorbeeld aan nieuwe gebouwen of wegen die worden aangepast. Dit beïnvloedt de drukte in het verkeer. Verder is de Aarde zelf ook dynamisch, het weer, de seizoenen en de tijd hebben ook allemaal een grote invloed op de drukte.

Als de omgeving genoeg verandert dan kan het model snel fout getraind zijn. Hierdoor kan het model de eerste paar maanden goed werken, maar na een verbouwing verminderde resultaten laten zien. Het systeem zou dan opnieuw moet worden getraind maar het is maar de vraag of hier de tijd en ruimte voor is. Als dit bij elke verbouwing of infrastructuur aanpassing moet uiteraard niet.

Ethische Overwegingen

Reward functie

De grootste ethische overwegingen komen bij het bepalen van de reward functie. Voor de reward functie zal moeten worden bepaald hoe hoog de beloning is voor het verhogen van de doorstroom voor elk vervoerstypen.

- Moet er onderscheid worden gemaakt tussen verschillende auto's? De ene vervuilt meer dan de ander.
- Tellen fietsers zwaarder mee omdat we fietsen willen promoten of minder zwaar omdat ze minder uitstoten?
- Krijgt een bus altijd voorrang omdat zo meer mensen sneller op hun locatie aankomen?

Dit zijn maar een paar van de overwegingen die moeten worden genomen voor de beloning. Deze voorbeelden komen allemaal uit hetzelfde probleem namelijk. Wanneer wegen de belangen van de samenleving niet meer op tegen het leed van de individuen en in hoeverre zijn wij bereid goed gedrag te stimuleren als de stimulans een nadelig effect heeft.

Op deze twee vragen moet een antwoord worden gevonden voor het bepalen van de beloning.

In het geval van ongelukken

Bij het huidige systeem wordt er geen rekening gehouden met overtreders. Iemand die door rood rijd plaatst zichzelf en andere in een verhoogd risico en er is geen manier voor de huidige oplossing om hier rekening mee te houden. In een RL model kan hier potentieel wel rekening mee worden gehouden. Overtreders kunnen worden gesimuleerd en als het model hier niet op aanpast krijgt het een straf. Door hier rekening mee te houden wordt het gedrag van verkeersdeelnemers uiteraard onvoorspelbaarder hierdoor is het mogelijk dat het systeem niet meer trained. Als dit wordt genegeerd en er komt een ongeluk is er dan sprake van (ethische) (mede) verantwoordelijkheid? Buiten overtreders om moeten risicovolle situaties worden voorkomen. Dit zal moeten worden gedaan door een straf in de reward. Een te hoge straf leidt tot een te voorzichtig systeem. Dit systeem zal dan geen oplossing vormen die beter werkt dan de huidige oplossing. Een te lage straf tot meer risico en mogelijk ongelukken. Als er een te lage straf wordt gekozen en er gebeurt een ongeluk hangt de vraag van aanraakbaarheid in de lucht. Door alleen al deze alinea te schrijven kan worden beargumenteerd dat management en de ontwikkelaars zich bewust waren van de risico's, niet de nodige maatregelen hebben genomen om deze risico's uit te sluiten en dus verantwoordelijk zijn voor de gevolgen.
Hoewel er handmatig regels kunnen worden geschreven als "Niet twee stoplichten tegelijk op groen" zullen nooit alle risicosituaties kunnen worden afgevangen.

Praktische Overwegingen

Simpel of een complex systeem

Een praktisch probleem is de grote van het systeem. Hoe groot maak en hoeveel inputs geef je het systeem. De omgeving van een kruispunt heeft effect op het kruispunt zelf. Hierdoor kan je meerdere kruispunten meenemen om één kruispunt te voorspellen. Hierom gaan we kijken naar twee verschillende opties. De eerste is een simpel systeem met één kruispunt, en de tweede is een model dat gebruik maakt van de gehele stad.

Simpel [1 kruispunt]

Wanneer we gaan kijken naar een enkel kruispunt dan hebben we het probleem dat we niet weten wat er in de omgeving gebeurd. Ook weten we niet of we een probleem opschuiven naar een volgende kruispunt. Verder is het ook lastig om te bepalen wie voorrang krijgt bij een kruispunt. Zijn groene auto's, vrachtwagens, lang wachtende auto's , groene stromen of fietsers belangrijk op een kruispunt. Hiervoor moet je een slimme reward functie voor bedenken.

Maar als je dan een model gaat trainen is de kans groot dat een model te specifiek op een reward gaat trainen. Hierdoor is het risico dat mensen lang wachten of dat een groep te vaak de voorkeur krijgt. Dit is ethisch gezien niet eerlijk.

Complex [hele stad]

Als we de gehele stad nemen als input voor het model, dan hebben we het probleem dat het model heel snel erg complex is. Elk kruispunt wordt meegenomen in een enkele berekening. Het probleem hiermee is dat wanneer één enkele sensor stopt met werken dat het gehele systeem eruit is. Wanneer de stad veranderd, zoals een verbouwing, dan moet het hele systeem opnieuw getraind worden.

Verder moeten er op gelet worden dat de gehele stad eerlijk geregeld wordt, het mag niet zo zijn dat een straat altijd de voorkeur krijgt over een andere straat. Als je er achterkomt dat er een fout in het systeem zit, dan moet ook het hele model opnieuw getraind worden.

Training Data

Het is ook erg lastig om trainingsdata te krijgen voor een model. Er zijn twee manieren om data te kunnen krijgen. De eerste manier is om data te verzamelen van de verkeers lichtkasten, dit geeft niet veel informatie maar genoeg om een model te trainen. Hierdoor heb je weinig inputs en weet het model niets af over de omgeving.

De tweede manier is om een gehele simulatie te bouwen om te trainen, het risico hiermee is dat je snel een bias krijg afhankelijk van hoe de simulatie werkt. Verder kan je niet een simulatie zomaar omzetten naar een echte wereld scenario. Een kruispunt in de simulatie

zal nooit op precies dezelfde manier werken als een kruispunt in de echte wereld. Hierdoor kan je een model perfect trainen in een simulatie, maar kan het in de echte wereld niet werken.

Planning

1. Data krijgen
    1.1. Simulatie + onderzoek
        1.1.1. Arbeid: Groot
        1.1.2. Risico: Gemiddeld
        1.1.3. Tijd: Veel
        1.1.4. Uitleg: Er moet een volledig onderzoek worden uitgevoerd voor een goede simulatie.
    1.2. Verkeers kasten
        1.2.1. Arbeid: Weinig
        1.2.2. Risico: Hoog
        1.2.3. Tijd: Weinig
        1.2.4. Uitleg: De data bestaat al maar is erg gelimiteerd,

2. RL model bouwen
    2.1. Arbeid: Gemiddeld
    2.2. Risico: Gemiddeld
    2.3. Tijd: Gemiddeld
    2.4. Uitleg: Dit is puur een prototype bouwen van een reinforcement model.

3. Optimalisatie
    3.1. Double Q
        3.1.1. Arbeid: Weinig
        3.1.2. Risico: Hoog
        3.1.3. Tijd: Gemiddeld
        3.1.4. Uitleg: We zagen geen verschil tussen wel en geen DQN, dit komt omdat het trainen extreem lang duurde.

    3.2. Memory meegeven van 4 vorige frames zodat snelheid meegenomen kan worden
        3.2.1. Arbeid: Hoog
        3.2.2. Risico: Gemiddeld
        3.2.3. Tijd: Hoog
        3.2.4. Uitleg: Dit zorgde ervoor dat het model het een klein beetje beter werkte, maar het kost veel tijd om te implementeren. Ook is het heel lastig om het goed te implementeren.

    3.3. Parameters optimaliseren
        3.3.1. Arbeid: Weinig
        3.3.2. Risico: Weinig
        3.3.3. Tijd: Veel

3.3.4. Uitleg: Door met de parameters rond te testen kan je ongeveer voelen wat goed en niet goed is. De parameters aanpassen is redelijk simpel maar kost veel tijd om te zien of het effect heeft.

3.4. Reward functie
   3.4.1. Arbeid: Weinig
   3.4.2. Risico: Extreem Hoog
   3.4.3. Tijd: Gemiddeld.
   3.4.4. Uitleg: Door de reward functie aan te passen is het mogelijk dat het model sneller traint. Hierdoor is de kans dat het model een bias krijgt extreem groot.

4. Bepalen juridische verantwoordelijkheden en aansprakelijkheid
   4.1. Arbeid: Laag
   4.2. Risico: Gemiddeld
   4.3. Tijd: Laag
   4.4. Uitleg: Het is essentieel dan wordt onderzocht in hoeverre er sprake is van juridische aansprakelijkheid bij ongevallen en wat het potentieel risico is van schade claims.

5. RL model toepassen in de echte wereld
   5.1. Arbeid: Gemiddeld
   5.2. Risico: Gemiddeld
   5.3. Tijd: Hoog
   5.4. Uitleg: Het model gebruiken in de werkelijke wereld. Het is slim om het huidige systeem langzaam te vervangen in bijvoorbeeld nachturen door het systeem.

Conclusie

In het verkeer zijn er te veel variabelen die een situatie kunnen beïnvloeden. Dit zorgt ervoor dat het model onvoorspelbaar kan worden wanneer er iets klein veranderd. Een DQN in het verkeer toepassen is eigenlijk een toekomstscenario, omdat het rijgedrag van mensen onvoorspelbaar is en omdat de stad constant verandert is het nu praktisch niet slim om toe te passen. DQN zou in de toekomst wanneer alle auto’s met een server verbonden zijn beter werken dan nu.
DQN wordt dus niet aangeraden vanwege een paar redenen. Als eerste zijn de kosten te hoog om een DQN toe te passen in een verkeerssituatie. Daarentegen wekt het huidige model van de verkeerslichten goed genoeg, de kans dat een DQN dit significant verbeterd is klein.

Extra

Voor de huidige casus is een RL oplossing dus niet aan te raden maar misschien is er met een lichte aanpassing aan de casus wel een interessante oplossing. Door de huidige situatie

te simuleren en een concrete reward opstellen (zie Ethische overwegingen en Nadelen:Reward voor obstakels hierin) kan er binnen de simulatie worden gezocht naar knelpunten. De simulatie zorgt er voor dat elke afslag/kruispunt/stoplicht als een punt kan worden gezien waar n hoeveelheid verkeersdeelnemers aanwezig zijn. De verkeersdeelnemers kunnen worden gestructureerd naar een verkeerd soort. (Snelheid verkeer is snelheid auto * kans op auto + snelheid vrachtwagen * kans op vrachtwagen, etc). Door deze abstracte vorm van verkeer te nemen kan je de states gigantisch reduceren.

(Dit is niet mogelijk in de huidige casus omdat het verkeer daar concreet moet blijven. Als je een abstract gemiddelde neemt voor verkeer dan zul je krijgen dat het niet werkt als er auto's voor het stoplicht staan en niet als er vrachtwagens staan omdat het op beide net niet is getraind.)

In deze simulatie kunnen dan knelpunten worden herkent. Aanpassingen kunnen dan worden gemaakt en nieuwe knelpunten kunnen worden bepaald door het model. Op deze manier kan worden onderzocht welke aanpassingen er moeten komen in de infrastructuur om de doorstroom te verhogen.