1. Probleemstelling

We hebben een fiets met een automatische transmissie. Hiervoor gebruikt de fiets een CVT (Continuously variable transmission). Er zijn 2 soorten CVT’s, een mechanische en een elektrische. De mechanische CVT gebruikt in plaats van tandwielen een kegelvormig object. Men schakelt dan door de band over dit object te bewegen. Hierdoor is een continue overgang mogelijk. In dit geval gebruikt de fiets een elektrische CVT die een mechanische CVT imiteert. Hiervoor gebruikt de fiets 2 motoren. 1 die de cadans die de fietser voelt aanpast, en 1 die de kracht die de fietser moet leveren aanpast. Dit zorgt ervoor dat de fiets natuurlijk aanvoelt.

Momenteel kan de fietser de snelheid van de trappers en het ondersteuningsniveau manueel aanpassen met een smartphone. Niet de beste user experience.

Dus het doel van deze thesis is het voorspellen van deze optimale cadans.

Misschien is het best om hier eerst uit te leggen wat juist het verschil is tussen cadans, FCC en optimale cadans. De cadans van de fietser kan op elk moment gemeten worden door de fiets. Dit is de snelheid dat de fietser op elk moment aan het trappen is. Meestal is dit een waarde tussen 0 en 120 rpm. FCC of Freely Chosen Cadence is een hypothese die gebruikt wordt om de gewenste cadans van de fietser voor te stellen. Hierover later meer. De optimale cadans is een input voor de fiets dat bepaalt hoe snel de trappers van de fiets zouden moeten draaien. Dit is hetgeen wat we gaan voorspellen.

1. Benodigdheden

De fiets genereert vrij veel data zoals de snelheid, hoek van de trapas, vermogen dat de fietser levert en meer. Al deze data wordt gelogd in een csv file aan 10Hz, maar de data wordt wel gegenereerd aan 1000Hz. Nu is het probleem dat er niet veel van deze log files beschikbaar zijn. Vanaf begin volgend jaar beginnen de user tests en vanaf dan is er dus veel data beschikbaar. Maar om al van start te geraken hebben we besloten om een fietssimulatie te maken.

**De Fietssimulatie**

De fietssimulatie genereert data aan 10Hz. De berekening die hiervoor nodig zijn werden geleverd door Jorrit.

De simulatie is zo geprogrammeerd dat de kracht dat de fietser uitoefent op de pedalen, dynamisch veranderd, naar gelang het verschil van de huidige snelheid, en een ingestelde optimale snelheid. Momenteel ligt deze optimale snelheid op 32km/h. En dus hoe dichter bij deze snelheid, hoe minder kracht dat de fietser levert.

We nemen ook aan dat de kracht die de fietser levert een cyclisch verloop toont. Dit kunnen we op de grafiek zien. Per omwenteling van de trapas zijn er 2 pieken. Elke piek staat voor 1 voet dat de trappers naar beneden duwt.

Er wordt ook terrein gegenereerd. Dit is in de vorm van een helling waarop de fietser zich bevindt. Het terrein wordt gegenereerd met behulp van perlin noise.

**Last Model**

We hebben natuurlijk ook een last model nodig. Anders zou de fietser bijvoorbeeld maar 10 keer moeten trappen voor 32km/h te behalen, en daarna nooit meer.

Er spelen 3 verschillende krachten op de fiets. Gravitationele krachten die voornamelijk worden beïnvloed door de helling waarop de fietser zich bevindt. Een kracht voor de wrijving met de weg en een kracht voor wrijving met de lucht. Samen vormen deze de totale last die op de fietser gezet wordt. Merk op dat er geen mee/tegen wind is. Er zijn hier 2 verschillende redenen voor. 1 het maakt het last model ingewikkelder.

2 Het heeft te maken met de hypothese dat we veronderstellen waar te zijn. Dit heeft te maken met de FCC. We verwachten dat de fietser zijn cadans verhoogt naar gelang de hoeveelheid last hij krijgt te verduren. Hoe meer last => hoe een hogere cadans. Met een minimum van 40 en maximum van 120 rpm.

1. Preprocessing

Ten eerste gaan we de data in sequenties onderverdelen. Dit is de input van de het machine learning model. Deze sequenties bevat data van de laatste 5 seconden. Hoe minder data we in de sequenties steken, hoe slechter de modellen presteren. Maar hoe sneller ze zijn. Momenteel zijn de input: de kracht die de fietser op de pedalen uitoefent, en de hoek van de trapas.

Data normaliseren hangt van het model af en is dus niet altijd nodig.

De hoek van de trapas moet ook gepreprocessed worden. Door de sinus en de cosinus van de hoek te namen zorgen we er namelijk voor dat data in het begin van de cyclus dicht bij data ligt op het einde van de cyclus. Zoals te zien op de 2 grafieken liggen het eerste en laatste punt ver van elkaar verwijderd als we gewoon de hoek gebruiken, maar als we de sin/cos hiervan nemen, is de afstand tussen deze punten veel kleiner.

Er kan ook noise optreden. Dit is voornamelijk noise van het wegdek. Dit kunnen we eruithalen door een FFT op de data toe te passen. Noise van het wegdek treed op bij een frequentie van ongeveer 20Hz. Noise van de motor ongeveer vanaf 13000Hz. Dit gaat echter minimalistisch zijn. We hebbe de mogelijkheid om tot 1000Hz aan data te analyseren en er noise uit weg te halen. Of dit echt nodig is, zal later bepaald worden. Op de grafiek zie je zo een FFT op echte data. Dit is 30 seconde aan data. En hier zie je dat er een piek is rond de 2Hz. Dit zijn de 2 pieken van elke voet.

1. Modellen

We hebben 4 verschillende modellen uitgeprobeerd. Een LSTM, Decision tree en random forest. En het PA algorithm. De test zijn uitgevoerd door te starten van een ongetrained model. De reden hiervoor is dat we willen weten hoe snel het algoritme eigelijk traint. Hoe sneller dat we een correct voorspelling maken, hoe beter.

Het eerste model is een LSTM netwerk. Zoals u kan zien op de grafiek ziet de voorspelling er niet al te goed uit. De grafiek toont de voorspellingen na 1 epoch. Dus is het normaal dat dit niet zo denderend is, maar als we de data na 10 epochs bekijken is het niet zo veel beter. Het duurt veel te lang om een LSTM te trainen. We hebbe veel data nodig, wat niet mogelijk is, aangezien bij het bijtrainen enkel relevante data nodig is. De laatste paar seconden dus.

De volgende 2 moddelen zijn rule based. De grafieken die u ziet tonen al een beter verloop. De decision tree vertoont een slechte consistentie, maar volgt wel meteen het juiste verloop. Een random forest, wat een aantal DT is, maakt zoals verwacht een betere voorspelling. De snelheid daalt wel, aangezien we meer bomen moeten trainen en gaan raadplegen voor een voorspelling.

Het laatste algoritme is het PA algoritme.

1. Postprocessing
2. Wat volgt?