KU LEUVEN

FACULTEIT COMPUTERWETENSCHAPPEN

Real-time cadansaanpassing in een automatische fietstransmissie

Arno Cools

Promotor:
Prof. M.-F. Moens

Begeleiders:
Ir. T. Keppens
Ir. J. Heidbuchel
Ir. R. Heidbuchel

Proefschrift ingediend tot het
behalen van de graad van
Master in de Toegepase Informatica

Academiejaar 2018-2019

Voorwoord

Ik zou graag iedereen bedanken die mij heeft geholpen met deze thesis. De personen waar ik het meeste aan te danken heb, zijn Jorrit en Rugen Heidbuchel. Beiden gaven zeer nuttige feedback en waren altijd beschikbaar. Dank aan Tomas Keppens om enkele jaren geleden het IntuEdrive project op te starten en samen te werken met de KU Leuven. Bedankt Sien Moens om de thesis te proeflezen. En natuurlijk ook een dank u aan iedereen die deze thesis gelezen heeft.

Abstract

Deze thesis gaat over het real time aanpassen van de fietscadans, toegepast op de E-bike van IntuEdrive. Deze tekst is de laatste van een reeks thesissen rond het E-bike concept van Tomas Keppens: een snelle en betrouwbare E-bike die makkelijk in gebruik is. Deze thesis werkt verder op het huidige prototype. Het doel is om de trapsnelheid van de fietser te kunnen voorspellen om zo de versnelling van de fiets automatisch te schakelen. De trapsnelheid is per gebruiker verschillend. Als eerste stap wordt een geparametriseerde simulatie gemaakt, zodat makkelijk veel consistente data gegenereerd kan worden. Vervolgens worden verschillende algoritmes van machinaal leren, het *Passive Aggressive algoritme*, binaire beslissingsbomen en *random forest*, bekeken en geëvalueerd. Er moet rekening gehouden worden met het aantal keer trainen, uitvoeringstijd en accuraatheid. Vervolgens wordt er nagegaan of de algoritmes om kunnen gaan met een stochastische keuze om het model al dan niet te updaten. De mens is immers een onvoorspelbaar wezen. Ten slotte wordt het probleem van conceptuele drift aangehaald. Een fiets kan immers door meerdere personen gebruikt worden. De resultaten tonen aan dat een random forest het best presteert in de verschillende experimenten.

Inhoudstafel

Lijst van figuren

Lijst van tabellen

Lijst van afkortingen

CVT	Continu Variabele Transmissie. 3
DC	Direct Current. 7
DEA	Droplet Ensemble Algorithm. 6
DT	Decision Tree. 18
ES	Exponential Smoothing . 19
FCC	Freely Chosen Cadence. 3
MA	Moving Average . 19
MSE	Mean Squared Error. 27
PA	Passive Aggressive. 17
RF	Random Forest. 18

Lijst van symbolen

A_{aero}	Frontaal oppervlak fietser. 13
FCC_{est}	Schatting van FCC geleverd door de cadanscontroller. 4
FCC_{pred}	Voorspelde FCC. 19
F_{aero}	Luchtweerstand. 13
$F_{friction}$	Wrijvings last. 13
F_{grav}	Gravitationele last. 13
F_{load}	Totale last. 13
K	Agressiviteits parameter voor proportionele regelaar. 8
P	Vermogen. 13
S	Ondersteuningsniveau. 11
T_{MG2}	Koppel geleverd door motor op het voorwiel. 11
$T_{cy,m}$	Gemeten koppel geleverd door de fietser. 4
T_{cy}	Koppel geleverd door de fietser. 3
$T_{dc,max}$	Maximum DC-koppel dat geleverd kan worden. 8
T_{dc}	Het DC-koppel van de fietser (gemiddeld koppel). 7
T_{rw}	Koppel op het achterwiel. 10
Δ_c	Absoluut verschil tussen fietsersmodel en voorspelling. 20
α	Helling. 4
λ	Applicatie specifieke parameter van biased reservoir sampling. 25
ω_{cr}	Cadans in rpm. 8
ρ_{aero}	Luchtdichtheid. 13
θ_{cr}	Hoek van de trapas. 4
c_d	Luchtweerstand coëfficiënt. 13
c_r	Rolweerstand coëfficiënt. 13
f_h	Factor voor fietsersmodel in functie van de helling. 11
f_k	Factor voor fietsersmodel in functie van het gemiddeld koppel. 11
f_v	Factor voor fietsersmodel in functie van de snelheid. 11
g	Gravitationele constante. 13
$k_{cr,r}$	Verhouding overbrenging trapas-ringwiel. 10
m	Totaal gewicht van fiets en fietser. 13
nr	Aantal tanden op het ringwiel. 10
ns	Aantal tanden op het zonnewiel. 10
r_w	Straal van het voor- en achterwiel. 15
r	Input vector fietser. 3
sf	Smoothing factor. 19

u_c	Toestands variabele die bepaalt of de cadans moet veranderen. 3
u_{contr}	Input vector fietser geleverd door de controller. 4
u_{cy}	Input vector fiets geleverd door de fietser. 3
v_{bike}	Snelheid van de fiets. 4
v_{ref}	Referentie snelheid van de fietser. 3
y	Output vector fiets. 4

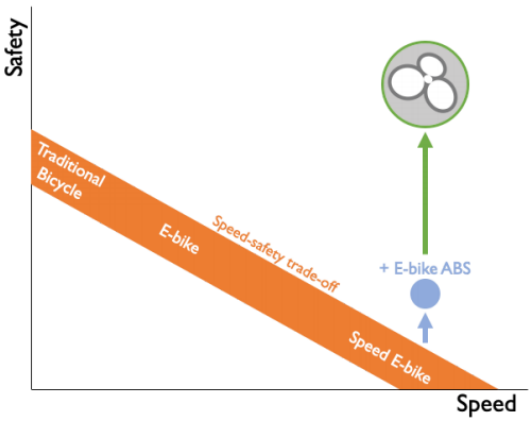
Hoofdstuk 1

Probleemstelling

1.1 Mobiliteitsvraagstuk

De auto is het slachtoffer geworden van zijn eigen succes: we staan meer dan ooit in de file en de CO_2-uitstoot van personenverkeer stijgt jaar na jaar. De Belg neemt al snel de auto voor korte afstanden (< 25 km). In deze auto zit meestal maar één persoon. Het Belgische wagenpark blijft groeien (figuur 2). Hier zien we wel een trend ontstaan. Er worden steeds meer elektrische en hybride wagens verkocht, maar die staan natuurlijk net zo goed in de file. Mobiliteit op twee wielen kan hier een oplossing bieden.

Mobiliteit op twee wielen kennen we al lang: fietsen bestaan al sinds de 19de eeuw. Elektrische fietsen hebben het potentieel van deze tweewielers enorm verhoogd: fietsen wordt moeiteloos en stukken sneller. Spijtig genoeg neemt het risico op ongevallen ook toe bij hogere snelheid. Dat komt omdat e-bikes en speed e-bikes precies dezelfde technologie gebruiken als normale fietsen – grote wielen met smalle banden, kettingaandrijving met manuele versnellingen, mechanische handremmen – bij veel hogere snelheden. IntuEdrive noemt dit de snelheid-veiligheid trade-off. De veiligheid kan beperkt worden verhoogd door componenten toe te voegen (bv. Bosch e-bike ABS), maar de functionaliteit van deze systemen blijft beperkt. Er is een meer holistische aanpak nodig. Bovendien bieden elektrische fietsen vandaag nog niet het gebruiksgemak en de betrouwbaarheid die de consument gewend is van zijn wagen.



Figuur 1: Snelheid-veiligheid trade-off (bron: IntuEdrive)

IntuEdrive's *Ellio* is een snelle elektrische fiets die veiliger is dan de klassieke mechanische fiets, dankzij de innovatieve tweewielaandrijving en elektrische remfunctie. Dit systeem reduceert de stopafstand met 60% en maakt schakelen overbodig (automatische versnellingen). Het stapt ook af van de onderhoudsintensieve fietscomponenten (ketting, tandwielen, mechanische remmen). Dit maakt Ellio de perfecte e-bike voor woon-werkverkeer: makkelijk, veilig en betrouwbaar.

Op 1 augustus van het jaar	2014	2015	2016	2017	2018
Personenwagens	**5.555.499**	**5.623.579**	**5.712.061**	**5.785.447**	**5.853.782**
- rijdend op benzine	2.029.688	2.091.327	2.199.038	2.335.349	2.518.942
- rijdend op diesel	3.458.424	3.457.526	3.424.592	3.339.034	3.193.658
- rijdend op gas	22.051	18.967	17.238	15.965	15.500
- met elektrische motor	1.792	2.871	4.368	6.552	9.244
- hybride	23.444	32.151	44.364	63.740	87.012
- niet nader bepaald	20.100	20.737	22.461	24.807	29.426
Inwoners per personenauto op 1 augustus (d)	2,01	1,99	1,97	1,96	1,94

Figuur 2: Grootte van het voertuigenpark 2014-2018 (bron: statbel.fgov.be)

Door automatisch te schakelen zorgt Ellio ervoor dat de fietser in elke situatie precies zo snel trapt als hij of zij wil. Deze gewenste trapsnelheid – of beter trapcadans – varieert van persoon tot persoon en hangt af van omstandigheden zoals helling, tegenwind en rijsnelheid. Omdat deze gewenste cadans niet op voorhand gekend is, schakelt de transmissie momenteel op basis van een vaste wetmatigheid die tijdens testen getuned is om voor zoveel mogelijk gebruikers comfortabel aan te voelen. Wijkt deze wetmatigheid af van de gewenste cadans van een specifieke gebruiker, dan kan deze gebruiker via knoppen op het stuur tijdens het fietsen zijn of haar cadans manueel aanpassen.

1.2 Machinaal leren voor geïndividualiseerde cadanscontrole

Deze thesis werkt verder op het Ellio-prototype van IntuEdrive. Zoals reeds aangehaald schakelt de fiets automatisch. De trapcadans wordt hierdoor stabiel gehouden op een referentiecadans, ook wanneer de fietser harder of zachter trapt. Het doel is om deze referentiecadans te personaliseren door de voor de gebruiker ideale cadans (ook *Freely Chosen Cadence (FCC)* genoemd) in real time te voorspellen aan de hand van de toestand van de fiets. Op die manier past de trapsnelheid zich niet alleen aan de omstandigheden aan, maar ook aan de individuele gebruiker. Deze implementatie zal ervoor zorgen dat de fietser minder frequent zelf de trapcadans moet bijstellen. Om de cadans te personaliseren en dynamisch te maken, zal een algoritme van machinaal leren ontwikkeld worden. Dit algoritme krijgt de toestand van de fiets als input binnen. Daarmee wordt de FCC berekent. Wanneer de fietser besluit om de cadans manueel aan te passen, interpreteert het algoritme dit als een signaal om bij te leren. De FCC kwam op dat moment immers niet overeen met de referentiecadans.

Om de performantie van het algoritme van machinaal leren te testen zal het volledig systeem fiets-fietser-cadanscontrole gesimuleerd worden. Het fietsmodel wordt geleverd door IntuEdrive en zal geïmplementeerd worden in Python. Vervolgens worden een aantal algoritmes van machinaal leren vergeleken op basis van een aantal vooraf gedefinieerde performance indicatoren. De algoritmes zijn afkomstig uit scikit-learn, een bibliotheek

van machinaal leren.

De cadanscontrole moet aan verschillende eisen voldoen. Het algoritme moet draaien op een Raspberry Pi, samen met het controleprogramma van de fiets. Door deze beperkte resources moet het algoritme zo efficiënt mogelijk zijn. De voorspellingen moeten bijna in real time berekend worden. Het doel is om aan 10Hz de cadans aan te passen, maar hoe meer voorspellingen per seconde, hoe beter. Tragere voorspellingen kunnen hinderlijk zijn voor het rijgedrag. Ten slotte moet er ook rekening gehouden worden met de veiligheid van de fietser. Opeenvolgende voorspellingen mogen niet te veel van elkaar verschillen, anders zou de fietser erdoor gestoord kunnen worden en zijn concentratie verliezen. Bovendien mag de cadans nooit hoger dan een bepaalde maximum limiet ingesteld worden.

De algoritmes worden geëvalueerd op basis van de gemiddelde kwadraten fout tussen de referentiecadans - afkomstig van het algoritme van machinaal leren - en de FCC van de fietser. De FCC is niet precies gekend en wordt in de simulatie bepaald aan de hand van een fietsersmodel. Dit is een functie die de toestand van de fiets en de fietser (rijsnelheid, helling,...) afbeeldt op de FCC. Simpel gezegd is het fietsersmodel een functie met als input de toestand van de fiets en als output een "optimale cadans". Deze functie is speculatief en kan makkelijk aangepast worden. Op welke basis de fietser precies zijn freely chosen cadence bepaalt is voor dit onderzoek weinig relevant. Het gaat er hier vooral om dat algoritme van machinaal leren het fietsersmodel kan achterhalen.

$$\text{Fietsersmodel:} \qquad \text{fcc} = \text{f(snelheid,koppel,vermogen,helling,...)}$$

Het algoritme moet kunnen bijleren met een kleine hoeveelheid data. De gebruiker zal immers niet vaak manuele aanpassingen doen aan de cadans. Te veel data gebruiken kan een negatieve invloed hebben op reeds correcte voorspellingen. Het algoritme moet ook snel bijleren. Elke verandering moet zo snel mogelijk doorgevoerd worden en moet een betekenisvolle impact hebben.

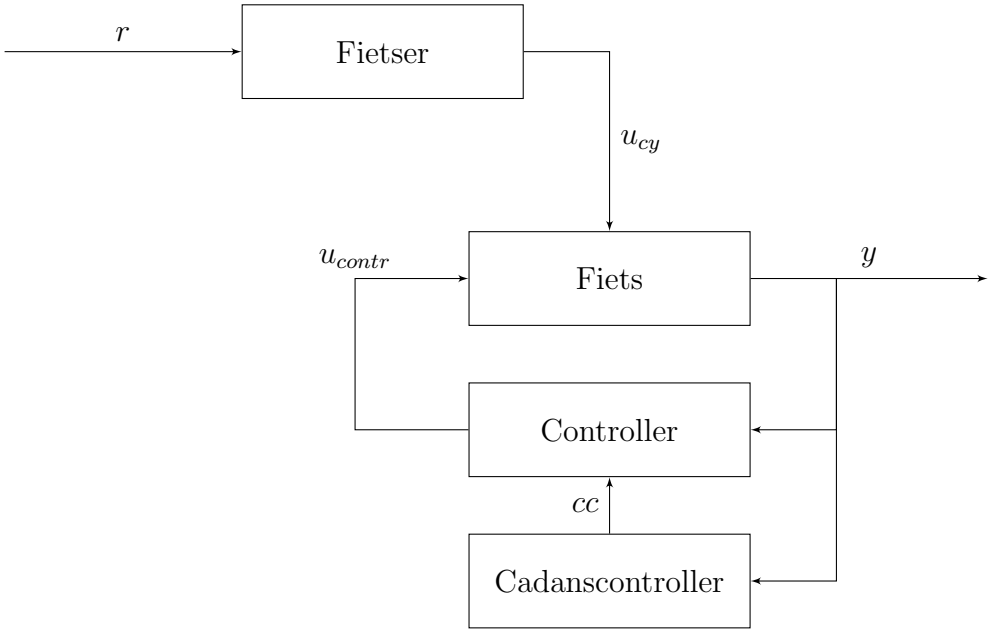
1.3 Huidige systeem

De fiets van intuEdrive gebruikt een elektrische Continu Variabele Transmissie (CVT) die ervoor zorgt dat er naadloos geschakeld kan worden tussen versnellingen, in tegenstelling tot het traditionele ketting-en-tandwiel systeem. Dit oude systeem schakelt in discrete trappen, waardoor de fietser tijdens het schakelen een discontinuïteit voelt. Het CVT-systeem gebruikt 2 motoren en schakelt traploos. Eén van de motoren regelt de trapcadans, de andere motor regelt het ondersteuningsniveau. Het ondersteuningsniveau bepaalt hoeveel extra elektrisch vermogen er geleverd wordt, bovenop wat de fietser zelf levert.

Figuur 3 toont een blokdiagram van het systeem fiets-fietser-controller. We gaan ervan uit dat de fietser op elk moment een bepaalde referentiesnelheid (v_{ref}) probeert te halen, hier aangeduid met r. Die kan variëren naargelang de situatie, maar is voor elke gebruiker anders. Tijdens het fietsen geeft de fietser input aan de fiets (u_{cy}). Zo kan hij of zij het geleverde koppel variëren (T_{cy}) – i.e. meer of minder kracht op de pedalen

zetten – of de cadans aanpassen met de knoppen (u_c). Inputs en fysische toestand van de fiets worden gemeten door sensoren op de fiets: het koppel ($T_{cy,m}$), de hoek van de trapas (θ_{cr}), snelheid (v_{bike}), helling (α), etc. T_{cy} en $T_{cy,m}$ zijn niet hetzelfde, want er kunnen fouten gebeuren tijdens het meten. De vector van meetwaarden (y) is input voor de fietscontroller. De fietscontroller stuurt de motoren in de E-bike aan (u_{contr}) op basis van de metingen y en de ingestelde referentiecadans. De cadanscontroller die in deze thesis uitgewerkt zal worden, zal op basis van dezelfde metingen een gepersonaliseerde referentiecadans (FCC_{est}) voorspellen die als input dient voor de controller.

$$r = \begin{bmatrix} v_{ref} \end{bmatrix} \qquad u_{cy} = \begin{bmatrix} T_{cy} \\ u_c \end{bmatrix} \qquad cc = \begin{bmatrix} FCC_{est} \end{bmatrix} \qquad y = \begin{bmatrix} \theta_{cr} \\ T_{cy,m} \\ v_{bike} \\ \alpha \end{bmatrix}$$



Figuur 3: Blokdiagram van het fiets-fietser-controller systeem

1.4 Gerelateerd werk

1.4.1 Hardware Implementation and Control Strategy of a High Dynamic CVT Transmission for an E-Bike: Jorrit Heidbuchel (2016-2017)

Jorrit werkte in zijn masterproef aan het verbeteren van de traditionele elektrische fiets. Elektrische fietsen gebruiken een elektrische motor om extra vermogen toe te voegen, bovenop het vermogen dat de fietser zelf levert. De klassieke fietssystemen die op E-bikes gebruikt worden, hebben hun beperkingen. Ten eerste kan er enkel geremd worden met behulp van mechanische remmen. Dit remsysteem is gevoelig voor wielslip en laat niet toe om remenergie te recupereren. Daarnaast schakelt de traditionele elektrische fiets in discrete stappen. Om deze tekortkomingen op te lossen, werkte Jorrit een continu variabele transmissie uit die ook toelaat om op de elektrische motoren af te remmen. Zo kan er volledig automatisch en traploos geschakeld worden en kan bovendien elektrische energie worden gerecupereerd tijdens een remmanoeuvre.

Jorrit's masterproef werkte verder op thesissen in het kader van ir. Tomas Keppens' concept voor een dergelijke aandrijving. Die aandrijving was uitgebreid gesimuleerd en moest worden uitgewerkt tot een eerste prototype. Het eerste IntuEdrive prototype was het resultaat van Jorrits masterproef.

1.4.2 Factors affecting cadence choice during submaximal cycling and cadence influence on performance: Ernst A. Hansen en Gerald Smith (2009)

Hansen en Smith bestudeerden de factoren die de cadans keuze van een fietser beïnvloeden. Factoren zoals hellingsgraad, leeftijd, fietsvermogen, duur en vele anderen blijken invloed te hebben op de freely chosen cadance van een fietser.

Hansen en Smith introduceren de termen *freely chosen cadence (FCC)* en *energetically optimal cadence (EOC)*. De FCC is de cadans die de fietser zelf kiest als meest comfortabele omwentelingssnelheid.. De EOC is de cadans waarbij de zuurstofopname optimaal is. Tijdens fietsen op lage intensiteit, kiezen fietsers een cadans die hoger is dan de energie optimale cadans en dus fietsen ze minder energie efficiënt. Tijdens fietsen op hoge intensiteit, kiezen fietsers een cadans die dichter ligt bij de energie optimale cadans wat leidt tot betere prestaties. De mens fietst dus niet altijd op een energie-efficiënte manier. Dit is een verschil met het stapgedrag van de mens. De gekozen stapcadans (*freely chosen step cadence*) is wel energie optimaal. Vanuit een evolutionair standpunt is dit logisch. We stappen en lopen al duizenden jaren. De fiets is in dit opzicht nog een recente uitvinding.

1.4.3 Adaptive machine learning algorithms for data streams subject to concept drifts: Pierre-Xavier Loeffel (2018)

Loeffel haalt in zijn thesis verschillende manieren aan om om te gaan met conceptuele drift in data streams. Conceptuele drift is de verandering van het concept na verloop van

tijd. Loeffel geeft de voor- en nadelen van verschillende technieken zoals: *sliding window, sampling* en *fading factor*. Alle technieken implementeren een vorm van "vergeten" en zijn mogelijk blind of geïnformeerd. Sliding window en sampling zijn vormen van abrupt vergeten en fading factor is een vorm van geleidelijk vergeten.

Naast de reeds aangehaalde technieken, introduceert Loeffel een nieuw algoritme, "the Droplet Ensemble Algorithm (DEA)", voor classificatieproblemen. DEA, in tegenstelling tot andere ensemble algoritmes, leert de expertise-regio's van de onderliggende algoritmes van machinaal leren. Het selecteert dynamisch een subset van deze onderliggende algoritmes om een betere voorspelling te maken. Zo kunnen betere voorspellingen gemaakt worden.

Hoofdstuk 2

Methode

2.1 De fietssimulatie

Er wordt een simulatie gemaakt die de toestand van de fiets zo goed mogelijk benadert. Er zal geen rekening gehouden worden met het manoeuvreren van de fiets of met tegenwind. Enkel de relevante meetwaarden worden bijgehouden. De simulatie is geparametriseerd om eenvoudig verschillende scenario's te testen. De simulatie is gebaseerd op werk van [1].

Het voordeel van de fietssimulatie is de enorme flexibiliteit. Uren aan data kunnen in een moment tijd gegenereerd worden, waardoor het makkelijk is om verschillende tests uit te voeren. Hiervoor moeten slechts enkele instellingen aangepast worden. Het is ook mogelijk om slechts een enkele parameter aan te passen tijdens tests, terwijl de rest constant blijft (*ceteris paribus*), wat praktisch onmogelijk is in een veldtest. Omdat de simulatie bovendien een duidelijke referentie genereert voor de FCC (output van het fietsersmodel), kan de performantie van de cadanscontroller op een kwantitatieve manier worden geëvalueerd. Tijdens een veldtest zou de fietser alleen kwalitatief kunnen aangeven of hij of zij de voorspelde cadans goed vindt.

2.2 Modelleren van het fietserkoppel

Het fietserkoppel wordt gemodelleerd als een sinusfunctie met twee pieken per omwenteling van de trapas (2 benen), met het DC-koppel van de fietser als parameter. Direct Current (DC) binnen de signaaltheorie wordt gezien als het signaal op nul Hertz. Het DC-koppel is vergelijkbaar met het gemiddeld koppel.

$$T_{cy} = T_{dc}(1 + sin(2\theta_{cr} - \frac{\pi}{6}))$$

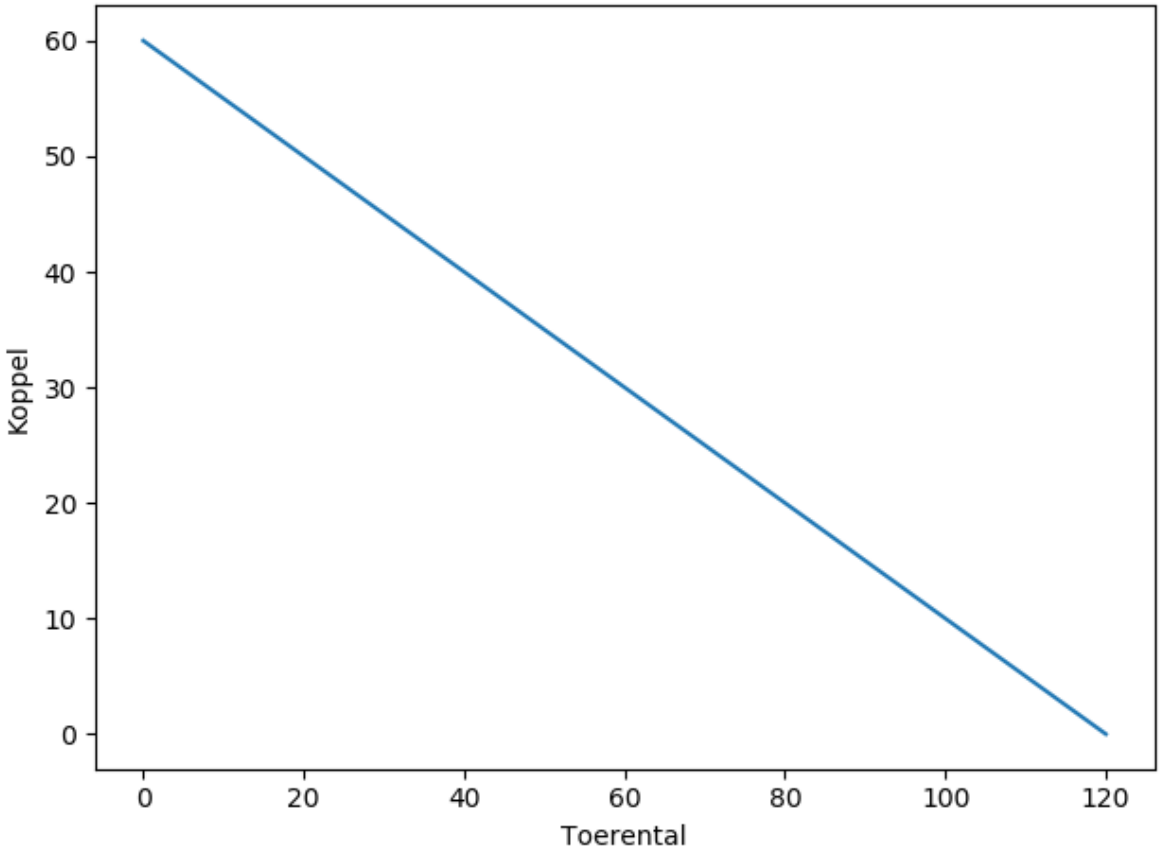
Uit deze formule is het ook meteen duidelijk dat het DC-koppel ook het vermogen-equivalent koppel is. Dat wil zeggen dat het DC-koppel gedurende een volledige omwenteling van de trapas evenveel arbeid levert als het fietserkoppel.

$$\begin{aligned}\int_0^{2\pi} T_{cy}(1 + sin(2x - \frac{\pi}{6}))dx &= T_{cy} \int_0^{2\pi} (1 + sin(2x - \frac{\pi}{6}))dx \\ &= T_{cy} \left[x - \frac{1}{2} sin(\frac{1}{3}(6x + \pi)) \right]_0^{2\pi} \\ &= 2\pi \ T_{cy}\end{aligned}$$

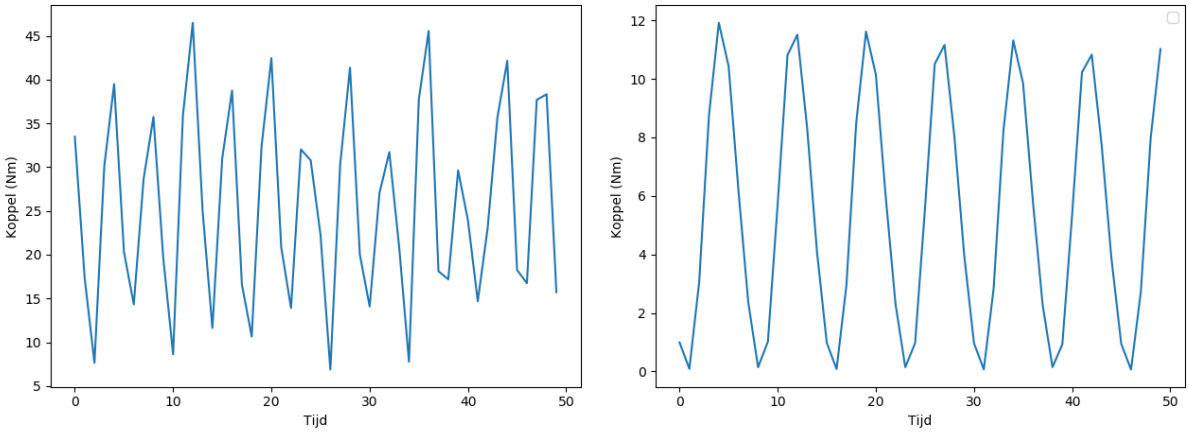
Het gemiddelde koppel geleverd door de fietser wordt gemodelleerd als een proportionele regelaar. Het doel is om een bepaalde snelheid, v_{ref}, te behalen. Hoe groter het verschil is tussen de referentiesnelheid en de eigenlijke snelheid, hoe meer kracht er geleverd zal worden. Als deze referentiesnelheid overschreden wordt, dan zal er geen koppel meer geleverd worden. Dit wordt ook wel freewheelen genoemd. Om de kracht van de actor te limiteren, wordt er een maximum koppel ingesteld ($T_{dc,max}$) naar gelang de huidige cadans (ω_{cr}). Zo wordt er meer kracht geleverd wanneer de cadans laag is, net zoals in de werkelijkheid. K bepaalt de agressiviteit van de regelaar. De formules zien er als volgt uit:

$$T_{dc,max} = \frac{-\omega_{cr}}{2} + 60 \qquad (Figuur\ 4)$$
$$T_{dc} = min(T_{dc,max}, max(0, -K * (v_{bike} - v_{ref}))$$

Figuren 5a en 5b tonen een menselijk koppelverloop en gesimuleerd koppelverloop, gesampled aan 10 Hz. Zoals te zien is het gesimuleerde koppel heel consistent. Het menselijk koppel volgt duidelijk een cyclische functie, maar vertoont vormen van inconsistentie. Merk wel op dat er telkens een afwisseling is van een hoge en een lage piek. Dit wijst op een dominant been. Figuur 5c toont een gesimuleerd koppelverloop van een fietser met een dominant been.

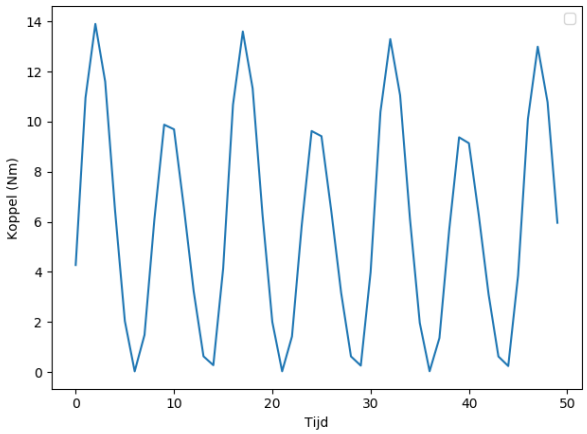


Figuur 4: Het koppel-toerentalkarakteristiek



(a) Menselijk koppelverloop

(b) Gesimuleerd koppelverloop

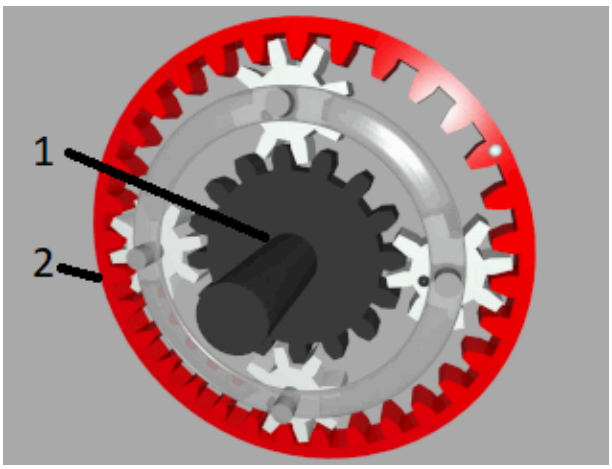


(c) Gesimuleerd koppelverloop met dominant been

Figuur 5: Het koppelverloop van een mens (linksboven), de simulatie (rechtsboven) en een gesimuleerd dominant been (onderaan)

T_{cy} is het koppel op de trapas. Dit moet nog overgebracht worden op het achterwiel. Ellio maakt gebruik van een planeetwielmechanisme (figuur 6). Dit mechanisme laat toe om een grote overbrengingsverhouding te voorzien in een kleine ruimte. Het achterwielkoppel wordt beïnvloed door het aantal tanden op het zonnewiel (1; ns) en het ringwiel (2; nr) en de overbrengingsverhouding tussen de trapas en het ringwiel ($k_{cr,r}$). Het koppel op het achterwiel (T_{rw}) ziet er als volgt uit:

$$T_{rw} = T_{cy} * k_{cr,r} * \frac{nr + ns}{nr}$$



Figuur 6: Planeetwielmechanisme (bron: Wikipedia)

Bovenop het vermogen geproduceerd door de fietser, levert Ellio extra ondersteuning a.d.h.v. een motor (T_{MG2}) gekoppeld aan het voorwiel. De fietser kan zelf een ondersteuningsniveau (S) instellen tussen 0 en 5. Hoe hoger dit ondersteuningsniveau, hoe minder inspanning de fietser moet leveren.

$$T_{MG2} = min(35, S\,.\,T_{cy})$$

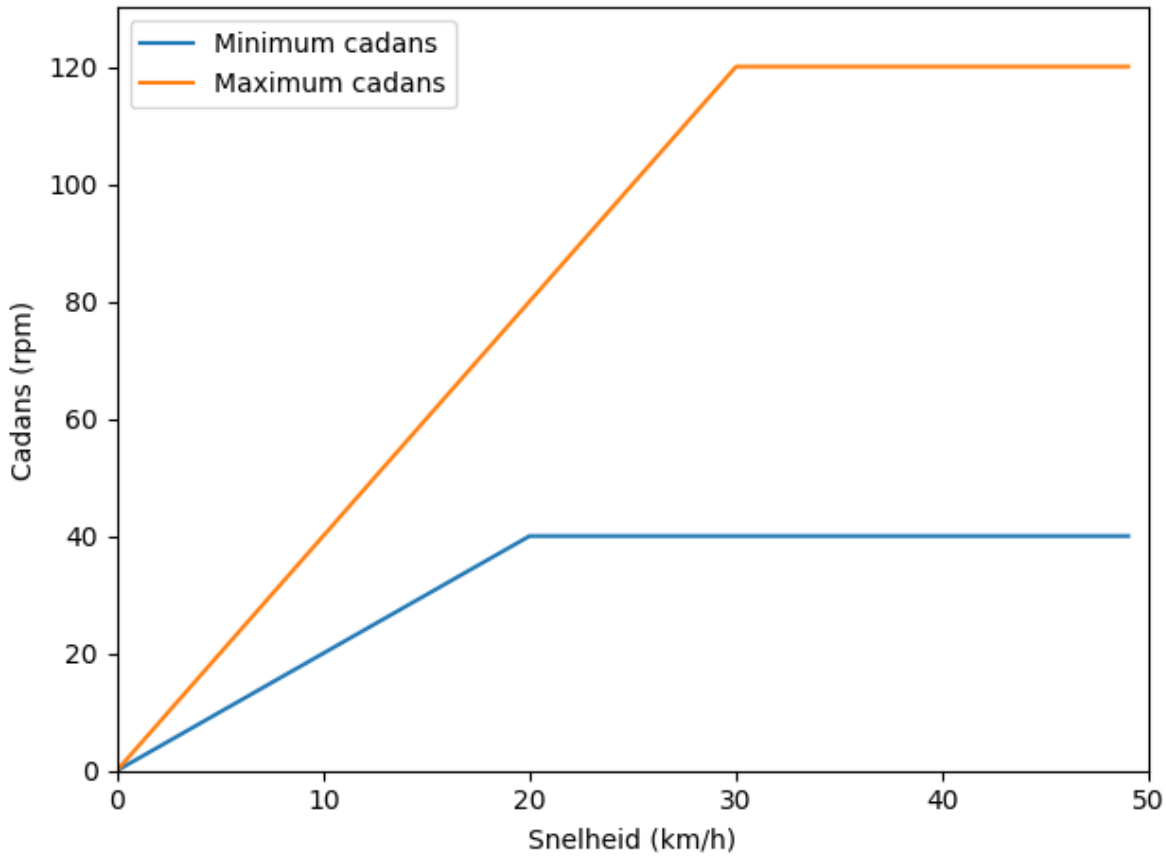
2.3 Het fietsersmodel

Hoe kiest een fietser zijn cadans? Dit is voor elke fietser verschillend en er is nog nauwelijks onderzoek naar gebeurd. Wielrenners trainen om sneller te kunnen trappen omdat dit efficiënter is. Ze kunnen een gemiddeld vermogen leveren van 300 Watt. De doorsnee fietser levert gemiddeld ongeveer 75 Watt tijdens een normale fietstocht. Het fietsersmodel zal hierop worden afgesteld, aangezien wielrenners niet de voornaamste doelgroep zijn voor Ellio.

Het fietsersmodel is een functie die op verschillende manieren uitgedrukt kan worden: op basis van de helling, gemiddeld koppel, of snelheid. Wat het correcte model is wordt in deze thesis niet uitgewerkt. Het is vooral van belang dat de cadanscontroller het model zo snel en zo nauwkeurig mogelijk kan achterhalen, ongeacht wat het model precies is. Hier wordt de volgende aanname gemaakt: hoe hoger het koppel geleverd door de fietser, hoe hoger de gewenste cadans. Wanneer de fietser bijvoorbeeld een helling oprijdt, schakelt hij of zij een versnelling omlaag zodat de kracht die op de pedalen gezet moet worden aangenaam blijft. We stellen hier volgende eenvoudige modellen voor:

$$\begin{aligned} Gemiddeld\ koppel: &\quad fcc = f_k.T_{dc} \\ Helling: &\quad fcc = f_h.\alpha \\ Snelheid: &\quad fcc = f_v.v_{bike} \end{aligned}$$

Er wordt verder aangenomen dat de fietser ook een zekere lineariteit verwacht bij lage snelheden. Dat wil zeggen dat een fietser het niet comfortabel vindt wanneer hij of zij snel moet trappen wanneer de fiets nog stilstaat of heel traag rijdt, ook al moet er op dat moment veel koppel geleverd worden om te kunnen vertrekken. Daarom wordt bij lage snelheden de FCC begrensd door een lineair oplopend maximum, te vergelijken met een mechanische fietsversnelling. Omdat de doorsnee fietser niet heel traag of heel snel trapt wordt de FCC begrensd tussen de 40 en 120 rpm.



Figuur 7: Verwacht cadansverloop in functie van de snelheid.

2.4 Het lastmodel

De simulatie is voorzien van een lastmodel. Zoals in realiteit, werken lasten in op de fiets. Zwaartekracht, wrijving met de weg en luchtweerstand zijn gemodelleerd als volgt:

$$F_{grav} = m \;.\; g \;.\; sin\ \alpha$$
$$F_{friction} = m \;.\; g \;.\; c_r \;.\; cos\ \alpha$$
$$F_{aero} = \frac{c_d \;.\; \rho_{aero} \;.\; A_{aero} \;.\; v_{bike}^2}{2}$$

Samen vormen ze de totale belasting op de fiets.

$$F_{load} = F_{grav} + F_{friction} + F_{aero}$$

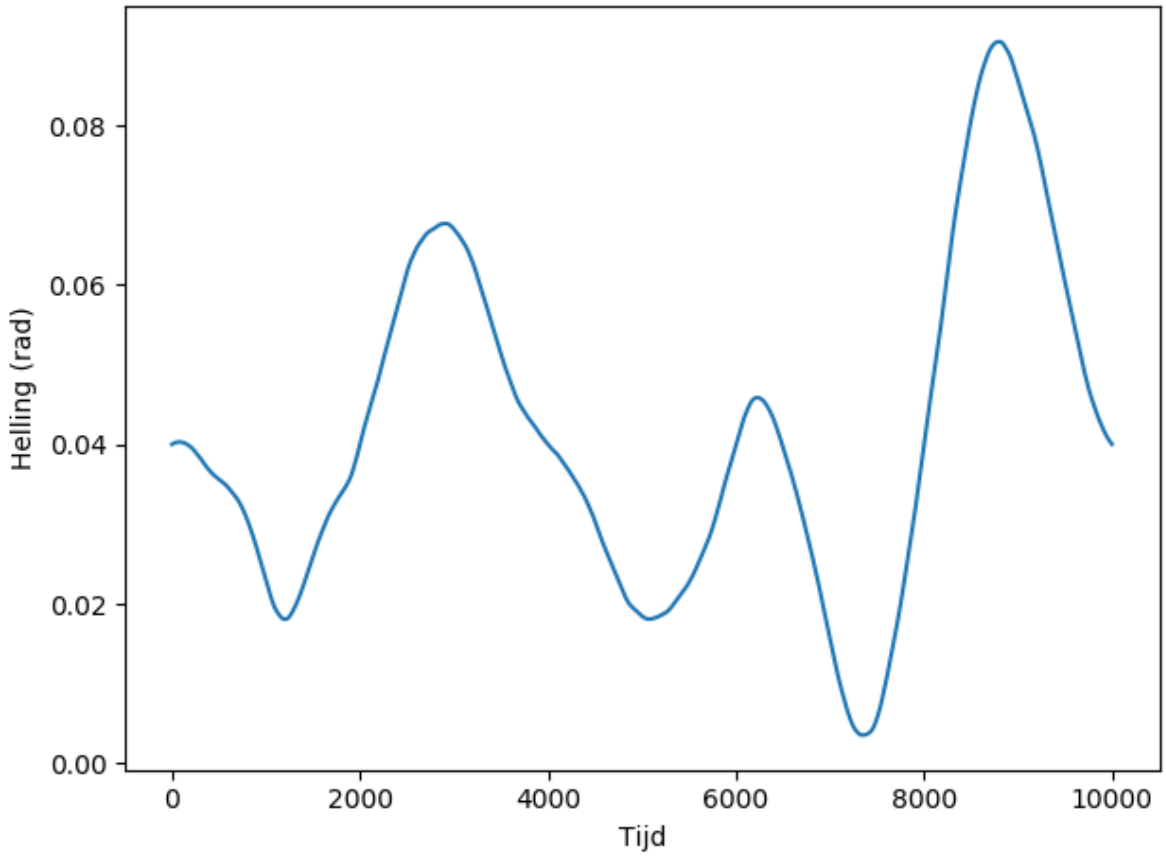
Deze lasten zorgen ervoor dat de simulatie een realistische hoeveelheid vermogen nodig heeft om een bepaalde snelheid te halen. Er wordt hier geen rekening gehouden met de wind. Ten eerste zou dit extra complexiteit toevoegen aan de simulatie. En ten tweede vertrekken we van volgende hypothese:

> De *freely chosen cadence* hangt af van de hoeveelheid last, van welke bron dan ook, die de gebruiker ondervindt.

Het voorgestelde lastmodel omvat deze vereiste. Door de helling en referentiesnelheid te variëren ondergaat de fietser een veranderende last. Zoals in de realiteit zoeken mensen een bepaalde snelheid te halen. Wanneer de fietser een te hoge last ondervindt, bijvoorbeeld door een berg op te rijden, moet hij of zij meer vermogen genereren om zijn of haar gewenste snelheid te behouden. Hiervoor zijn 2 mogelijkheden: het verhogen van het koppel of de trapsnelheid. Mensen zijn meer geneigd om hun gewenste cadans te behouden, ongeacht het koppel (binnen bepaalde grenzen). De formule voor mechanisch vermogen gaat als volgt:

$$P = T_{cy} \;.\omega_{cr}$$

Om het lastmodel correct te laten werken, moet er nog een helling gegenereerd worden. Om veel werk uit te sparen met het uitstippelen van parcours, wordt dit dynamisch gegenereerd met behulp van perlin noise. Perlin noise kan gebruikt worden om willekeurige getallen te genereren waarbij opeenvolgende getallen weinig van elkaar verschillen. Een perfecte kandidaat dus om terrein te simuleren. Om verschillende trajecten te creëren, kan de seed variabele aangepast worden. Een hellingsgraad wordt in de fietswereld vaak percentueel voorgesteld. Deze implementatie heeft echter radialen nodig. De helling zal beperkt worden tussen ≈ 0 en 10% (0 en 0.1 radialen). Ter vergelijking, de Koppenberg heeft een gemiddeld stijgingspercentage van 11.6%. Het minimum stijgingspercentage is zo gekozen dat de simulatie zo weinig mogelijk gaat freewheelen.



Figuur 8: Voorbeeld helling verloop

2.5 Snelheidsvergelijking

De snelheid wordt geïntegreerd met een voorwaartse Euler methode. De acceleratie is een functie van de last, het totaalgewicht (m), het koppel geleverd door de fietser op het achterwiel (T_{rw}) en het koppel van een motor bevestigd op het voorwiel (T_{MG2}).

De bewegingsvergelijking van de fiets is, met inbegrip van het lastmodel en het fietsersmodel:

$$F \ = \ m \ . \ a$$

Deze vergelijking wordt elke tijdsstap geïntegreerd met behulp van een voorwaartse Euler methode:

$$F = m.(\frac{v_{bike}[h] - v_{bike}[h-1]}{\Delta t})$$

$$\frac{F.\Delta t}{m} = v_{bike}[h] - v_{bike}[h-1]$$

$$v_{bike}[h] = v_{bike}[h-1] + \Delta t.\frac{1}{m}.F$$

$$v_{bike}[h] \ = \ v_{bike}[h-1] \ + \Delta t \ .\frac{1}{m} \ . \ (\frac{T_{MG2} \ + \ T_{rw}}{r_w} \ - \ F_{load})$$

De volledige simulatie ziet er als volgt uit:

Algoritme: Fietssimulatie

for $h = 1, \#tijdssprongen$ **do**

$T_{dc,max} = \frac{-\omega_{cr}[h-1]}{2} + 60$

$T_{dc} = min(T_{dc,max},\ max(0, -K * (v_{bike}[h-1] - v_{ref})))$

$fcc = f(T_{dc})$

$\omega_{cr} = cadans(v_{bike}[h-1],\ T_{dc},\ fcc)$

$\theta_{cr} = \theta_{cr}[h-1] + \Delta t\ .\ \omega_{cr}$

$T_{cy} = T_{dc}(1 + sin(2\theta_{cr} - \frac{\pi}{6}))$

$T_{rw} = T_{cy} * k_{cr,r} * \frac{nr+ns}{nr}$

$T_{MG2} = min(35,\ S\ .\ T_{cy})$

$F_{grav} = m\ .\ g\ .\ sin\ \alpha$

$F_{friction} = m\ .\ g.\ c_r\ .\ cos\ \alpha$

$F_{aero} = \frac{c_d\ .\ \rho_{aero}\ .\ A_{aero}\ .\ v_{bike}[h-1]^2}{2}$

$F_{load} = F_{grav} + F_{friction} + F_{aero}$

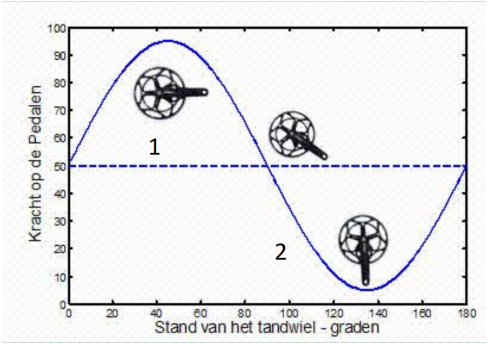
$v_{bike} = v_{bike}[h-1] + \Delta t\ .\frac{1}{m}\ .\ (\frac{T_{MG2} + T_{rw}}{r_w} - F_{load})$

end

2.6 Voorspellen van de cadans

Een groot deel van de toestand van de fiets wordt berekend. Om een efficiënt algoritme te creëren moet enkel de relevante data bekeken worden. Zo worden de volgende attributen gebruikt: snelheid, koppel, hoek van de trapas en helling.

De helling is vanzelfsprekend. Dit is de voornaamste vorm van last en zal dus een impact hebben op de freely chosen cadence van de fietser. De hoek van de trapas op zich heeft niet veel betekenis. En enkel het koppel ook niet. Er kan bijvoorbeeld een koppel geleverd worden van 20Nm. Dit koppel kan op verschillende plaatsen geleverd worden in de trapcyclus. Als dit in de situatie 2 geleverd wordt, dan is dit waarschijnlijk het laagste koppel in de trapcyclus. Wordt dit geleverd gedurende de neergaande beweging (1), dan is dit het hoogste koppel. Deze verschillende situaties zullen een verschillende cadans nodig hebben. Tijdens de eerst situatie wordt er gemiddeld meer koppel geleverd, wat wijst op een grote last. Dus verwachten we hier een hoge cadans. De tweede situatie daarentegen zal gemiddeld een lagere last hebben. Daarom wordt in conjunctie met het koppel de hoek van de trapas gebruikt. Snelheid is ook een relevant attribuut. In situaties met verschillende snelheden en dezelfde last gaat de cadans verschillen.



Figuur 9: Evolutie koppel in functie van hoek trapas (bron: fietsica.be)

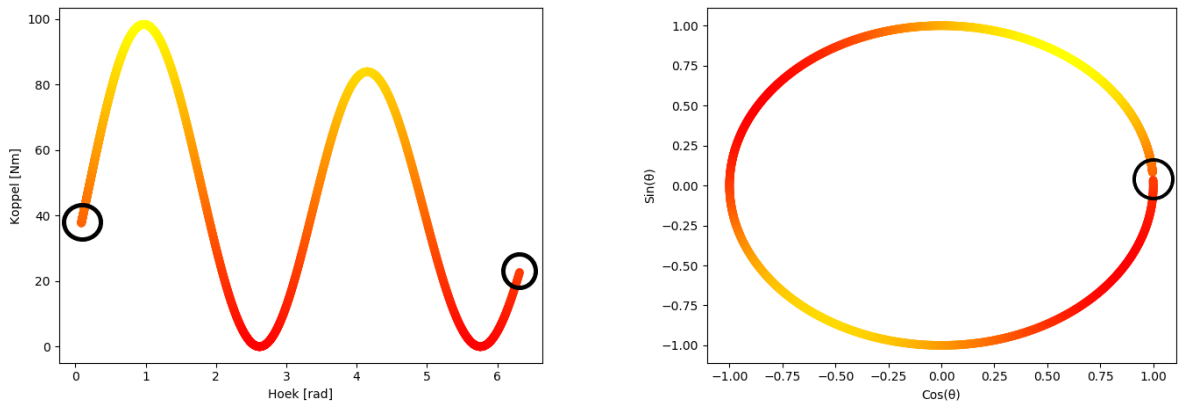
2.7 Preprocessing

Niet elk algoritme heeft nood aan genormaliseerde input. Voor algoritmes die een afstandsfunctie gebruiken is standaardisatie cruciaal. De doelvariabelen daarentegen moet niet genormaliseerd worden volgens Warren S. Sarle [4]. In zijn FAQ schrijft hij dat het standaardiseren van de output voornamelijk voordelige effecten heeft op de initiële gewichten. Wanneer er meerdere doelvariabele zijn en als deze ver uit elkaar liggen, kan het wel nuttig zijn om ze te normaliseren. In dit geval is dat niet nodig aangezien enkel de cadans voorspeld zal worden.

Ten eerste zal de data in sequentie gegoten worden. Een sequentie wordt gezien als een aantal vectoren (x_t) over verschillende tijdstippen. Een enkele data meting op zich is niet genoeg om een accurate voorspelling te maken, maar te veel data gebruiken is ook niet goed aangezien de voorspellingen tijdig moeten geleverd worden.

$$x_t = \begin{bmatrix} \theta_{cr} \\ T_{cy,m} \\ v_{bike} \\ \alpha \end{bmatrix} \qquad sequentie = \begin{bmatrix} x_t \\ x_{t-1} \\ \dots \\ x_{t-n} \end{bmatrix}$$

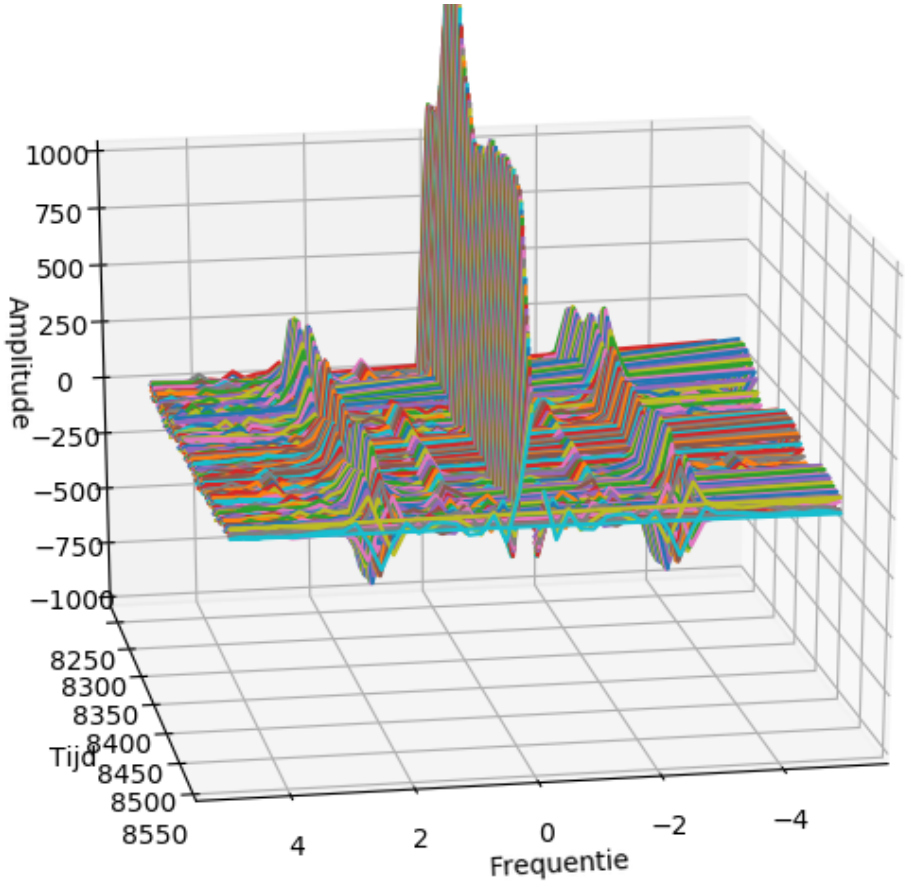
Ten tweede zal de hoek van de trapas niet in zijn zuivere vorm gebruikt worden. De hoek van de trapas is een variabele tussen nul en twee pi. Het probleem hier is dat het begin en het einde van een cyclus ver uit elkaar liggen. Voor de mens is het evident dat nul en twee pi hetzelfde zijn, maar voor de computer is dit een groot verschil. Daarom zal de sinus en de cosinus van de hoek genomen worden, zodat het begin en einde dicht bij elkaar liggen (figuur 10). Zo leren we het algoritme bij dat de data zich cyclisch gedraagt.



Figuur 10: Figuur links toont dat het begin en einde van een trapcyclus ver uit elkaar liggen. Figuur rechts toont dat beide punten van de linkse figuur dicht bij elkaar liggen.

Ten slotte kan er ruis zitten op de metingen. Er zal altijd wel een klein foutje zitten op de data omdat de meetapparatuur niet perfect is. Het is mogelijk dat trillingen van de motor of het wegdek een impact kunnen hebben, voornamelijk op het geleverde koppel. Voor deze iteratie zal hier geen rekening mee gehouden worden. Ruis van het wegdek komt voor op een frequentie van ongeveer 20 Hz en hoger. Dit kan nog correct opgenomen

worden als er op voorhand data wordt gesampled op hogere frequentie. Ruis van de motoren daarentegen komt voor op 13000 Hz, ver boven de sample frequentie en zal dus afgebeeld worden op lagere frequenties. Het huidige systeem zal geen rekening houden met beide soorten ruis. Om de gevoeligheid voor ruis te minimaliseren kan het ingangssignaal eenvoudig gefilterd worden met een laagdoorlaatfilter van bijvoorbeeld 10 Hz.



Figuur 11: Een Fast Fourier Transformatie van het menselijk koppelverloop.

2.8 Algoritmes

2.8.1 Passive Aggressive algorithm

Het Passive Aggressive (PA) algoritme, beschreven in de paper van Crammer et al. [5], is een on-line algoritme gelijkaardig aan een perceptron. Net zoals de perceptron, doet PA de matrixvermenigvuldiging $y_t = w_t \cdot x_t$ om de voorspelling te berekenen. Het grootste verschil tussen beide is hoe de gewichten geüpdatet worden. Het PA algoritme is bruikbaar voor classificatie, regressie, uniclass voorspellingen en multiclass problemen.

Het PA algoritme, zoals de naam weggeeft, kan zich zowel passief als agressief gedragen. Het trainen van PA bestaat uit twee stappen. In de eerste stap wordt er een voorspelling $y_{t,p}$ gemaakt a.d.h.v. de input vector x_t en gewichtenmatrix w. Hierna wordt het echte label y_t bekendgemaakt. Als de fout kleiner is dan een voorgedefinieerde waarde ϵ, dan

zullen de gewichten niet geüpdatet worden. Als de error toch groter is dan deze marge, dan zullen de gewichten w aangepast worden zodat de fout voor de huidige instantie nul wordt. PA past de gewichten aan zodat het verschil tussen het vorige gewicht en het nieuwe gewicht minimaal is.

$$loss_\epsilon(w_t; (x_t, y_t)) = \begin{cases} 0 & |w \cdot x - y| \leq \epsilon \\ |w \cdot x - y| - \epsilon & anderzijds \end{cases}$$

$$w_{t+1} = \underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}||w - w_t||^2 \qquad zodat\ loss_\epsilon(w_{t+1}; (x_t, y_t)) = 0$$

Door de agressiviteit van het standaard PA algoritme kunnen er problemen ontstaan wanneer er veel ruis zit op de data. Daarom heeft Crammer et al. twee extra versies, PA-I en PA-II, gemaakt die dit probleem oplost. Beide versies voegen een slack variabele ξ toe. Deze variabele zorgt ervoor dat bij het aanpassen van de gewichten, de fout kleiner of gelijk moet zijn aan ξ in plaats van nul. Beide versies hebben ook een agressiviteits parameter C die deze ξ beïnvloedt. Dit is een vorm van regularisatie om overfitting te voorkomen.

$$PA - I \qquad w_{t+1} = \underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}||w - w_t||^2 + C\xi \qquad zodat\ loss_\epsilon(w_{t+1}; (x_t, y_t)) \leq \xi$$

$$PA - II \qquad w_{t+1} = \underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}||w - w_t||^2 + C\xi^2 \qquad zodat\ loss_\epsilon(w_{t+1}; (x_t, y_t)) \leq \xi$$

De nieuwe gewichtenmatrix w_{t+1} wordt als volgt berekent:

$$w_{t+1} = w_t + \tau_t y_t x_t$$

$$\begin{aligned} \tau_t &= \frac{loss_t}{||x_t||^2} && (PA) \\ \tau_t &= min\{C, \frac{loss_t}{||x_t||^2}\} && (PA - I) \\ \tau_t &= \frac{loss_t}{||x_t||^2 + \frac{1}{2C}} && (PA - II) \end{aligned}$$

2.8.2 Decision Tree en Random Forest

Een Decision Tree (DT) is een rule-based model. Dit algoritme is snel (greedy), maar kan niet goed om met ruis. Dit model is gekend om makkelijk te overfitten. Daarom wordt het Random Forest (RF) algoritme simultaan bekeken.

Een DT is een binaire boom. In elke knoop wordt een binair keuzepunt gemaakt op basis van een attribuut. Dit keuzepunt is gekozen zodat de data optimaal gesplitst is over beide takken. Sci-kit learn biedt de mogelijkheid aan om de maximum diepte van de boom te beperken. Wanneer deze parameter niet ingesteld is, zal de DT blijven groeien totdat alle blad nodes "puur" zijn. Een correcte diepte kiezen is een enorm moeilijke taak.

RF is een ensemble. Dit wilt zeggen dat meerdere algoritmes, in dit geval meerdere DT's, gebruikt worden om een betere voorspelling te maken. L. Breiman [2] beweert is zijn paper dat alle RF's convergeren zodat overfitting geen probleem is. In tegenstelling tot DT's, kiest een RF geen optimaal attribuut wanneer een node gesplitst wordt. De variabele worden at random gekozen, waardoor geen enkele boom dezelfde is. De verschillende bomen trainen niet met exact dezelfde trainingsdata. Ze passen Bootstrap Aggregating toe, of bagging. Dit houdt in dat uit de originele trainingsset data wordt gesampled at random. Een instantie kan meerdere keren gesampled worden. Deze techniek reduceert de variantie.

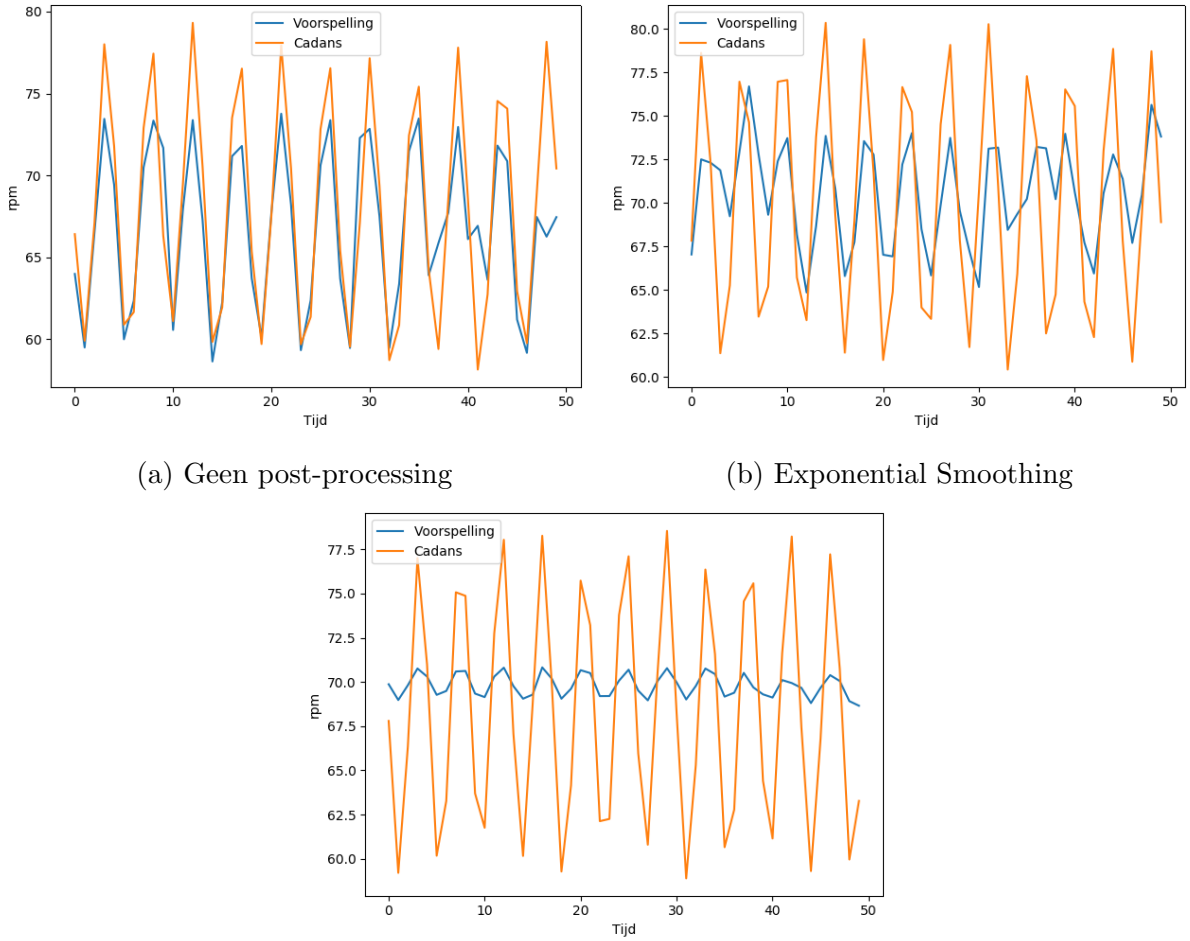
2.9 Post-processing

De cadans variëert lichtjes doorheen een trapcyclus, zowel in het echt als in de simulatie. De voorspellingen zullen dezelfde trend vertonen. Bovendien kan een beetje ruis of het gedrag van de fietser voor afwijkingen zorgen, bijvoorbeeld een grote sprong tussen twee voorspellingen. Dit kan ergerend zijn voor de fietser.

Dit probleem kan op verschillende manieren opgelost worden. Gebruikmakend van de huidige en vorige voorspellingen (FCC_{pred}) of de vorige schatting (FCC_{est}). De vorige instelling is de cadans die is doorgegeven aan de fiets controller.

$$\begin{aligned} \textit{Moving Average } (MA) \qquad & FCC_{est,t} = \frac{\sum_{i=0}^{n} FCC_{pred,t-i}}{n} \\ \textit{Exponential Smoothing } (ES) \qquad & FCC_{est,t} = sf.FCC_{pred,t} + (1 - sf).FCC_{est,t-1} \end{aligned}$$

MA lost beide problemen goed op. Meer voorspellingen leidt tot stabielere schattingen, maar dit introduceert een vertraging (lag) op de cadans. Namelijk wanneer de omstandigheden veranderen, zal de ingestelde cadans slechts na enkele iteraties optimaal zijn, in plaats van onmiddellijk. ES vermindert de amplitude van de oscillaties slechts in mindere mate. De onderstaande figuren tonen wat de impact is van ES en MA op ruizige voorspellingen (20% kans op ruis tussen -10 en +10).



(a) Geen post-processing

(b) Exponential Smoothing

(c) Moving Average

Figuur 12: De effecten van verschillende post-processing technieken.

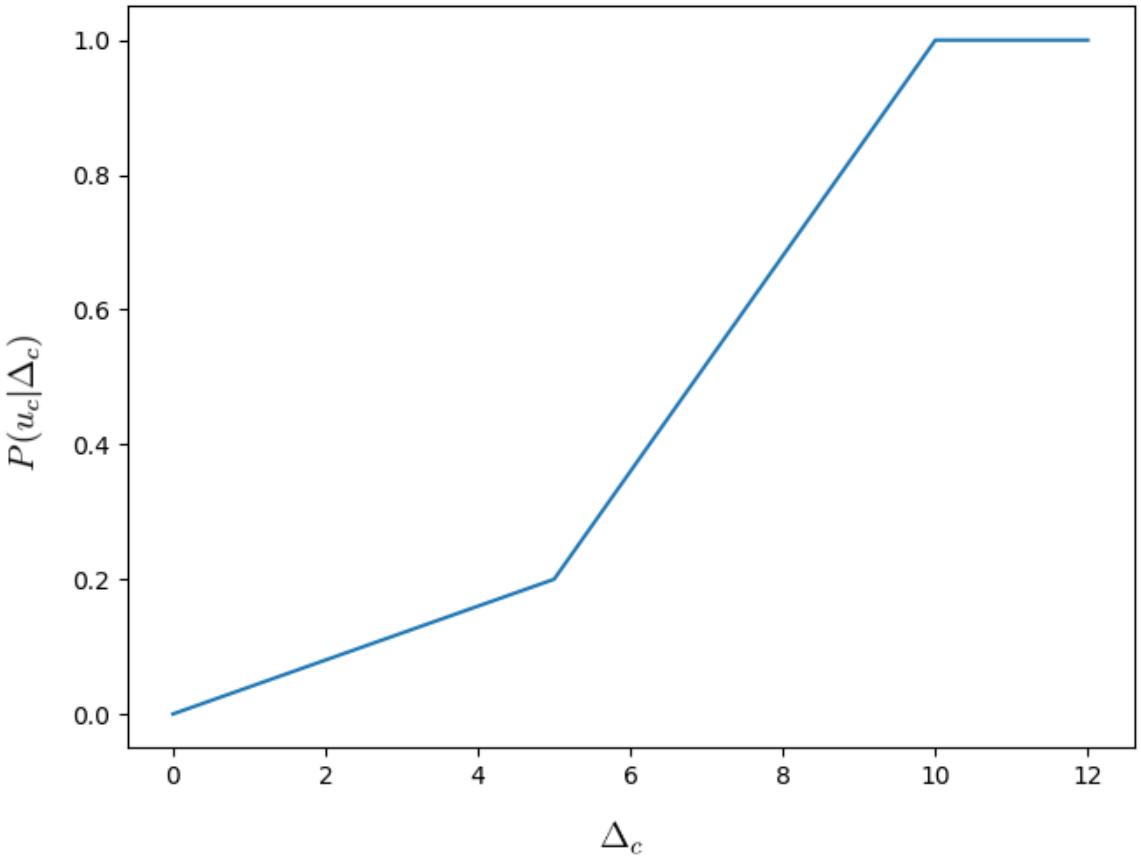
2.10 Stochastisch bijleren

Het volgende hoofdstuk van deze thesis bespreekt de experimentele resultaten. In deze tests leren de algoritmes op een deterministische manier bij. Een mens daarentegen is onvoorspelbaar. Soms is de fietser snel geërgerd en drukt hij sneller op de knop om zijn cadans aan te passen. Soms vindt hij een incorrecte cadans niet zo erg om daarvoor op de knop te duwen. Dit zou gesimuleerd moeten worden om te achterhalen of de modellen met dit non-determinisme omkunnen.

Net zoals volgens de deterministische update-strategie, zal het fietsersmodel als de waarheid genomen worden. Hoe groter het verschil is tussen de cadans en het fietsersmodel (Δ_c), hoe hoger de kans dat er bijgeleerd wordt. De probabiliteit van het bijleren, hier $P(u_c|\Delta_c)$, zal lineair evolueren van 0% naar 20% kans wanneer het verschil vijf bedraagt gaande naar 100% kans wanneer het verschil groter is dan tien. Deze strategie is bedoeld voor een simulatie gesampled aan één Hertz. Als de frequentie verhoogt naar bijvoorbeeld 10000 Hz, evolueert dit stochastische beslissingsmodel naar een deterministische, aangezien er enorm vaak een kans is op de updaten. Daarom moet $P(u_c|\Delta_c)$ nog vermenigvuldigd worden met de grootte van de tijdstap (x0,1 voor 10 Hz). Of het vermenigvuldigen van

de frequentie met de probabiliteit van updaten correct is, is evident aan te tonen met behulp van de verwachtingswaarde over een binomiale verdeling (met $\Delta_c = 5$) [13]:

$$\begin{aligned} E(u_c|\Delta_c)_{1hz} &= p(u_c = 1|\Delta_c)_{1hz} \\ E(u_c|\Delta_c)_{1hz} &= 0.2 \\ E(u_c|\Delta_c)_{10hz} &= 10 * p(u_c = 1|\Delta_c)_{10hz} \\ E(u_c|\Delta_c)_{10hz} &= 0.2 \end{aligned}$$



Figuur 13: Het verloop van de kansverdeling in functie van Δ_c

Hoe lang het gemiddeld duurt om bij te trainen kan ook achterhaald worden met behulp van de verwachtingswaarde, deze keer als een geometrische distributie [14]. Het verschil tussen een binomiale distributie en een geometrische distributie is als volgt: een binomiale distributie is een kansverdeling over het aantal successen met kans p na k onafhankelijke tests. Een geometrische distributie vertelt iets over wat de kans is dat na k pogingen, zodat de laatste poging een succes is met de kans op succes p constant. Bijvoorbeeld: wat is de kans dat de simulatie na tien iteraties aan 10 Hz een update zal doorvoeren als $\Delta_c = 5$?

$$p(u_c = 10|\Delta_c = 5) = (1 - p)^9 * p = 0.98^9 * 0.02 = 0.01667$$

Het gemiddeld aantal iteraties dat de simulatie nodig zou hebben om een update door te

voeren, komt neer op het uitwerken van volgende oneindige serie:

$$E(X) = \sum_{k=1}^{\infty}(1-p)^k pk$$
$$E(X) = p + 2(1-p)p + 3(1-p)^2 p...$$

Wat neerkomt op:

$$E(X) = \frac{1}{p}$$

Voor een Δ_c van vijf zal het dus gemiddeld 5 iteraties (aan 1 Hz) en 50 iteraties (aan 10 Hz) duren vooraleer er geüpdatet wordt.

Ten slotte wordt deze strategie beperkt, zodat het onmogelijk is om direct twee keer na elkaar op de knop te duwen.

2.10.1 Verwachtingen

Er wordt verwacht dat alle modellen dit stochastisch probleem aankunnen, omdat het verschil tussen fietsersmodel en cadans uiteindelijk groot genoeg zal worden zodanig dat het zeer onwaarschijnlijk is dat er niet op de knop geduwd wordt.

2.11 Conceptuele drift

Conceptuele drift is de verandering van het concept, in dit geval het fietsersmodel, na verloop van tijd. Deze verandering kan op verschillende manieren gebeuren: abrupt, geleidelijk en terugkerend. De FCC van een mens is een vorm van geleidelijke drift. Volgens Hansen et al. [7] heeft de leeftijd invloed op de FCC, zijnde oudere mensen trappen in het algemeen trager. De snelheid waaraan de FCC daalt is enorm klein. Het gaat hier om drie rpm per decennium. Eens de fiets is ingesteld, zou er dus technisch gezien nooit meer moeten bijgeleerd worden. Tenzij de fiets verandert van eigenaar. Dit is een vorm van een abrupte drift, aangezien twee individuën hoogstwaarschijnlijk niet dezelfde FCC hebben. Dit onderdeel onderzoekt welke technieken er zijn om met deze abrupte drift om te gaan.

Standaard hebben algoritmes van machinaal leren geen manier om met een drift om te gaan. Als het concept verandert, zal het nieuwe concept geleidelijk aan een groter aandeel krijgen in de trainingsset. Met als gevolg dat het algoritme van machinaal leren uiteindelijk het nieuwe concept leert. Hierdoor kan het lang duren vooraleer het nieuwe concept goed voorspeld kan worden. Een techniek om met conceptuele drifts om te gaan, is het vergeten van oude data (het oude concept). In dit deel zullen drie technieken besproken worden die Loeffel [6] in zijn doctoraat aanhaalt. *Fixed sliding window* en een sampling techniek zijn zelf geïmplementeerd zoals beschreven door Loeffel [6] en Aggarwal [9] respectievelijk.

2.11.1 Fixed sliding window

Een fixed sliding window is de simpelste vorm om een vergeetmechanisme toe te voegen aan een algoritme van machinaal leren voor abrupte drifts. Een fixed sliding window is een set van de n meest recente elementen in een data stream. Wanneer er data in het sliding window zit, nemen we aan dat dit het meest recente concept vertegenwoordigt. Mocht het concept plots veranderen, en het sliding window zit vol, dan zal de oude data plaats maken voor de nieuwe data waardoor het oude concept na verloop van tijd vergeten wordt.

Fixed sliding window	
Voordelen	Nadelen
+ Simpel + Start meteen met vergeten op het moment van de conceptuele drift + Groot window maakt het model robuust tegen ruis en zorgt voor een accuraat en stabiel model	- Een correct grootte voor het window is moeilijk te vinden - Een te klein window kan mogelijk het concept niet goed omvatten - Een te groot window vergeet trager - Ruizige data kan goede, relevante data uit het window verwijderen

Tabel 1: Voor- en nadelen van fixed sliding window

2.11.2 Variabele sliding window

Een variabele sliding window is gelijkaardig aan een fixed sliding window. Het enige verschil is dat de grootte van het window kan variëren. Dit mechanisme introduceert een *change detector*, een algoritme dat moet uitmaken of het concept verandert of niet. Als de change detector geen verandering opmerkt (zelfde concept), zal het sliding window vergroten. Daardoor stijgt het aantal elementen van het huidige concept. Dit brengt het voordeel met zich mee dat het model meer trainingsdata heeft en daardoor accurater en robuust is. Eens de change detector opmerkt dat het concept verandert, dan zal de lengte van het sliding window verkleinen waardoor er een grote hoeveelheid data vergeten wordt. Dit zorgt ervoor dat het model snel het nieuwe concept kan bijleren.Dit is wel een riskante strategie. De change detector kan geactiveerd worden met ruizige data (*"catastrophic forgetting"*). Dit mechanisme kan bovendien ook niet goed om met een zeer trage drift. Al is dit hier niet van belang. Omwille van het risico zal deze strategie niet uitgewerkt worden.

Variabele sliding window	
Voordelen	Nadelen
+ Groot window maakt het model robuust tegen ruis en zorgt voor een accuraat en stabiel model + Kan window verkleinen zodat het nieuwe concept snel bijgeleerd wordt + Change detector kan ervoor zorgen dat een model hergebruikt kan worden (als hetzelfde concept terugkeert)	- Een change detector implementeren is complex - Riskante strategie - Onnauwkeurige change detector zorgt voor problemen - Change detector werkt met een drempelwaarde voor detectie. Te klein en er kan een vals alarm optreden. Te groot en het zal te traag reageren

Tabel 2: Voor- en nadelen van variabele sliding window

2.11.3 Sampling

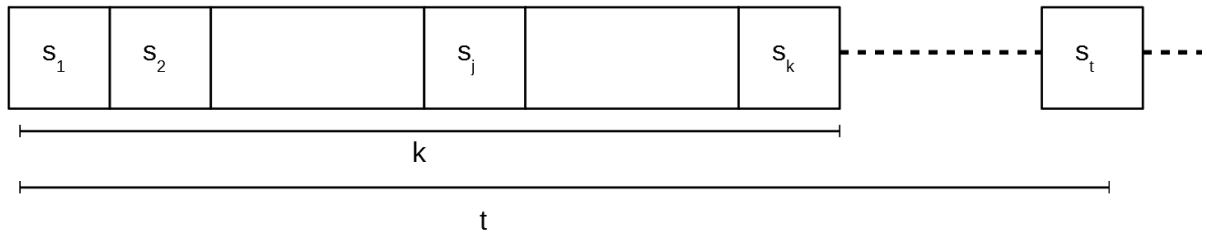
Ten slotte kan er gesampled worden om een relevante trainingset te genereren. Om een bias te krijgen voor recentere data zou een kansverdeling geraadpleegd kunnen worden met een bias naar recentere data. Dit is echter een naïeve implementatie die veel geheugen inneemt, aangezien alle trainingsdata moet bijgehouden worden. In de paper van Aggarwal C. [9] worden twee technieken voorgesteld die data van een stream samplen met een bias voor recente data. Beide technieken zijn gebaseerd op reservoir sampling.

Reservoir sampling

Reservoir sampling is een sampling strategie waar er gesampled wordt uit een stream S. Voor elk element $x \in S$, willen we een gelijke kans hebben dat dit element x gesampled wordt. Stel dat er k elementen gesampled moeten worden, dan zal de probabiliteit voor elk element gelijk zijn aan $k/|S|$. Het probleem hier is dat we niet weten hoe groot S precies is. Bovendien is het mogelijk dat niet alle elementen van S in geheugen passen. Vitter J. [8] lost dit op met een eenvoudig algoritme R.

Algoritme: R

$R = \perp$ met capaciteit k
$t = 0$
while *data stream S heeft nog elementen* **do**
  $t++$
  if $t < k$ **then**
    $A[t] = S[t]$
  else
    $i = random(0, t)$
    if $i <= k$ **then**
      $A[i] = S[t]$
    end
  end
end



Figuur 14: Reservoir sampling voorstelling

Hypothese: $P(s_t \in R) = \frac{k}{n}$
Bewijs:

$$P(s_t \in R) = \frac{k}{t} * (1 - \frac{1}{t+1}) * (1 - \frac{1}{t+2}) * ... * (1 - \frac{1}{n})$$
$$P(s_t \in R) = \frac{k}{t} * (\frac{t+1-1}{t+1}) * (\frac{t+2-1}{t+2}) * ... * (\frac{n-1}{n})$$
$$P(s_t \in R) = \frac{k}{\cancel{t}} * (\frac{\cancel{t}}{\cancel{t+1}}) * (\frac{\cancel{t+1}}{\cancel{t+2}}) * ... * (\frac{\cancel{n-1}}{n})$$
$$P(s_t \in R) = \frac{k}{n}$$

Biased reservoir sampling

Aggarwal C. [9] introduceert een bias in reservoir sampling.

Algoritme: Biased reservoir sampling

```
R = ⊥ met capaciteit k
t = 0
while data stream S heeft nog elementen do
    t++
    f = de volheid van R, zijnde een getal tussen [0, 1]
    i = random(0, 1)
    Voeg element S[t] toe aan R
    if i <= f then
        Verwijder een willekeurig element uit R
    end
end
```

In zijn paper geeft Aggarwal aan dat de kans dat een element in R zit exponentieel daalt in functie van t. Dit wil zeggen dat oudere elementen een exponentieel kleinere kans hebben om in R te zitten dan recente elementen. De probabiliteit gaat als volgt, met r het r-de punt in R, t het t-de punt in S en λ een bias ratio die applicatie specifiek is:

$$f(r,t) = e^{-\lambda(t-r)}$$

De bias ratio λ wordt bepaald aan de hand van de grootte van het reservoir (k):

$$k = \frac{1}{\lambda}$$

Biased reservoir sampling	
Voordelen	Nadelen
+ Geheugen efficiënt + Kan een diverse set van data beter leren (data die verder uit elkaar ligt, heeft meer betekenis dan twee observaties vlak na elkaar) + Het achterliggende model kan efficiënter worden omdat er minder data opgeslagen wordt	- Als het concept vaak verandert, heb je het probleem "van alles iets" - Het kan zijn dat het oude concept nog voor een lange tijd in het reservoir zit

Tabel 3: Voor- en nadelen van biased reservoir sampling

2.11.4 Verwachtingen

Er wordt verwacht dat kleinere datastructuren (klein sliding window/reservoir), die net groot genoeg zijn om één concept te omvatten, slechter zullen presteren dan grotere datastructuren. Het is namelijk mogelijk dat relevante informatie te snel uit het geheugen zal verdwijnen. De eerste observaties van het nieuwe concept zouden al uit het geheugen verwijderd kunnen zijn voordat het concept geleerd is.

In het algemeen denk ik dat sampling beter zal presteren dan een sliding window, aangezien observaties langer in het geheugen blijven. Ik denk dat dit het "vergeten" niet gaat tegenwerken, omdat het oude observaties geleidelijk aan verwijderd. Als een sliding window en sampling dezelfde grootte hebben, dan zal data van update u_t na k updates helemaal uit het geheugen verwijderd zijn. Dit is niet het geval bij sampling. Data kan langer dan k updates in het geheugen blijven, maar steeds in een kleinere hoeveelheid.

Hoofdstuk 3

Resultaten

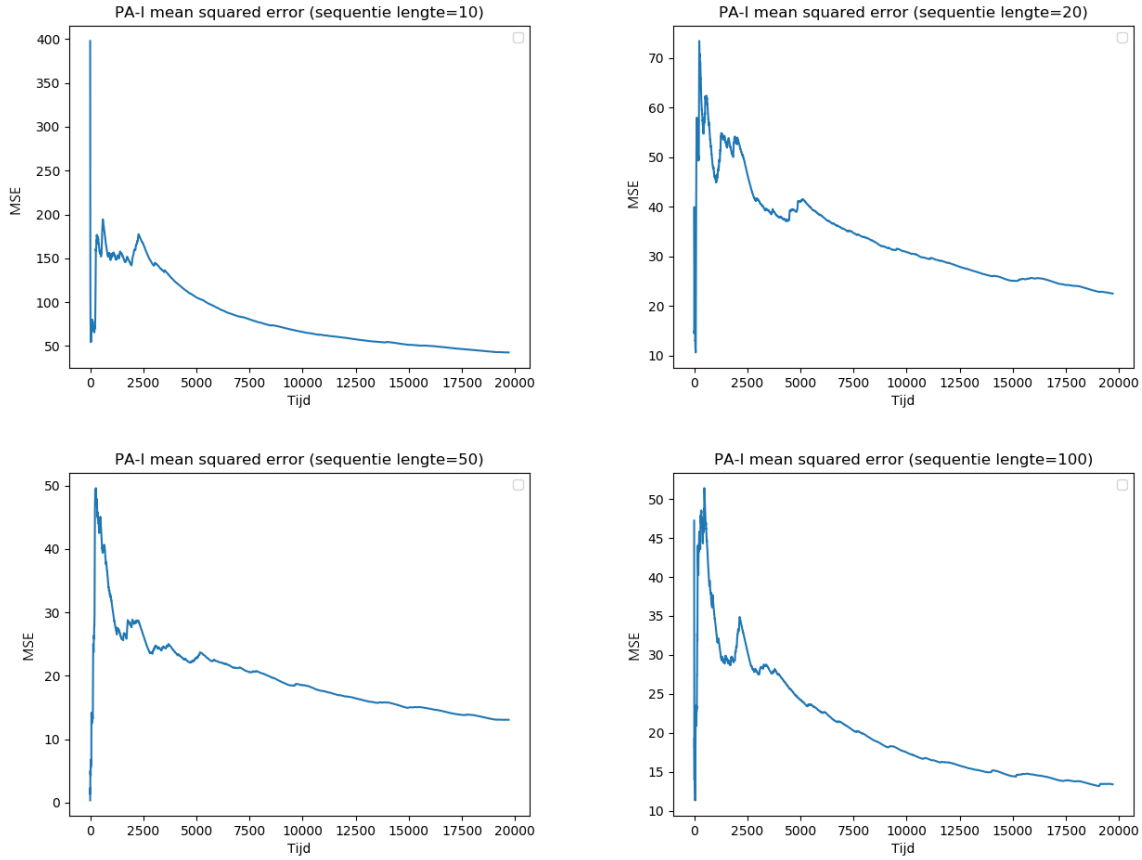
De algoritmes worden getest in een online situatie gegenereerd door de simulatie. Er is enkel ruis toegevoegd op het koppel geleverd door de fietser. De modellen beginnen van nul (niet vooraf getraind). Gedurende een startperiode leren de algoritmes bij en zal de cadans bepaald worden door het fietsersmodel. De algoritmes nemen deze instelling over na de startperiode. Na elke voorspelling wordt de Mean Squared Error (MSE) berekend tussen de voorspelling en het fietsersmodel. Elke 30 iteraties wordt er afgewogen of de algoritmes te ver afwijken van het fietsersmodel. Wanneer de absolute fout de grens van vijf rpm overschrijdt, zal er geleerd worden. De data die gebruikt wordt om bij te leren, bestaat uit de toestand van de fiets van de afgelopen 100 iteraties (10s). Deze data wordt verwerkt tot een set van 50 training instanties, zijnde data van iteratie 0-49,1-50,..., 50-99. Deze set wordt toegevoegd aan de trainingsset die gebruikt wordt door de algoritmes. Met deze evaluatiemethode wordt er nagegaan hoe snel en hoe vaak het algoritme bijleert. Alle algoritmes worden geëvalueerd in exact dezelfde omstandigheden.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

3.1 Sequentie preprocessing

De lengte van de sequenties heeft een invloed op de resultaten. Zoals te zien op figuur 15 heeft een te kleine sequentie negatieve invloeden op de resultaten van PA. Hoe groter de lengte van de sequentie, hoe accurater de voorspellingen worden. De tijd die het algoritme nodig heeft om de testen te voltooien stijgt ook naargelang de grootte van de sequenties. De error en uitvoeringstijd bij DT en RF ondervinden een kleine impact bij het variëren van de lengte van de sequenties.

Om goede resultaten te krijgen, zetten we de lengte van de sequenties boven de 20. Wat de beste optie is, is moeilijk te zeggen. Een kleine sequentie omvat maar enkele omwentelingen van de pedalen. Als er zich hier een grote inconsistentie voordoet, kan dit slechte resultaten opleveren. Daarom zullen alle tests vanaf dit punt een constant lengte van 50 hebben.



Figuur 15: De invloed van sequentielengte op de error

3.2 Algoritmes

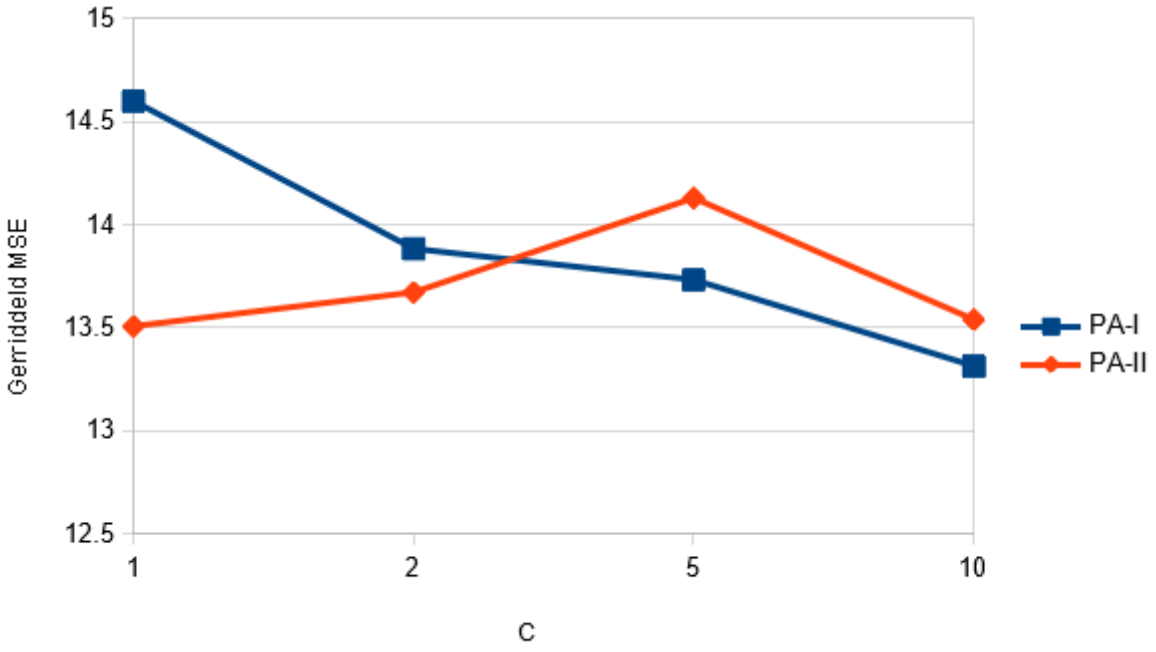
3.2.1 Passive Aggressive Algorithm

Er zijn enkele hyperparameters die ingesteld kunnen worden. Max_iter is het maximum aantal iteraties dat het algoritme probeert bij te leren. Tol is een parameter die bepaalt of het algoritme vroegtijdig stopt. Dit gebeurt wanneer de fout na een leercyclus met minder dan tol verbetert. In de testomgeving, met $max\_iter = 25$ en $tol = 0.1$, wordt deze vroegtijdige stop altijd behaald. Met andere woorden, na 25 iteraties heeft het algoritme de trainingsdata geleerd.

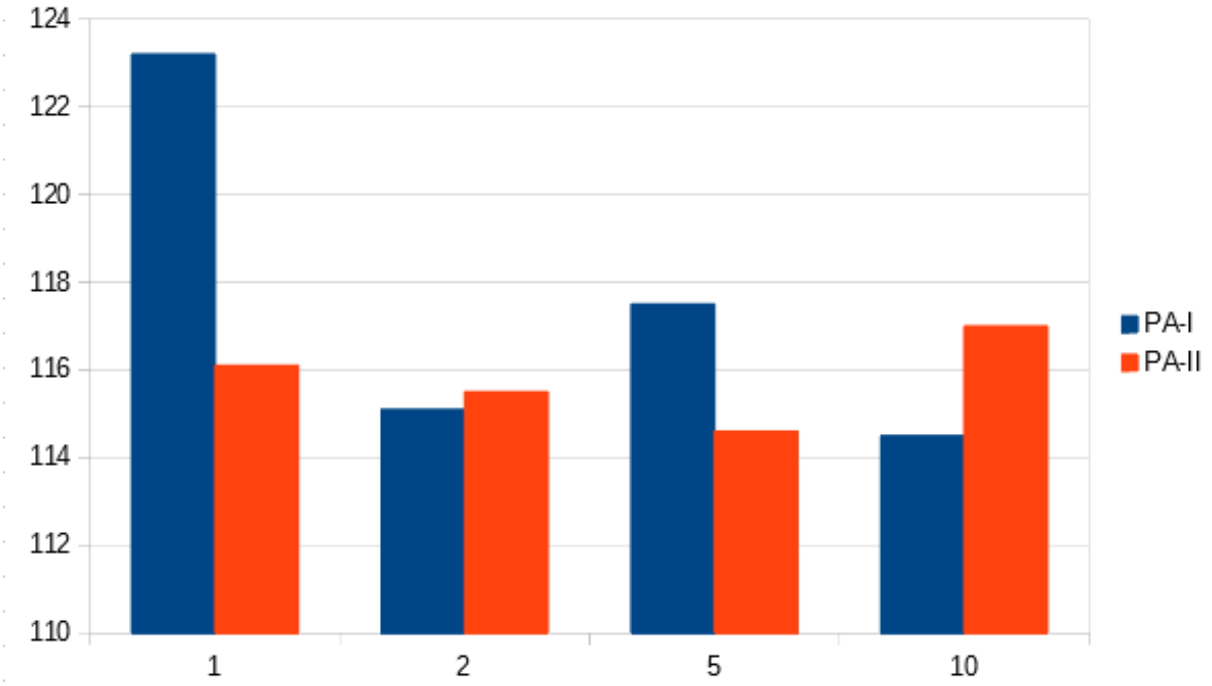
Een laatste interessante parameter is C: de agressiviteit parameter. Hoe hoger deze is, hoe agressiever het algoritme de gewichten gaat bijwerken. De onderstaande figuren tonen de MSE van PA-I (links) en PA-II (rechts). C is de enige parameter die aangepast wordt.

Bijna alle tests convergeren naar een MSE tussen tien en vijftien, met een gemiddelde tussen dertien en veertien. Wat het verschil is tussen PA-I en PA-II valt niet duidelijk te onderscheiden. In de paper van Crammer et al. [5] worden beide versies vergeleken op basis van instance noise en label noise. In beide gevallen scoren PA-I en PA-II aanzienlijk beter dan het standaardalgoritme. In dit experiment scoren PA-I en PA-II gelijkaardig.

Figuur 17 toont de invloed van C op het aantal keer trainen. Het verschil tussen de verschillende settings is klein. De "stappen" die genomen worden tijdens het trainen, zullen dus vaak klein genoeg zijn zodat C=1 agressief genoeg is. Het is dus niet nodig om hoge C-waarden (5-10) te gebruiken. Deze data is genomen over tien sessies, van elk 20000 iteraties, en duurde telkens gemiddeld 90 seconden.



Figuur 16: De invloed van C op MSE van PA



Figuur 17: Gemiddeld aantal keer trainen PA

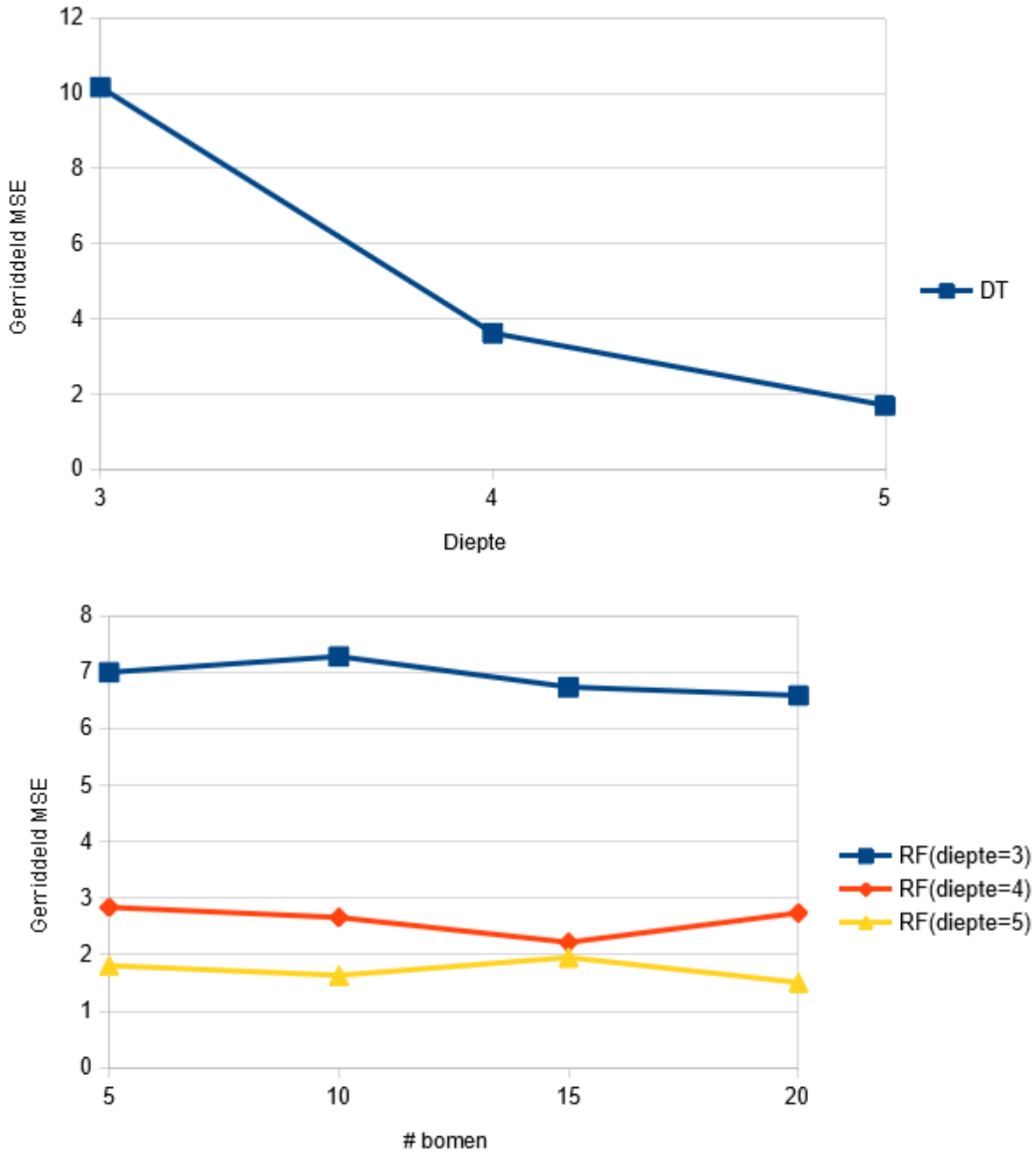
3.2.2 Decision Tree en Random Forest

De belangrijkste parameter bij deze rule-based learners is de *max_depth*. Dit is een moeilijk in te schatten parameter, vooral voor de DT. Diepere DT's maken betere voorspellingen, maar op een bepaald punt begint de DT te overfitten. Bij RF kan ook het aantal bomen ingesteld worden. Hoe meer bomen er gebruikt worden, hoe minder invloed ruis heeft op voorspellingen en hoe beter de voorspellingen worden.

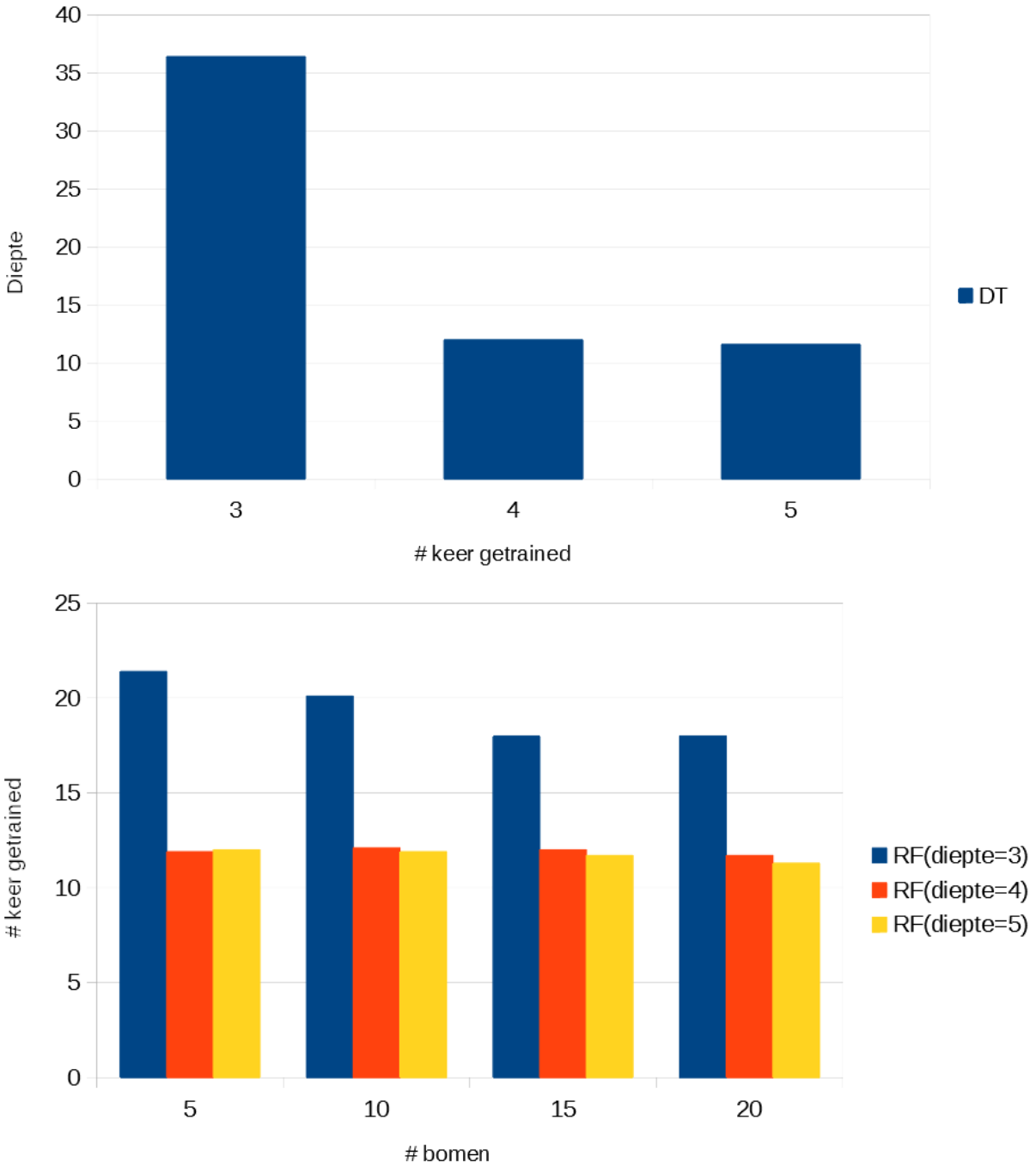
DT's en RF's convergeren beide naar ongeveer dezelfde MSE. Diepere bomen leiden evident naar een lagere MSE. Algemeen zijn grotere RF's beter, maar in deze situatie is het verschil klein.

Voor DT, daalt het gemiddeld aantal trainingen spectaculair tussen diepte drie en vier (figuur 19). Een DT van diepte drie zal dus geen goede oplossing zijn. RF's daarentegen presteren wel goed met een diepte van drie. Een DT/RF dieper dan vier is niet nodig, aangezien dit geen groot voordeel oplevert en mogelijk overfit.

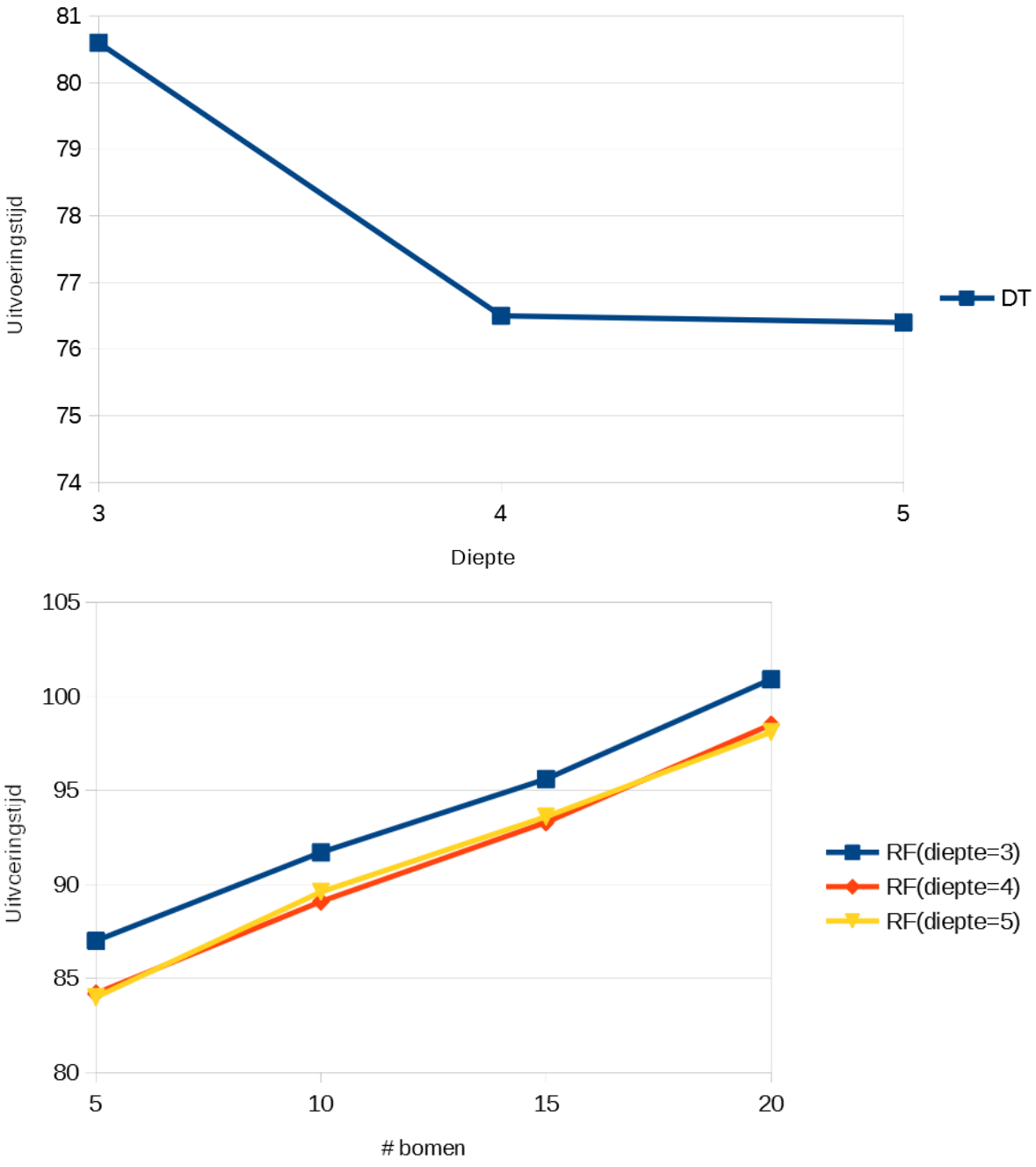
De uitvoeringstijd (figuur 20) lijkt geen probleem te vormen voor grotere RF. Het verschil tussen de uitvoeringstijden van een RF met diepte drie en de anderen is te wijten aan het aantal keer dat getraind moest worden op diepte drie.



Figuur 18: De invloed van diepte en aantal bomen op de gemiddelde MSE van DT en RF



Figuur 19: De invloed van diepte en aantal bomen op het gemiddeld aantal keer trainen van DT en RF



Figuur 20: De invloed van diepte en aantal bomen op de uitvoeringstijd van DT en RF

3.3 Stochastisch bijleren

In de onderstaande tabellen (tabellen 4-6) zijn de resultaten te zien van gelijkaardige tests zoals in deel 2 van dit hoofdstuk. Het enige verschil met de voorgaande test, is de update-strategie voor het updaten van de verschillende modellen.

In vorig hoofdstuk (sectie 2.10.1) werden verwachtingen opgesteld rond deze tests. Er werd verwacht dat de modellen deze probabilistische update-strategie aankunnen. In sommige gevallen is de MSE gelijkaardig aan de resultaten van de tests in deel 2 van dit hoofdstuk (PA-I, PA-II en RF en DT met diepte drie). Alle andere gevallen vertonen een betere MSE dan de originele tests. Dit komt waarschijnlijk door de hoeveelheid trainingsdata die gebruikt wordt. De probabilistische update-strategie zorgt ervoor dat er meer wordt bijgeleerd, waardoor er meer data beschikbaar is. De verwachting dat de modellen stochastisch kunnen bijleren is dus deels correct. Bomen van diepte drie moeten echter veel meer leren (aangezien Δ_c hoger is). Deze keer valt er wel een verschil te zien tussen een klein RF en een groot RF. Een groter RF traint minder vaak.

Bij DT en RF is de stijging van het gemiddeld aantal keer trainen het grootst. PA presteerde al slechter dan de andere modellen waardoor de stijging hier minder extreem is. De algemene stijging is te wijten aan het feit dat er geen tolerantiegrens meer is. Het verschil tussen de voorspelde cadans en het fietsersmodel (Δ_c) kan maar enkele rpm groot zijn, het kan nog steeds een update veroorzaken. Figuur 21 toont hoe Δ_c evolueert doorheen de test voor een RF bestaande uit 20 bomen met diepte vier. Dit model update gemiddeld 59.6 keer. Aan de grafiek te zien zijn deze updates meestal doorgevoerd wanneer het verschil klein is ($\Delta_c < 2$).

Passive Aggressive Algorithm		c			
		1	2	5	10
Gemiddelde MSE	PA-I	14.76 (+1%)	14.63 (+5%)	14.66 (+7%)	14.60 (+10%)
	PA-II	14.52 (+7%)	14.50 (+6%)	14.05 (-1%)	15.56 (+15%)
Gemiddeld aantal keer trainen	PA-I	175.6 (+42%)	170.3 (+48%)	148.8 (+26%)	147.7 (+28%)
	PA-II	170.3 (+46%)	168.1 (+45%)	176.0 (+53%)	179.0 (+54%)

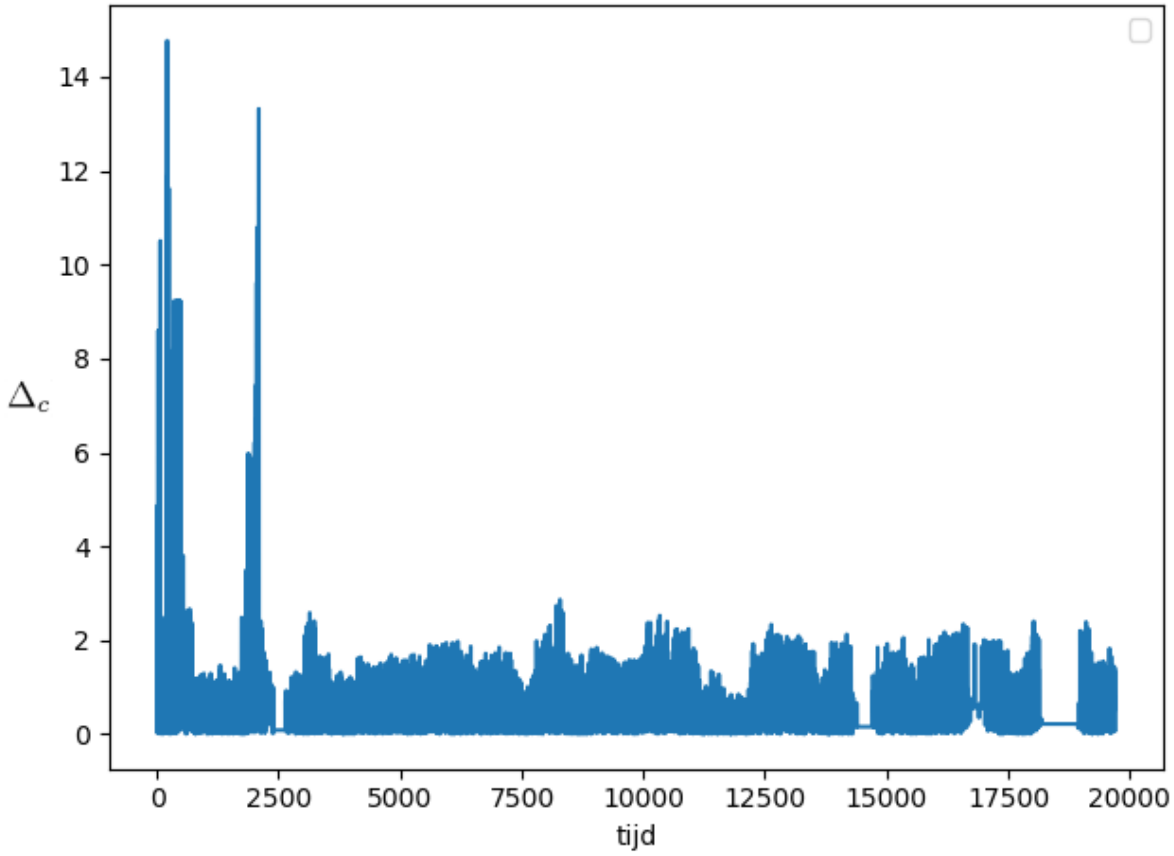
Tabel 4: Resultaten stochastisch bijleren PA

Decision tree	Diepte		
	3	4	5
Gemiddelde MSE	9.37 (-8%)	2.41 (-33%)	1.06 (-37%)
Gemiddeld aantal keer trainen	160.8 (+341%)	88.9 (+640%)	51.3 (+342%)

Tabel 5: Resultaten stochastisch bijleren decision tree

Random forest		Aantal bomen			
		5	10	15	20
Gemiddelde MSE	Diepte 3	7.35 (+5%)	7.42 (+2%)	7.33 (+9%)	7.37 (+12%)
	Diepte 4	1.47 (-48%)	1.38 (-38%)	1.37 (-38%)	1.35 (-50%)
	Diepte 5	0.56 (-69%)	0.59 (-64%)	0.53 (-73%)	0.56 (-63%)
Gemiddeld aantal keer trainen	Diepte 3	145.3 (+578%)	143.8 (+615%)	145.4 (+707%)	144.7 (+704%)
	Diepte 4	72.8 (+511%)	58.2 (+380%)	61.6 (+413%)	59.6 (+409%)
	Diepte 5	40.4 (+236%)	31.8 (+167%)	33.4 (+185%)	30.4 (+169%)

Tabel 6: Resultaten stochastisch bijleren random forest

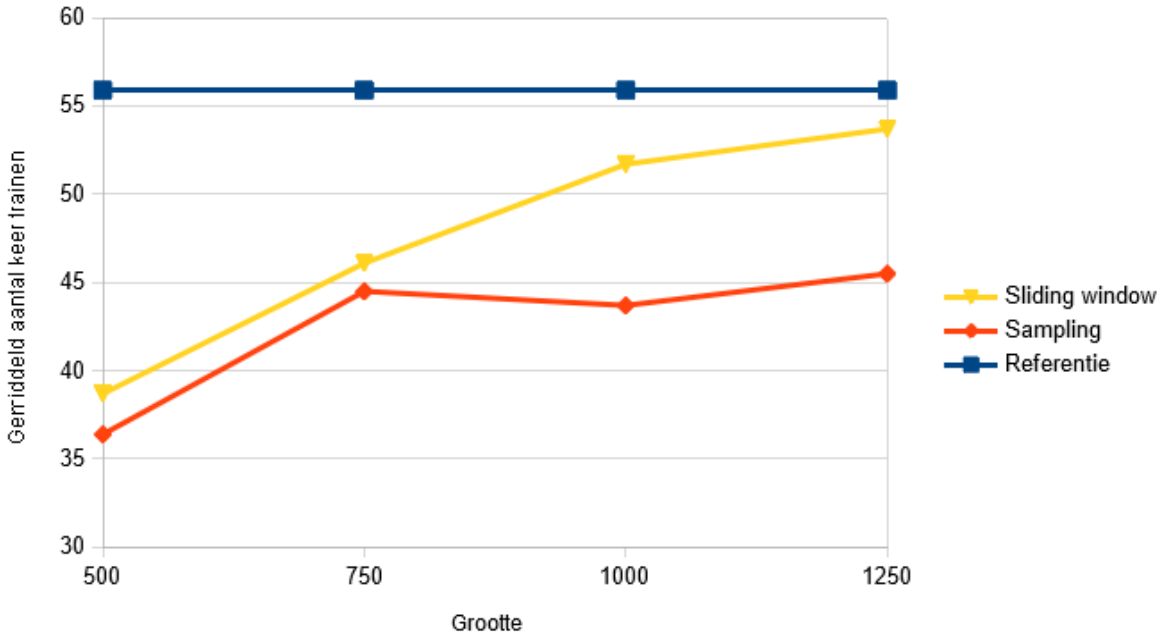


Figuur 21: Het absoluut verschil tussen het fietsersmodel en de voorspellingen (Δ_c) in verloop van tijd

3.4 Conceptuele drift

Wederom wordt dezelfde test als in deel 2 van dit hoofdstuk gebruikt als basis om resultaten te verkrijgen van de twee algoritmes die omgaan met conceptuele drift. Het model voor deze test is een RF bestaande uit 20 bomen met diepte vier. Eerst wordt het RF getraind op basis van een fietsersmodel in functie van het DC-koppel (gemiddeld koppel) voor 20000 iteraties. Vervolgens "stopt" de fiets en wordt het fietsersmodel vervangen door een ander model, dit keer in functie van de helling. In werkelijkheid wordt een nieuw fietsobject geïnstantieerd en maakt dit gebruik van de bestaande cadans controller. Deze nieuwe actor fietst dan voor 40000 iteraties. De tweede fietstocht is langer genomen om na te gaan of de datastructuren wel groot genoeg zijn, zodat het hele concept in deze structuur past. Er worden vier verschillende groottes (500, 750, 1000 en 1250) met elkaar vergeleken die respectievelijk overeenkomen met data voor 10, 15, 20 en 25 trainingen. Deze worden dan vergeleken met een referentietest, een uitvoering zonder drift technieken.

In figuur 22 is te zien dat wanneer er minder data wordt bijgehouden, er in totaal minder moet geleerd worden. Dit verschilt met de verwachting, die opgesteld is in sectie 2.11.4, dat kleinere structuren slechter zouden presteren dan grotere structuren. Waarschijnlijk is het betere resultaat juist te wijten aan de kleine structuur, waardoor er snel vergeten wordt. In grotere structuren blijft data langer rondhangen. Het oude concept vult niet de hele structuur. Hierdoor zal er meerdere keren geüpdatet moeten worden vooraleer het nieuwe concept geleerd is, net zoals wanneer er geen drift mechanisme is. Uiteindelijk is het oude concept vergeten, maar hiervoor was er veel meer data nodig. Bijvoorbeeld, bij het veranderen van concept hebben we twee of drie keer zoveel data nodig van situatie A, die ook voorkomt tijdens de vorige fietstocht (situatie is hier de algemene toestand van de fiets). Vervolgens komt situatie B die weeral twee of drie keer zoveel data nodig heeft, omdat het sliding window nog niet genoeg vergeten heeft. Uiteindelijk hebben we zoveel data toegevoegd van het nieuwe concept dat het oude concept helemaal vergeten is. Maar omdat er zoveel meer data nodig was, zijn nog niet alle situaties bekeken. Als de fiets dan in een ongeziene situatie komt, kan data van situatie A al vergeten worden. Stel dat situatie A uiteindelijk terugkomt, dan moet dit opnieuw geleerd worden. Deze keer minder aangezien er geen conflict is met een ouder concept. Dit probleem is zichtbaarder in een sliding window dan bij sampling, aangezien sampling "sneller" data vergeet. Elke keer er een observatie toegevoegd wordt, is er een kans dat er één verwijderd wordt. Waardoor elke situatie in mindere mate aanwezig is, maar toch nog genoeg om een goede voorspelling te leveren. Merk op dat voor zowel sliding window als sampling bij grootte 500 er minstens 1750 observaties bekeken zijn om met conceptuele drift om te gaan.



Figuur 22: Resultaten van beide technieken om met conceptuele drift om te gaan.

Hoofdstuk 4

Discussie

4.1 Algoritmes

Het vorige hoofdstuk toont de resultaten van de verschillende algoritmes. RF en DT presteren duidelijk beter dan beide types van PA. DT presteert goed, maar is nog steeds bekend om het slecht omgaan met ruis en het overfitten. Aangezien het verschil in uitvoeringstijd tussen DT en RF minimaal is, is het altijd beter om een RF te gebruiken. Alhoewel in de test blijkt dat de grootte van het RF geen al te grote impact heeft op de uitvoeringstijd, kan dit mogelijk nog een probleem vormen door de beperkte rekenkracht van de Raspberry Pi. Het controleprogramma voor de fiets zal immers parallel lopen met de cadanscontroller. Gelukkig heeft de Raspberry Pi vier cores en is een RF makkelijk te parallelliseren (door het instellen van het aantal parallelle taken). Het maximaal aantal taken zal dus vier zijn, maar hoogstwaarschijnlijk zal dit lager zijn zodat het controleprogramma er niet onder lijdt.

Voor een RF zijn volgende hyperparameters aangeraden:

RF diepte=4 of 5, aantal bomen=10-20

Een groot nadeel van RF is dat het niet goed om kan met ongeziene omstandigheden. Bijvoorbeeld als het algoritme van niets of weinig data begint, kan het niet uit de reeds geziene data een goede voorspelling maken (t.o.v. lineaire regressie voor een simpel lineair probleem). Alvorens dit algoritme op de fiets getest wordt, zou er al een standaard model ingesteld moeten worden dat dan aangepast wordt door de fietser. Een andere, betere manier wordt hier voorgesteld. In plaats van de exacte cadans te voorspellen van de fietser, kan er een verschil voorspeld worden met een standaard cadans, bijvoorbeeld 70 rpm. Wanneer het model nog niets geleerd heeft, zal de cadanscontroller altijd een cadans voorspellen van 70 rpm. Als de fietser de cadans aanpast, dan moet het algoritme het verschil voorspellen tussen de standaard cadans en de gewenste cadans. Het model zal dus bijvoorbeeld een zeven kunnen voorspellen waardoor de aangeboden cadans 70+7=77 is.

De huidige implementatie van een RF, geïmplementeerd in de scikit-learn bibliotheek, doet niet aan lokale regressie in de bladeren. In plaats daarvan bevatten de bladeren een gewoon getal. Mogelijk kan lokale regressie in de bladeren een betere prestatie leveren.

4.2 Waarheid

In de verschillende tests werd het fietsersmodel altijd als waarheid (*ground truth*) gezien. Dit kan echter niet gebruikt worden, aangezien dit ongekend is. Als de fietser een update wil doen, dan weet het algoritme niet hoe hard het moet aanpassen. Als de fietser sneller wil trappen, is het dan twee rpm sneller of vijf? Dit is in deze thesis niet onderzocht, maar vormt nog een potentieel probleem in het succes van de real-time cadansaanpassing in een automatische fiets transmissie.

4.3 Conceptuele drift

In sectie 3.4 werden de resultaten van twee algoritmes, sliding window en biased reservoir sampling, getoond. In beide gevallen moest er minstens drie keer zoveel geleerd worden om het nieuwe concept te leren, wat toch wel een groot verschil is met het beginnen vanaf nul. Deze technieken doen het alleszins wel beter dan technieken zonder "vergeet" algoritme. Het voornaamste probleem hier is dat mensen die de fiets tweedehands kopen of als leen-, familiefiets gebruiken, een slechtere fietservaring zullen hebben dan mensen die de fiets nieuw kopen en alleen zelf gebruiken. Als de fiets als familiefiets wordt gebruikt, is het misschien nuttig om een conceptueel drift mechanisme te implementeren, zoals beschreven in 2.11.2 (tabel 2), dat gebruik maakt van een change detector en kan wisselen tussen getrainde modellen. Een familiefiets zal slechts gebruikt worden door enkele mensen. Een leenfiets daarentegen bereikt een veel groter publiek. Hier kan een reset functionaliteit nuttig zijn.

Hoofdstuk 5

Conclusie

Eerst werd er een realistische geparametriseerde simulatie opgezet. Die houdt rekening met verschillende lasten, maar niet allemaal. De simulatie zorgt ervoor dat er gemakkelijk en snel data kan gegenereerd worden voor verschillende tests. Ten tweede werden enkele algoritmes besproken en geëvalueerd. De resultaten tonen dat PA niet geschikt is voor dit probleem, aangezien de fout te groot is en dat PA te vaak moet bijleren. DT en RF presteerden hier goed. Ten derde werd nagegaan of de algoritmes kunnen omgaan met een stochastische update-strategie. Wederom presteert PA slecht. Binaire beslissingsbomen, zowel bij DT als RF, met diepte drie presteren hier ook slecht omdat het verschil tussen fietsersmodel en voorspelling vrij groot blijft gedurende de test. Uit deze twee tests werd er besloten dat enkel DT en RF, met diepte vier of dieper, geschikt zijn voor dit probleem. Uiteindelijk werd de keuze gemaakt om enkel verder te werken met een RF. Er werd wel nog een probleem aangehaald met een RF: het kan niet goed om met ongeziene omstandigheden, zoals wanneer de fiets nog niet is ingesteld door de gebruiker. Er wordt hiervoor volgende oplossingen voorgesteld:

- een standaard model voorzien
- hetgeen wat het RF voorspelt veranderen van de eigenlijke cadans naar een verschil ten opzichte van een ingestelde basis cadans.

Vervolgens werden twee technieken getest die kunnen omgaan met conceptuele drift, het veranderen van concept over tijd. De drift die hier getest werd was abrupt in deze context. De resultaten tonen aan dat kleinere structuren beter presteren dan grotere. Sampling presteert in het algemeen beter dan een sliding window. Of omgaan met conceptuele drift noodzakelijk is, hangt af van hoe men de fiets gebruikt. Er werd ten slotte nog een probleem aangehaald met de tests en trainingsdata. In de simulatie werd een "waarheid" (*ground truth*) berekend met behulp van het fietsersmodel. In realiteit bestaat dit niet en zullen updates incrementeel moeten doorgevoerd worden. Dit moet zeker nog uitgewerkt worden. Uiteindelijk moet er ook nog een implementatie voorzien worden die samenwerkt met de fietscontroller van Ellio.

Of een RF het meest geschikte algoritme is, is moeilijk te zeggen. Er zijn zoveel verschillende algoritmes van machinaal leren dat het praktisch onmogelijk is om ze allemaal te bekijken. Een type algoritme dat hier niet aangehaald is, maar mogelijk wel nog goed kan presteren in clustering.

De cadanscontroller moet natuurlijk ook nog uitgewerkt worden, zodat het samenwerkt met de fietscontroller.

Appendix

IntuEdrive

Ir. Tomas Keppens werkte als departementshoofd bij Toyota toen hij in 2010 met het IntuEdrive project begon. In het kader van een aantal masterproeven aan de KU Leuven werd het project stap per stap uitgewerkt.. In het academiejaar 2016-2017 werkte ingenieursstudent Jorrit Heidbuchel de aansturing van het intuEdrive CVT systeem uit. Halverwege 2017 was er het eerste prototype. Later dat jaar richtten Tomas Keppens en Jorrit Heidbuchel samen intuEdrive op.

IntuEdrive wil mobiliteit duurzamer en efficiënter maken door twee-wielmobiliteit veiliger te maken. Het bedrijf bouwt en ontwikkelt E-bikes die perfect passen in het leven van zijn klanten: makkelijk in gebruik, veilig en betrouwbaar.
Vandaag telt IntuEdrive 4 werknemers. Ellio gaat in het najaar van 2019 voor het eerst in productie en mikt in 2020 op een verkoop van 400 stuks in België.

Figuur 23: Logo IntuEdrive

Bibliography

[1] Heidbuchel, J. (2017). *Hardware Implementation and Control Strategy of a High Dynamic CVT Transmission for an E-Bike*. Katholieke Universiteit Leuven, Departement werktuigkunde.

[2] Breiman, L. (2001). *Random Forests*. University of California, Berkely.

[3] London I. (2016). *Encoding cyclical continuous features - 24-hour time*. Geraadpleegd op 8 mei, 2019 via https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/

[4] Sarle, W. *comp.ai.neural-nets FAQ*. Geraadpleegd op 8 mei, 2019 via http://www.faqs.org/faqs/ai-faq/neural-nets/part1/preamble.html

[5] Crammer, Dekel, Keshet, Shalev-Shwartyz, Singer (2006). *Online Passive-Aggressive Algorithms*. Hebrew University of Jerusalem, Jerusalem.

[6] Loeffel, P. *Adaptive machine learning algorithms for data streams subject to concept drifts*. Université Pierre et Marie Curie, Paris VI

[7] Hansen E. en Smith G. (2009). Factors Affecting Cadence Choice During Submaximal Cycling and Cadence Influence on Performance. *International Journal of Sports Physiology and Performance,4 (1),* pp. 3-17.

[8] Vitter J. *Random Sampling with a Reservoir*. Brown University, Rhode Island

[9] Aggarwal, C. (2006). On Biased Reservoir Sampling in the Presence of Stream Evolution. *VLDB, 32 (1),* pp. 607-618.

[10] Bijdragers van Wikipedia. *Continuously variable transmission*. Geraadpleegd op 8 mei, 2019 via https://en.wikipedia.org/wiki/Continuously_ variable_ transmission

[11] Bijdragers van Wikipedia. *Vermogen (natuurkunde)*. Geraadpleegd op 8 mei, 2019 via https://nl.wikipedia.org/wiki/Vermogen_%28natuurkunde%29

[12] Bijdragers van Wikipedia. *Random forest*. Geraadpleegd op 8 mei, 2019 via https://en.wikipedia.org/wiki/Random_ forest

[13] Bijdragers van Wikipedia. *Binomial distribution*. Geraadpleegd op 8 mei, 2019 via https://en.wikipedia.org/wiki/Binomial_ distribution

[14] Bijdragers van Wikipedia. *Geometric distribution*. Geraadpleegd op 8 mei, 2019 via https://en.wikipedia.org/wiki/Geometric_ distribution

Computerwetenschappen
Celestijnenlaan 200 A bus 2402
3000 LEUVEN, BELGIË
tel. + 32 16 32 77 00
fax + 32 16 32 79 96
www.kuleuven.be