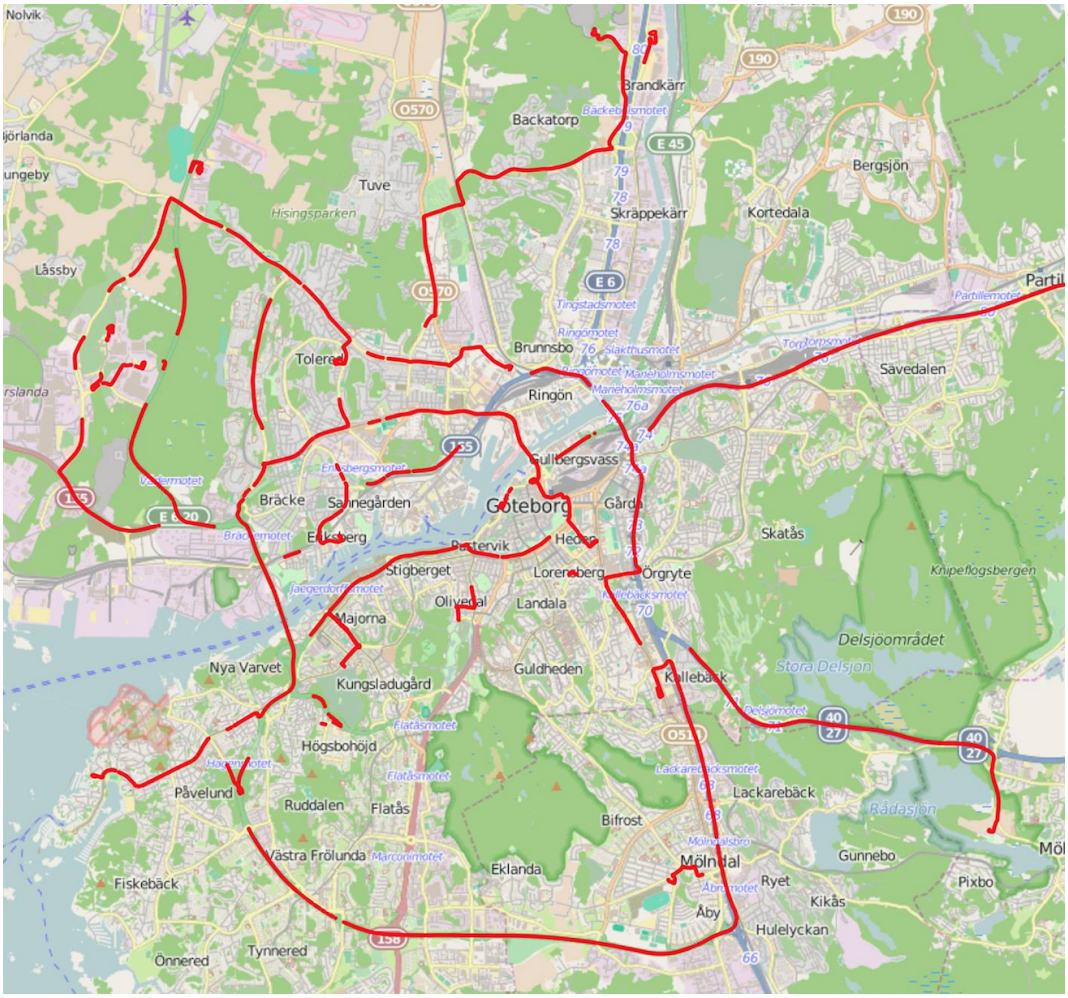
CHALMERS
UNIVERSITY OF TECHNOLOGY



Klassificering och prediktering av bilfärder

SSYX02-15-32

Viann Karlsson
Niklas Liljegren
Joakim Pålsson
Hampus Ramström

Institutionen för Signaler och system
Chalmers tekniska högskola
Göteborg, Sverige 2015

Slutrapport kandidatarbete 2015:05

Klassificering och prediktering av bilfärder

SSYX02-15-32

Viann Karlsson
Niklas Liljegren
Joakim Pålsson
Hampus Ramström

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Signaler och system
Avdelningen för Signalbehandling och medicinsk teknik
SSYX02-15-32
Chalmers tekniska högskola
Göteborg, Sverige 2015

Klassificering och prediktering av bilfärder
SSYX02-15-32
Viann Karlsson
Niklas Liljegren
Joakim Pålsson
Hampus Ramström

Handledare: Lars Hammarstrand, Signaler och system
Examinator: Lennart Svensson, Signaler och system

Slutrapport kandidatarbete 2015:05
Institutionen för Signaler och system
Avdelningen för Signalbehandling och medicinsk teknik
SSYX02-15-32
Chalmers tekniska högskola
412 96 Göteborg
Telefon +46 31 772 1000

Framsida: Datamodellen utritad på en karta över Göteborg med hjälp av MATLAB, skapad utifrån GPS-data erhållen från Volvo Personvagnar. ©OpenStreetMap Contributors

Typeset in LaTeX
Göteborg, Sverige 2015

Förord

Denna rapport med tillhörande plattform byggd i MATLAB utgör kandidatarbetet 'Klassificering och prediktering av bilfärder' som är utfört våren 2015. Arbetet är genomfört på institutionen för Signaler och system av fyra civilingenjörsstudenter från programmen Automation och Mekatronik, Datateknik och Teknisk matematik.

Vi vill först och främst tacka vår handledare Lars Hammarstrand för ett mycket gott samarbete och många goda råd under arbetets gång.

Vi vill även tacka herrarna Alvarez-Garcia, Ottega, Gonzalez-Abril och Velasco för dess tillåtelse i användandet av bilder och resultat från deras artikel *Trip destination prediction based on past GPS log using a Hidden Markov Model (2010)* i vår rapport.

Till sist vill vi även tacka Volvo Personvagnar som bistod oss med rutter för att bygga upp och testa vår modell.

Classification and Prediction of Route

Viann Karlsson, Niklas Liljegren Joakim Pålsson & Hampus Ramström
Department of Signals and systems
Chalmers University of Technology

Abstract

The environment are one of mankinds most urgent subjects to tackle, as it will affect us all. One of the big villains are the emissions from the regular traffic. A contribution to lower these emissions would be the possibility to be able to avoid and bypass traffic jams and analogous time-consuming difficulties. An aid to a solution for this problem would be a device that predicts a drivers intended route destination.

The purpose of this project is to create a platform, utilizing the program MATLAB, that is able to predict a drivers trip destination. With GPS-data, that is representing routes collected by Volvo Car, a model has been built that the platform applies to be able to predict based on earlier travelled routes. A platform like this one would be a desired contribution for the traffic industry in their quest to decrease emissions.

This report describes how data is filtered, classified and grouped. It also describes how a data model and a mathematical model is built. The mathematical model is rooted in a Hidden Markov Model, but with some minor changes in the application.

Complete result has unfortunetly not been yielded, as the developed platform isn't able to predict the last 25 % of an route. The result based on the first 75 % on the other hand holds the same standard as prior studies within this area.

This report is written in Swedish.

Keywords: Predictive HMM, Machine Learning, HMM, GPS

Klassificering och prediktering av bilfärder
Viann Karlsson, Niklas Liljegren, Joakim Pålsson & Hampus Ramström
Institutionen för Signaler och system
Chalmers tekniska högskola

Sammandrag

Miljön är idag svårt ansatt av alla de utsläpp som sker på jorden, där en av huvudbovarna är den trafik som sker dagligen. En av de stora bidragande faktorerna till detta problem är alla de trafikköer som uppstår där fordon står stilla och avger stora mängder avgaser. En bidragande lösning till att dämpa utsläppen och att tackla detta problem skulle vara utvecklandet av en prediktionsplattform.

Syftet med detta arbete är därmed att skapa en plattform i MATLAB som med given GPS-data erhållen från Volvo Personvagnar predikterar ett fordons slutdestination i ett så tidigt skede som möjligt. Den information som en prediktion ger skulle innebära ännu ett bidrag till att minska utsläppen samt minimera tid och bränsleåtgång i trafiken.

Denna rapport kommer att behandla filtrering av erhållen data, klassificering och gruppering av rutter samt uppbyggnad av en datamodell och en matematisk modell. Modellen som tas fram i detta projekt grundar sig i en Dold markovmodell, men har modifierats för att passa den plattform som utvecklats.

Fullständigt resultat från plattformen har olyckligtvis ej erhållits där målet var att jämföra med resultat från tidigare studier inom samma område. Av i skrivande stund okänd anledning avbryts prediktering när cirka 25 % återstår av den rutt som färdas där felet troligen har med modellen som skapats i MATLAB. Resultatet som ges efter att 75 % av rutten färdats är väldigt lovande och uppfyller samma grad, om inte bättre, prediktion av det korrekta slutmålet som tidigare studier inom samma område erhållit.

Nyckelord: Prediktion, HMM, Maskininlärning, GPS

Innehåll

Figurer

Tabeller

1
Inledning

1.1 Bakgrund

Uppdagandet av den klimatuppvärmning som väntar, samt den insyn forskare ger allmänheten, har gjort miljö till ett ämne som aktualiseras dagligen. Konsekvenserna är många och förändringarna stora om inte människan förändrar sitt beteende och antar de utmaningarna som väntar. En huvudbov i sammanhanget är fordonstrafiken som förutspås växa med den ökade fordonsanvändning som väntar i och med utvecklingsländernas förväntade uppgång. Problemet med utsläppen från fordonstrafiken och dess påverkan på miljön är komplext och behöver angripas från flera håll.

Ett bidrag till en lösning för detta problem är möjligheten att kunna förutse ett fordons färdväg i ett så tidigt skede som möjligt. I dagens trafik är vanligt med trafikköer, där fordon får stå stilla länge och pumpa ut avgaser till både förarnas och miljöns förtret. Trots att utsläppen från vägtrafik har minskat totalt sett de senare åren, där trafiken ökade med 2 procentenheter år 2014, skulle minskningen kunnat vara större enligt Trafikverket (2014). Anledningen till den minskning som har skett är tack vare mer energieffektiva fordon samt att användandet av biobränslen har ökat (Naturvårdsverket, 2015).

En utveckling av en plattform för prediktering av bilfärder, kombinerad med tidigare framsteg inom bilindustrin skulle kunna innebära ännu ett nyttigt bidrag för att minska utsläppen ytterligare. Ett exempel på ett område där plattformen skulle kunna appliceras inom är hybridmotorer. Optimering av hybridmoroter hade kunnat utföras om en hybridmotor i tidigt skede hade fått information gällande destination och trolig färdväg. Detta genom att fördelningen av el och bensin hade kunnat effektiviseras, vilket hade medfört besparingar både för föraren och miljön.

Smartphones är ett annat område som är väl lämpat för användandet av en prediktionsplattform. Redan nu existerar applikationer som med hjälp av den inbyggda GPS:en förutspår den optimala vägen utifrån färddata. Ett exempel är smartphoneapplikationen Waze, där föraren skriver in sin destination i applikationen varav den snabbaste vägen dit presenteras utifrån alla användares färddata. Samtidigt som föraren kör och använder applikationen bidrar den passivt genom att förstärka databasen med dess egna färdhistorik. Applikationen är därmed ett kollektivt sätt att hjälpa varandra att minimera tiden och bränsleåtgången vid körsträckor.

1.2 Mål och Syfte

Målet med kandidatarbetet är att skapa en plattform i MATLAB, som med given indata i form av GPS-punkter och tid ska leverera troliga rutter och slutmål för färden samt sannolikheten för dessa. De troliga rutterna, slutmålen och sannolikheterna för dem ska vara helt baserade på förarens tidigare färdhistorik.

Syftet med plattformen är att skapa en modell som utifrån den givna indatan kan prediktera en bilfärd med största möjliga säkerhet. Plattformen ska kunna användas för vidareutveckling, exempelvis appliceras i en smartphone.

2
Teori

Inom området prediktering av bilfärder har ett flertal tidigare studier gjorts, där olika metoder har använts för att komma fram till en lösning på problemet. I detta kapitel kommer en del av dem presenteras. Från dessa rapporter kommer det beskrivas vilka metoder och teorier som använts och inte använts i detta projekt och varför.

För att få en tydligare överblick kommer även begreppen GNSS och Dold Markovmodell beskrivas mer ingående. Den dolda markovkedjan är en välanvänd metod i dessa sammanhang. Därför anses den vara en naturlig bas även för detta projekt. Exakt hur den fungerar beskrivs i avsnitt 2.3.

2.1 Tillvägagångssätt i tidigare studier

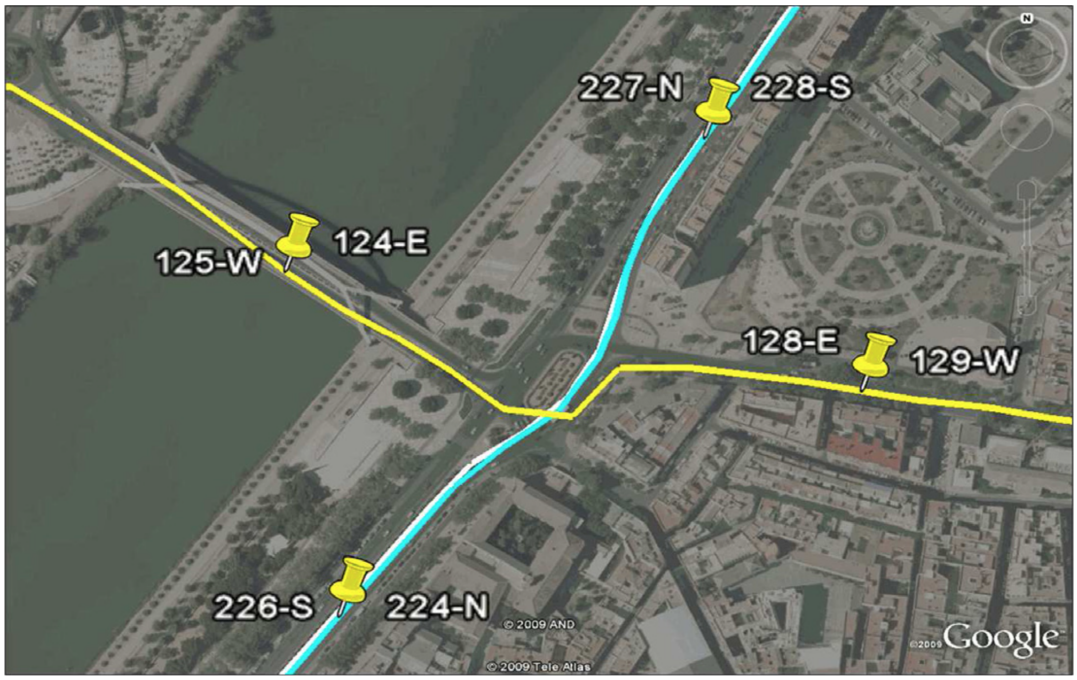
Området gällande att prediktera en bilfärd är sedan tidigare ett välutforskat område, där olika metoder har applicerats för att finna en lösning på problemet. Många av dessa presenteras i samband med den årliga SAE kongressen som hålls i bilens vagga, Detroit, USA, årligen. Det är från en del av dessa rapporter samt andra rapporter som presenterats utanför SAE som detta rapport grundas i.

2.1.1 Hantering av data och segmentering

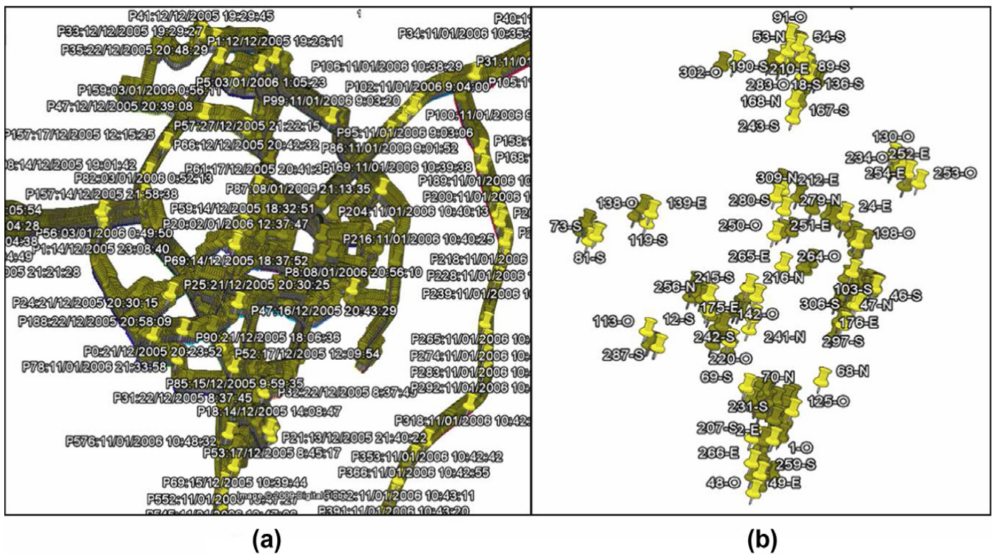
En GPS-mottagare mottar GPS-punkter med ett tätt intervall, vilket bidrar till att en stor mängd data samlas in. För att minska datamängden sätts ett krav på att distansen mellan GPS-punkterna måste vara på ett visst tiotal meter vilket finns beskrivet av Alvarez-Garcia, Ortega, Gonzalez-Abril & Velasco (2010).

För att få den resterande datamängden av GPS-punkterna hanterbar i beskrivandet av rutter finns det två alternativ förklarade i den litteratur som legat till grund för detta projekt. Det allmängiltiga alternativet, vilket är alternativet som appliceras i detta arbete, är användandet av segment. Ett segment är en area kring en vägsnutt som går mellan två delningar av rutter. Ett segment kan också beskrivas som en följd av diskreta GPS-punkter. Det andra är användandet av "support points"(stödpunkter) som finns beskrivet av Alvarez-Garcia et al. (2010). Metoden med stödpunkter går ut på att endast använda punkter i plattformen skapade när två eller fler rutter korsar varandra. Till exempel skapas fyra punkter när två rutter korsar varandra istället för fyra stora segment, det vill säga en vid varje ut-/ingång till korsningen, se *Figur 2.1.* Genom användandet av endast dessa stödpunkter i

plattformen under predikterandet fås en väldigt låg mängd data som kan ses i *Figur 2.2*. Detta gör plattformen mer applicerbar i mobila enheter som till exempel i en applikation till en smartphone.



Figur 2.1: En korsning av två rutter med stödpunkter utsatta (s.4, Alvarez-Garcia et al., 2010). Återgiven med tillstånd.



Figur 2.2: **(a)** Modell uppbyggd av samtliga GPS-punkter efter filtrering. **(b)** Modell uppbyggd av stödpunkter (s.4, Alvarez-Garcia et al., 2010). Återgiven med tillstånd.

2.1.2 Klustring

I Alvarez-Garcia et al. (2010) finns det utöver metoden med stödpunkter användbar information gällande klustring. En start-/slutdestination behöver klustras med en viss radie, då en destination inte enbart är en punkt. Låt oss exempelvis säga att en parkeringsplats är en slutdestination. Eftersom hela parkeringsplatsen ska räknas som en start-/slutdestination i modellen och det finns en viss osäkerhet i GPS-positionen räcker det inte att enbart ha med den parkeringsrutan fordonet är parkerat på. Om programmet istället klustrar med en viss radie utifrån den parkeringsruta fordonet står på blir hela parkeringsplatsen, eller stora delar av den, med som en start-/slutdestionation i modellen.

2.1.3 Filtrering

För filtrering av den erhållna datan finns det sedan tidigare väldokumenterat av Froehlich & Krumm (2008) vad som bör tänkas på. I den ovannämda artikeln reduceras antalet rutter från 27 429stycken till 14 523 stycken. Exempel på den filtrering som berörs är bland annat att en rutt minst måste innehålla ett visst antal GPS-punkter samt att en rutt måste sträcka sig över en viss sträcka. Via den filtreringen tas rutter där föraren bara flyttat bilen lite grand. Till exempel från en parkeringsruta till en annan, eller om föraren bara startat bilen för att efter en stund stänga av den igen. Utöver detta finns även ett smidigt sätt att upptäcka och därefter avlägsna GPS-punkter med orimligt avstånd från resterande punkter i rutten. Genom att stega igenom rutten och ta GPS-punkterna parvis för att undersöka om accelerationen samt hastigheten mellan dessa är på en rimlig nivå syns det enkelt om GPS-punkten bör avlägsnas eller inte.

2.2 GNSS

För att kunna följa en bils färd och veta vart bilen befinner sig används en GNSS-mottagare. Satellitbaserade navigations och postioneringssystem benämns oftast som GNSS, där GPS är det system som detta arbete baseras på. GNSS-mottagaren tar emot kodade radiosignaler från satelliter och jämför dessa signaler med mottagarens identiska signaler. Genom att mäta skillnaderna i signalerna vid samma tidpunkt kommer tiden det tog för signalen att nå GNSS-mottagaren erhållas, och därmed kan avståndet mellan mottagare och sändare beräknas genom formeln $s = v \times t$.

För just GPS som används i detta arbete krävs det tillgång till minst fyra av de 24 GPS-satelliterna som är i drift för att mottagarklockans fel och de tredimensionella koordinaterna i det globala referensssytemet WGS84 ska kunna tas fram av GPS-mottagaren.

Trots att GNSS-mottagaren oftast visar korrekt position finns det givetvis felkällor vid mätning. Felkällorna kan bland annat bero på att GNSS-signalerna inte alltid tar den korrekta sträckan, eftersom att det ibland finns objekt i vägen som reflekterar signalen fel eller att satelliten inte befinner sig på korrekt plats i satellitbanan.

(Lantmäteriet, u.å)

2.3 Markovmodellen

Den statistiska modell som applicerats i de tidigare studierna och är grund för kandidatprojektet är en dold Markovmodell (Hidden Markov Model, HMM). Det är en aktuell modell som används i de senare forskningarna inom ämnet, det vill säga predikterandet av en färddestination.

2.3.1 Markovkedja

En dold Markovmodell grundas i den simplaste av Markovmodellerna, vilket är en Markovkedja. En Markovkedja är en tupel, det vill säga en ändlig objektsekvens bestående av $< \mathbf{S}, \mathbf{A}, \pi >$, där $\mathbf{S}$ är ett ändligt antal tillstånd, $\mathbf{A}$ är ett ändligt antal övergångssannolikheter och π är den stationära fördelningen. Ett tillstånd kan vara allt från en plats, en viss väderlek eller till exempel erhållandet av en viss siffra vid tärnkastning. Där beskriver en handling a_{ij} sannolikheten för hur tillstånden förändras mellan tidssteget t och $t+1$. $A = \{a_{ij}\}$ är därmed en $N \times N$ matris av möjliga övergångar mellan N tillstånd.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix}$$

där

$$a_{ij} = P(\text{tillstånd } s_i \text{ vid } t+1 | \text{tillstånd } s_i \text{ vid } t).$$

Den stationära fördelningen π innehåller de initiala sannolikheterna för i vilket tillstånd som är aktivt, det vill säga

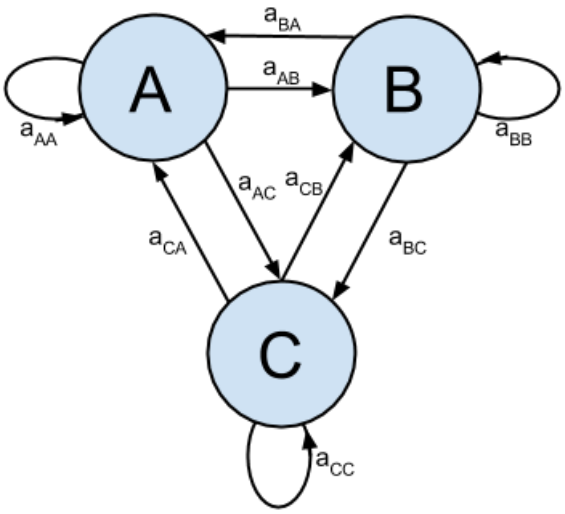
$$\pi = [p(\text{tillstånd } s_1 \text{ vid } t = 0), \dots, p(\text{tillstånd } s_1 \text{ vid } t = 0)].$$

I *Figur 2.3* kan ett allmänt fall av en Markovkedja beskådas, där $\mathbf{S} = [A, B, C]$ är tillstånd och

$$\mathbf{A} = \begin{bmatrix} a_{AA} & a_{AB} & a_{AC} \\ a_{BA} & a_{BB} & a_{BC} \\ a_{CA} & a_{CB} & a_{CC} \end{bmatrix} \tag{2.1}$$

är de möjliga övergångarna.

För att konkretisera det hela följer nu ett exempel av en Markovkedja. Låt oss säga att sannolikheten $\mathbf{x}^{(t)}$ ska beräknas för en viss väderlek dag t, beroende av gårdagens väderlek. Väderleken är representerad som tillstånd där det i detta exempel finns tre tillstånd, vilka är $\mathbf{S} = [Sol, Regn, Molnigt]$. Vädret kan skifta mellan dessa och utifrån statistik har sannolikheterna för övergångarna mellan de olika tillstånden

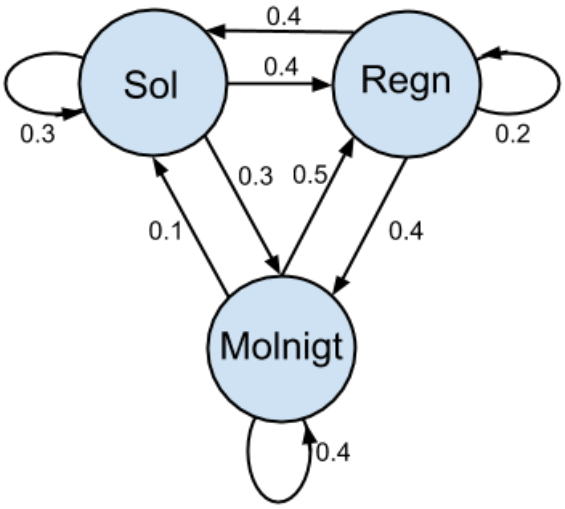


Figur 2.3: En Markovkedja där $\mathbf{S} = [A, B, C]$ är tillstånden och $\mathbf{A}$, vilket ges av (2.1), är de möjliga övergångarna. Författarnas egna bild.

beräknats. Sannolikheterna kan representeras med en övergångsmatris

$$\mathbf{A} = \begin{array}{c} \\ Sol \\ Regn \\ Molnigt \end{array} \begin{array}{c} \begin{array}{ccc} Sol & Regn & Molnigt \end{array} \\ \begin{bmatrix} 0.3 & 0.4 & 0.3 \\ 0.4 & 0.2 & 0.4 \\ 0.1 & 0.5 & 0.4 \end{bmatrix} \end{array}, \tag{2.2}$$

där matrisen är stokastisk i och med att varje rad av $\mathbf{A}$ har summan 1. Matrisen $\mathbf{A}$ representerar därmed vädermodellen, se *Figur 2.4*, där den till exempel visar att om det är soligt ute är sannolikheten 0.4 för att det ska regna dagen därefter.



Figur 2.4: En Markovkedja föreställande en vädermodell, där $\mathbf{S} = [Sol, Regn, Molnigt]$ är tillstånden och $\mathbf{A}$, vilket ges av (2.2), är de möjliga övergångarna. Författarnas egna bild.

Till exempel, låt oss säga att det är soligt när predikteringen börjar. Därav är $\pi = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ och sannolikheterna för morgondagens väder är

$$\mathbf{x}^{(1)} = \pi\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.3 & 0.4 & 0.3 \\ 0.4 & 0.2 & 0.4 \\ 0.1 & 0.5 & 0.4 \end{bmatrix} = \begin{matrix} Sol & Regn & Molnigt \\ \begin{bmatrix} 0.3 & 0.4 & 0.3 \end{bmatrix} \end{matrix}.$$

För att beräkna vädret för dagen därefter, det vill säga dag 2, utförs beräkningarna på samma sätt

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)}\mathbf{A} = \begin{bmatrix} 0.3 & 0.4 & 0.3 \end{bmatrix} \begin{bmatrix} 0.3 & 0.4 & 0.3 \\ 0.4 & 0.2 & 0.4 \\ 0.1 & 0.5 & 0.4 \end{bmatrix} = \begin{matrix} Sol & Regn & Molnigt \\ \begin{bmatrix} 0.28 & 0.35 & 0.37 \end{bmatrix} \end{matrix}.$$

Därav är den generalla uträkningen för dag n:

$$\mathbf{x}^{(n)} = \mathbf{x}^{(n-1)}\mathbf{A} = \pi\mathbf{A}^n.$$

2.3.2 Dold Markovmodell

Med en dold Markovmodel görs antagandet att en Markovkedja finns, men att dess tillstånd inte går att observera. Tillståndet som är dolt kan dock ge utdata som är observerbar. Till skillnad från en Markovkedja är HMM en fem-tupel, $< \mathbf{S}, \mathbf{A}, \pi, \mathcal{O}, \mathbf{B} >$ där $\mathbf{S}, \mathbf{A}, \pi$ är detsamma som i Markovkedjan. Därtill är $\mathcal{O}$ en ändlig mängd observationer samt $\mathbf{B}$, vilket är en ändlig mängd observationssannolikheter. $\mathbf{B} = \{b_{jk}\}$ är därmed en $N \times M$ stokastisk matris av M möjliga observationer vid N tillstånd,

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1M} \\ b_{21} & b_{22} & \dots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \dots & b_{NM} \end{bmatrix}$$

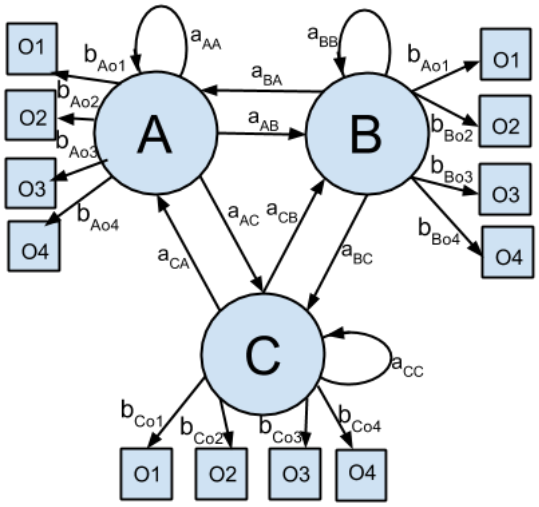
där

$$b_{jk} = P(\text{observation } k \text{ vid } t | \text{tillstånd } s_j \text{ vid } t).$$

I *Figur 2.5* kan ett allmänt fall av en dold Markovkedja beskådas, där $\mathbf{S} = [A, B, C]$, $\mathbf{A}$ ges som tidigare av (2.1) och

$$\mathbf{B} = \begin{bmatrix} b_{Ao_1} & b_{Ao_2} & b_{Ao_3} & b_{Ao_4} \\ b_{Bo_1} & b_{Bo_2} & b_{Bo_3} & b_{Bo_4} \\ b_{Co_1} & b_{Co_2} & b_{Co_3} & b_{Co_4} \end{bmatrix}. \tag{2.3}$$

Att få förståelse för en HMM kan vara prövande, dock har Stamp, M. (2012) ett välbeskrivet exempel i *A Revealing Introduction to Hidden Markov Models* som underlättar inlärandet.



Figur 2.5: En dold Markovmodell där $\mathbf{S} = [A, B, C]$ är tillstånden, $\mathbf{A}$ som ges av (2.1) och $\mathbf{B}$ som ges av (2.3). Författarnas egna bild.

Låt säga att den årliga medeltemperaturen ska bestämmas vid en specifik plats på jorden under ett flertal år. De år som är intressanta i detta fall ligger långt bak i tiden, innan termometern uppfunnits. I och med att det behövs istället indirekta bevis eftersökas för hur temperaturen betedde sig.

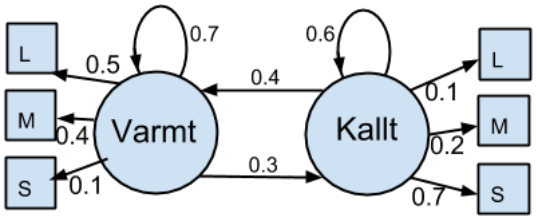
För att simplifiera problemet övervägs endast två årliga temperaturer, *Varmt* och *Kallt*. Låt oss även säga att moderna bevis indikerar att sannolikheten för att ett varmt år följs av ännu ett varmt år är 0.7 och att sannolikheten för att ett kallt år följs av ännu ett kallt år är 0.6. Antagande om att informationen även håller för långt bak i tiden görs, varav den information som erhållits kan summeras till

$$\begin{matrix} & V & K \\ V & 0.7 & 0.3 \\ K & 0.4 & 0.6 \end{matrix} \tag{2.4}$$

där V representerar $Varmt$ och K representerar $Kallt$.

Utöver detta antas även att nutida forskning indikerar att en korrelation existerar mellan storleken på träds årsringar och temperatur. För enkelhetens skull är det endast tre storlekar av årsringar som tas hänsyn till, vilka är *Liten*, *Mellan* och *Stor* eller L, M och S. Slutligen antas baserat på framsteg inom forskningen att det sannolikhetsmässiga sambandet mellan den årliga temperaturen och storleken på trädringar ges av

$$\begin{matrix} & S & M & L \\ V & 0.1 & 0.4 & 0.5 \\ K & 0.7 & 0.2 & 0.1 \end{matrix}. \tag{2.5}$$



Figur 2.6: En dold Markovmodell föreställande den årliga medeltemperaturen, där $\mathbf{S} = [Varmt, Kallt]$ är tillstånden, $\mathbf{A}$ ges av (2.6) och $\mathbf{B}$ av (2.7). Författarnas egna bild.

För det här systemet, se *Figur 2.6* sätts tillstånden till den årliga temperaturen, det vill säga $\mathbf{S} = \begin{bmatrix} V & K \end{bmatrix}$. I och med att tillstånden är dolda och att probabilistisk information erhålls angående temperaturen från trädringarnas storlek kan problemet modelleras som en dold Markovmodel. Utifrån (2.4) fås den stokastiska övergångsmatrisen

$$\mathbf{A} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \tag{2.6}$$

och från (2.5) fås den stokastiska observationsmatrisen

$$\mathbf{B} = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix}. \tag{2.7}$$

I det här exemplet antas även att den stationära fördelningen π är

$$\pi = \begin{matrix} V & K \\ \begin{bmatrix} 0.6 & 0.4 \end{bmatrix} \end{matrix}. \tag{2.8}$$

Låt oss nu säga att en fyraårsperiod undersöks, där årsringar med följande storlek L, M, L, S observeras. 0 representerar L, 1 representerar M och 2 representerar S, vilket ger observationssekvensen

$$\mathcal{O} = \begin{bmatrix} 0 & 1 & 0 & 2 \end{bmatrix}. \tag{2.9}$$

Utifrån dessa observationer behöver sannolikheten för vilket tillstånd som var under de fyra observationerna predikteras, det vill säga för de olika storlekarna av årsringarna. För varje observation finns det två olika tillstånd, V och K. Det medför att det med dessa fyra observationer finns 16 olika kombinationer av tillståndsföljder. Till exempel $VVVV$, $VVVK$, $VVKK$. Alla dessa sannolikheter behöver beräknas för att slutligen sammanställa den totala sannolikheten för att det fjärde tillståndet antingen är V eller K.

I detta fall beskrivs en generisk följd med tillstånd av längden fyra som följande

$$X = (x_0, x_1, x_2, x_3)$$

med de korresponderande observationerna som beskrivs enligt

$$\mathcal{O} = [\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3]$$

Tillstånd	Sannolikhet	Normaliserad sannolikhet
$VVVV$	0.000412	0.042787
$VVVK$	0.000035	0.003635
$VVKV$	0.000706	0.073320
$VVKK$	0.000212	0.022017
$VKVV$	0.000050	0.005193
$VKVK$	0.000004	0.000415
$VKKV$	0.000302	0.031364
$VKKK$	0.000091	0.009451
$KVVV$	0.001098	0.114031
$KVVK$	0.000094	0.009762
$KVKV$	0.001882	0.195451
$KVKK$	0.000564	0.058573
$KKVV$	0.000470	0.048811
$KKVK$	0.000040	0.004154
$KKKV$	0.002822	0.293073
$KKKK$	0.000847	0.087963

Tabell 2.1: Tillståndssekvenssannolikheter

Sannolikheten för tillståndssekvensen X beräknas som följande

$$P(X) = \pi_{x_0} b_{x_0\mathcal{O}_0} \prod_{i=1}^{3} a_{x_{i-1}x_i} b_{x_i\mathcal{O}_i} = \pi_{x_0} b_{x_0\mathcal{O}_0} a_{x_0x_1} b_{x_1\mathcal{O}_1} a_{x_1x_2} b_{x_2\mathcal{O}_2} a_{x_2x_3} b_{x_3\mathcal{O}_3} \quad (2.10)$$

där π_{x_0} är sannolikheten att starta i tillståndet x_0, $b_{x_0}(\mathcal{O}_0)$ är sannolikheten att observera $\mathcal{O}_0$ i tillståndet x_0 och a_{x_0,x_1} är sannolikheten för en övergång från tillståndet x_0 till x_1.

Om till exempel sannolikheten för att tillståndsföljden $VVKK$ sker ska beräknas och den information tidigare sammanställts appliceras, se (2.6), (2.7), (2.8) och (2.9), i (2.10) erhålls:

$$P(VVKK) = 0.6(0.1)(0.7)(0.4)(0.3)(0.7)(0.6)(0.1) = 0.000212$$

Med (2.10) återupprepas uträkningen för alla tillståndsföljder, där dess sannolikheter kan ses i *Tabell 2.1.* I samma tabell i kolumnen längst till höger finns de normaliserade sannolikheterna listade, vilket innebär att de har multiplicerats med en normaliseringskonstant för att summan av alla sannolikheter ska vara lika med 1.

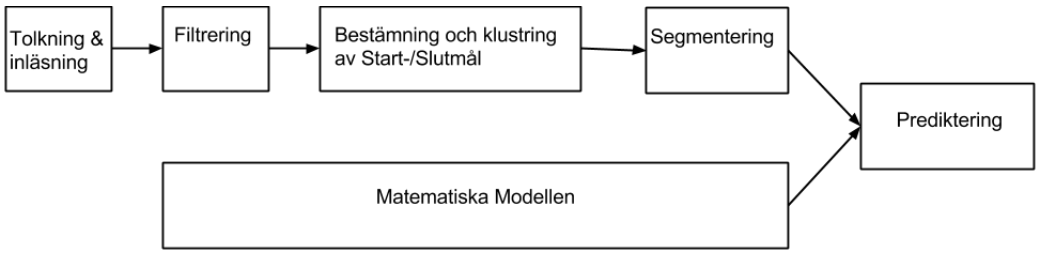
I *Tabell 2.2* kan sannolikheterna för att medeltemperaturen ett visst år har varit kallt eller varmt ses. Dessa fås ut genom att summera de som delar samma tillstånd ett visst år i *Tabell 2.1.*

	År			
	0	1	2	3
P(V)	0.188182	0.519576	0.228788	0.804029
P(K)	0.811818	0.480424	0.771212	0.195971

Tabell 2.2: Sannolikheterna, uträknade med en HMM, för att det varit ett kallt eller varmt år.

3
Metod

I detta kapitel kommer tillvägagångssättet beskrivas. Kapitlet börjar med vilka resurser som funnits tillgängliga och hur datan från Volvo Personvagnar bearbetats för att kunna användas i den slutgiltiga produkten. Efter det följer uppbyggnaden av datamodellen och till slut den matematiska modellen som även sköter predikteringen. I *Figur 3.1* syns det tydligt hur arbetet fortlöpt, samt vilka delar som varit beroende av varandra och vilka delar som har genomförts parallellt. För att mäta hur bra modellen predikterar bildfärderna kommer resultat från tidigare studier tas i beaktning och jämföras med resultaten för denna modell.



Figur 3.1: Flödesschema över arbetsmetoden. Författarnas egna bild.

Den största skillnaden från tidigare forskning inom området är att den lösning som presenteras för prediktering i denna rapport är mer baserad på vart föraren kommer ifrån och vilken väg den tar, än vart föraren befinner sig just nu. Hur detta påverkar predikteringen diskutera i diskussionen.

3.1 Resurser

För att genomföra detta projekt kommer det krävas ett par hjälpmedel. I detta avsnitt beskrivs dessa för att underlätta det fortsatta läsandet.

3.1.1 Rutter

Rutterna som används i uppbyggnaden och testningen kommer i ett format som liknar JSON-format. Alla rutter är erhållna från Volvo Personvagnar. De är insamlade med hjälp av en GPS-sensor i en personbil. Alla rutter är körda i Göteborgsområdet under 2 månader. Totalt erhålls 242 stycken rutter som är insamlade med ett intervall med en GPS-punkt per sekund. Efter omvandling och tolkning kommer denna data kunna användas i MATLAB.

3.1.2 MATLAB

MATLAB är ett programmeringsspråk och interaktiv miljö som används överallt av ingenjörer och forskare. Det är specifikt för matematiska algoritmer och för att visualisera idéer och lösningar. MATLAB är skapat av företaget MathWorks, Inc som är ledande i att skapa matematiska mjukvaror. I MATLAB finns många inbyggda funktioner, vilket kommer leda projektet framåt smidigt.

3.1.3 OpenStreetMap

Alla bilder tagna från MATLAB innehåller kartor från OpenStreetMap. All deras data är tillgänglig under ”Opesn Database License” och deras kartografi är licenserad under CC BY-SA.

3.2 Tolkning och Inläsning

De filer som tillhandahölls av Volvo Personvagnar kom i ett format liknande JSON-format. JSON är ett kompakt format som är känt för att både vara lätt att läsa in av en dator och vara lättläst för en människa. JSON är baserat på Javascript, men kan lätt läsas in i en mängd olika program och språk. Bland annat går det utmärkt att läsa in i MATLAB. Eftersom JSON kan läsas in av en rad olika programmeringsspråk används det ofta för att kunna kommunicera mellan olika språk (Ecma International, 2013).

Totalt erhölls 242 filer, dock var det tyvärr ett par mindre fel i filerna som först behövdes lösas för att kunna behandla filerna som JSON. Detta löstes med hjälp av en parser som går igenom samtliga rutter och rättar till dem. De få filerna som inte kunde lagas med hjälp av parsern kastades bort. Totalt kastades fem filer bort. För att inte behöva ändra många felaktiga filer manuellt kan en parser skapas för att automatisera processen. De filer som kastades bort var tillräckligt få för att inte påverka slutresultatet. När rutterna är tillrättade kan de läsas in i MATLAB och kategoriseras efter de krav som den matematiska modellen kräver.

3.3 Klassificering och gruppering

3.3.1 Filtrering

En viktig del av projektet är att göra den data som erhållits mer arbetsvänlig. Den första delen av detta var att filtrera bort felaktiga och onödiga GPS-punkter och sträckor. Filtreringen kommer att ske med avseende på en rad olika faktorer.

- **Få bort felaktiga punkter**.
  I vissa rutter finns det exempelvis punkter mitt i sydatlanten, utanför Afrikas kust. Dessa punkter avlägsnas genom att kontrollera avstånd och tid mellan två punkter och se om det är rimligt att hålla den hastighet som krävs. Den

maximalt tillåtna hastigheten bestäms till 1 km/s, och hastigheter som överstiger det kommer filtreras bort. Detta är en orimligt hög hastighet för en bil, men eftersom den enbart behöver plocka bort punkter som är extremt fel kommer det vara tillräckligt lågt. Ibland när GPS:en startas har den inte kalibrerat sig och kan få in flera punkter med koordinaterna (0,0). För att undvika att dessa punkter ansätts som startpunkter tas dessa bort direkt.

- **Få ner antalet punkter**.
  I och med att datan ursprungligen kommer från en GPS-sensorn i fordonet som uppdateras inom väldigt korta tidsintervall erhålls en stor datamängd för en liten sträcka. Mycket av den data som samlas in från GPS-mottagaren innehåller nästan identisk information. Fenomenet förvärras ytterligare när ett fordon håller väldigt låg hastighet eller står stilla vilket resulterar med en datamängd som är orimligt stor. För att göra datamängden mer hanterlig och arbetsbar sätts ett krav att mellanrummet mellan två GPS-punkter skall minst vara 30 meter annars avlägsnas den senare punkten. Detta kommer effektivisera uppbyggnaden av modellen och göra processen mindre tidskrävande.

- **Ta bort för korta rutter**.
  Det finns rutter som är för korta för att ge relevans till predikteringen. Exempelvis finns det rutter där bilen bara startas och sedan inte körs iväg, eller att föraren bara byter parkeringsruta med sitt fordon. Därför kommer rutter som innehåller mindre än 10 GPS-punkter efter tidigare utförd filtrering, det vill säga är kortare än 300 meter att filtreras bort.

Efter denna filtrering fanns det 198 rutter kvar som kunde användas för att bygga upp och testa modellen.

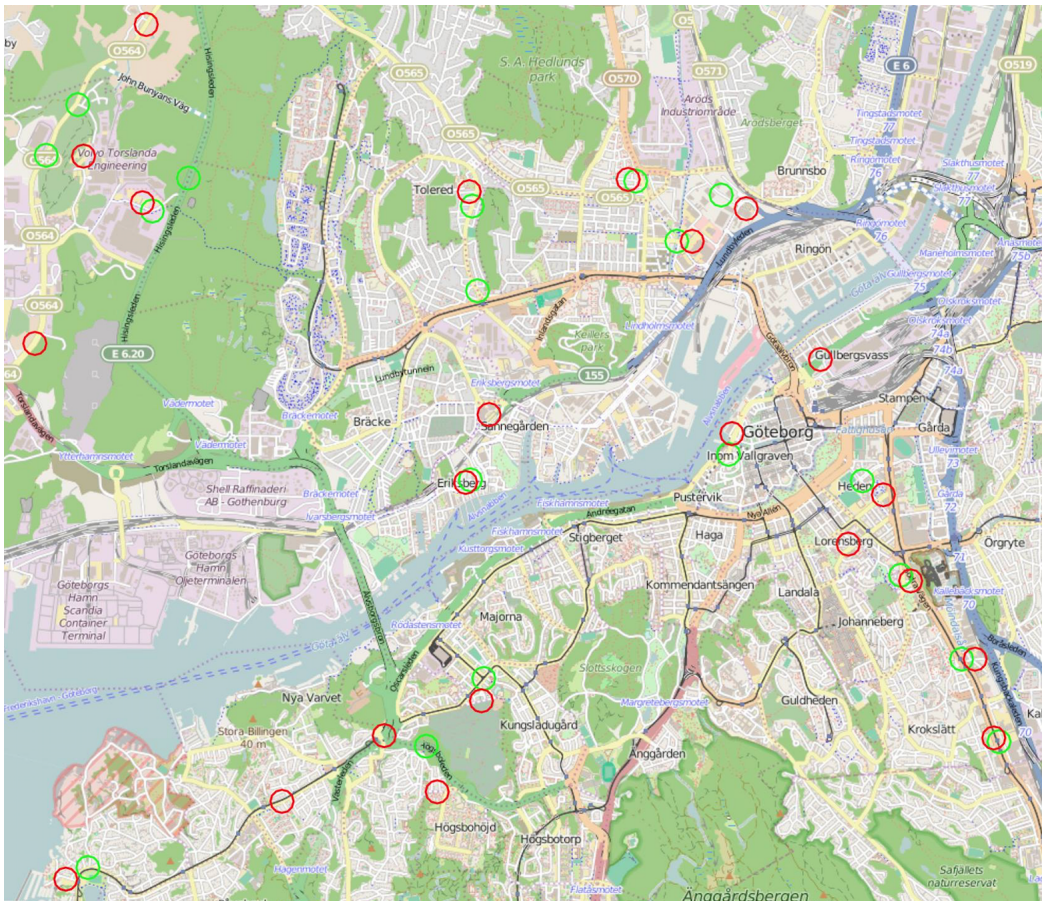
3.3.2 Bestämma och klustra start-/slutmål

GPS-punkterna som beskriver start- och slutmål för rutterna kan skilja sig från varandra trots att det är samma start- eller slutmål. Detta kan bero på en osäkerhet i GPS-punkterna, men även att en parkeringsplats oftast inte begränsas till en parkeringsruta utan till ett areamässigt större område. Det kan även bero på att en förare oftast inte parkerar på exakt samma parkeringsplats vid en slutdestinaton, eller möjligtvis parkererar bilen i kvarteret runt bostaden.

Genom att klustra ihop start- och slutpunkter, som tycks tillhöra samma område till ett enda kluster kan tilldelning av punkter till olika start- eller slutpositioner trots att de egentligen tillhör samma position undvikas. Detta kommer att göras genom att plattformen anser att punkter med en radie på maximalt 60 meter från en tidigare start- eller slutpostition tillhör samma start- eller slutmål, se *Figur 3.2*. Detta innebär att ett klustrat start-/slutmål är en area med en viss radie som täcker in flera GPS-punkter som ska tillhöra samma start-/slutmål.

I MATLAB kommer start-/slutmålen representeras som två vektorer med de GPS-punkter som skapar punkten. När programmet ska kolla om en punkt tillhör samma

start eller slutmål kommer den kolla om den nya punkten ligger inom en radie på 60 m från någon tidigare punkt i start eller slutmålsvektorn. Om den gör det, kommer programmet anse att det är samma start eller slutmål. Annars läggs den nya punkten till i start eller slutmålsvektorn. Om en GPS-punkt inte matchar ett klustrat startmål har rutten aldrig körts tidigare, och prediktering kommer inte kunna utföras.



Figur 3.2: Klustrade start- och slutmål utritade på en karta över Göteborg, där de gröna cirklarna är startmål och röda slutmål.
©OpenStreetMap Contributors

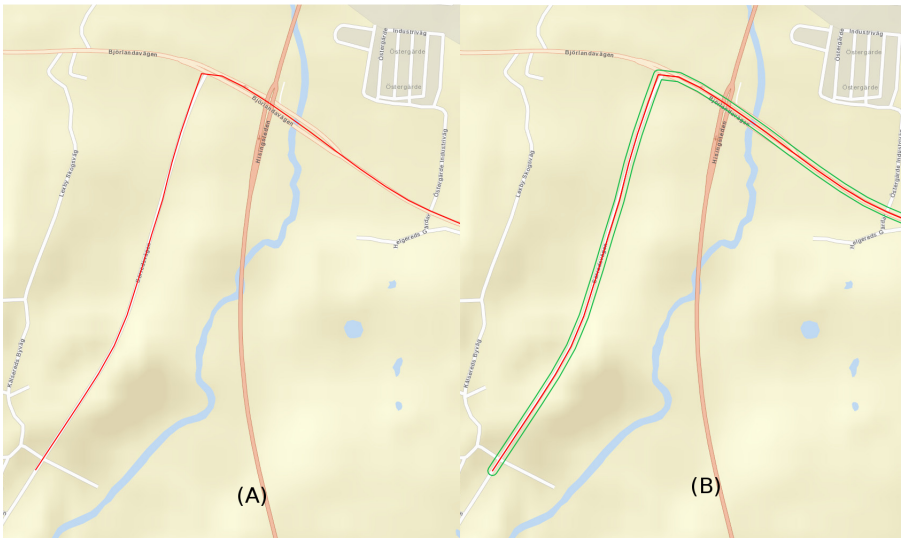
3.3.3 Segmentering

I bakgrunden beskrivs två olika sätt att bygga upp en karta. Antingen med hjälp av segmentbitar eller med stödpunkter. I detta projekt används segment då det är mest välutforskat och lättare att implementera.

Plattformen ska bygga upp en egen karta med de rutter som erhålls. Rutterna kommer att segmenteras genom att först skapa en buffertzon runt varje punkt som bildar rutten. Buffertzonen kommer skapas med en radie på 30 m. Därefter kommer

alla dessa buffertzoner läggas samman och bilda ett segment. Anledningen till att rutterna behöver segmenteras är för att GPS-punkterna, för nästkommande gång samma rutt färdas, troligtvis inte kommer att vara identiska med den första gången. Detta på grund av att föraren då troligtvis inte kör på exakt på samma del av vägen, men även på grund av att det finns en osäkerhet i GPS-positionen och att en väg till exempel kan vara flerfilig. Om GPS-punkter för senare rutter hamnar i en buffertzon tillhörande en rutt, kan det antas att fordonet färdats på samma sträcka. Själva radien på buffertzonen bör vara tillräckligt stor för att få med de rutter som antas vara lika, men inte heller för stor. Om buffertzonen har en för stor radie finns det en risk att den överlappar intilliggande vägar. Det skulle betyda att de två helt skilda vägarna skulle dela samma segment.

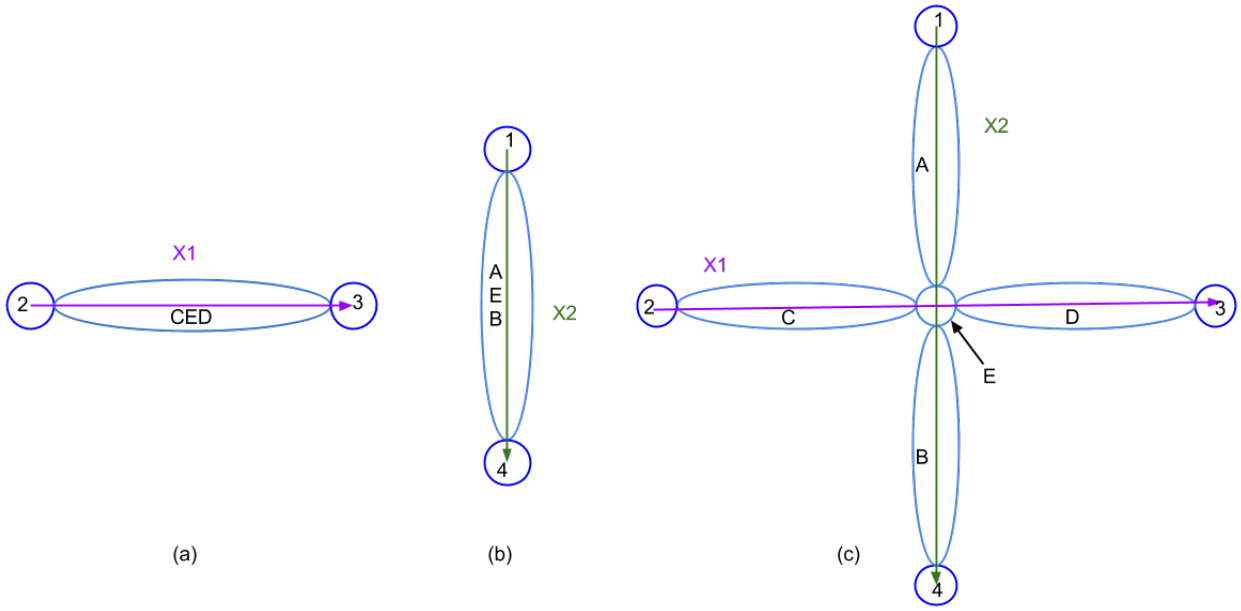
I *Figur 3.3* syns det i (A) hur rutten såg ut innan en buffertzon lades runt ruttens punkter. I (B) syns det hur det ser ut när programmet har lagt en buffertzon runt den befintliga rutten.



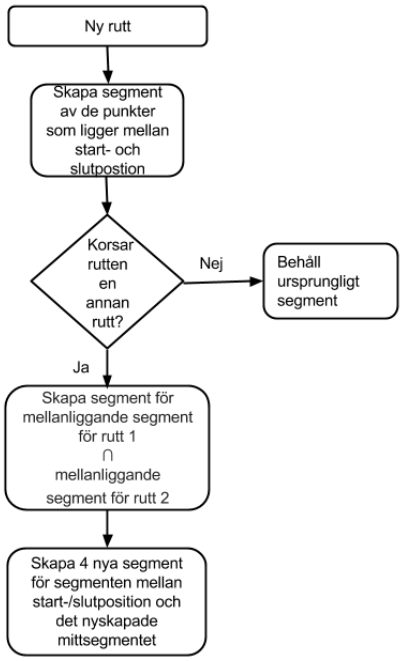
Figur 3.3: (A) Rutt utan buffertzon. **(B)** Rutt med buffertzon.
©OpenStreetMap Contributors

När endast en rutt har lästs in kommer den första punkten räknas som startpunkt och den sista punkten som slutmål. Punkterna däremellan kommer bilda ett segment. Därefter när en ny rutt läses in som skiljer sig från tidigare rutter samt korsar någon av dessa kommer segmentering att ske och nya segment skapas. Segmenten är alltså inte beroende av korsningar och svängar i verkligheten, utan bara de delningar som sker mellan de rutter som föraren själv har utfört och start-/slutmålen. I *figur 3.4* visas hur segmenteringen ser ut när två vägar, X1 och X2 korsas. I (a) och (b) visar rutterna X1 respektive X2 om de inte hade korsat varandra. 1, 2, 3 och 4 är start- och slutpositionerna med en cirkel med en radie på X runt som används för att se om om en ny punkt tillhör ett tidigare start-/slutmål. När rutterna korsar varandra delas segmenten upp genom att snittet av X1s segment som tidigare var CED, och X2s segment som tidigare var AEB beräknas. Ett nytt segment, E bildas

i korsningen av snittet för de två rutterna. Vidare delas segmenten för X1 och X2 upp i C och D respektive A och B, vilket kan ses i (c). Tillvägagångssättet för segmentering kommer alltså vara enligt *Figur 3.5*.



Figur 3.4: (a) Segmentering för rutten X1 (b) Segmentering för rutten X2 (c) Segmentering när rutten X1 och X2 korsar varandra. Författarnas egna bild.



Figur 3.5: Flödesschema segmentering. Författarnas egna bild.

I MATLAB kommer segmenten att representeras i form av en matriser med de

$$\mathbf{ST} = \begin{matrix} Namn \\ 1 \\ 2 \end{matrix} \begin{matrix} GPS\text{-}punkt\ för\ start\text{-}/slutmål \\ \left[\begin{matrix} p_1 \\ p_2 \end{matrix}\right] \end{matrix} \text{ där } p_1 = (lat_1, long_1)$$

Figur 3.6: Startvektorn **ST** utifrån *Figur 3.4*.

GPS-punkter som ligger innanför segmentet. Det vill säga punkterna som bygger upp sträckan CED och AED i *figur 3.4 (a) och (b)*. Alla vektorer kommer beskrivas mer i detalj i avsnittet om datamodellen, 3.4.

3.4 Datamodellen

3.4.1 Tillstånd

En viktig byggsten i en HMM och även i den modell som byggs upp är de tillstånd som används. Ett tillstånd kan innehålla variabler som tid, plats, handling, start-/slutmål med mera. I detta fall kommer två variabler tas i beaktning, ett segment som fordonet befunnit sig i under färdens gång och slutmålet för rutten. Ett segment kan därmed innehålla ett flertal tillstånd då segmentet kan ha passerats ett flertal gånger med olika slutmål.

3.4.2 Datamodellens uppbyggnad

Datamodellen ska vara självinlärande, vilket innebär att den ska sköta segmenteringen av rutterna samt skapandet av start-/slutmål som nämnts tidigare. Därefter ska två vektorer och två matriser skapas som senare kommer användas av den matematiska modellen. Den ska även skapa de tillstånd som används i denna modell som bygger på en HMM. Ett tillstånd i vår modell består av ett segment och ett slutmål. Det gör att en rutt lätt kan representeras av en rad med tillstånd. Detta gör också att när bilen befinner sig i ett specifikt segment finns det flera olika tillstånd bilen kan befinna sig i.

För start och slutmål kommer det skapas två vektorer som kommer ha samma uppbyggnad, en som benämns startvektor (**ST**) och en som benäms slutmålsvektor (**SL**). Dessa två kommer bestå av samtliga start-/slutmål, se *Figur 3.6*. Varje rad i vektorn innehåller en GPS-punkt som beskriver ett start-/slutmål. GPS-punkten kommer beskrivas som latitud och longitud i vektorn. I *Figur 3.4* representeras de fyra start-/slutmålen av mörkblå cirklar och segmenten av de ljusblå ovala.

För att se om nytt start-/slutmål är samma som ett befintligt kommer programmet kolla om den nya punkten ligger inom en radie på X från någon av de befintliga punkterna i **ST** eller **SL**. Om det nya start-/slutmålet inte är samma som någon av de befintliga kommer **ST** eller **SL** uppdateras beroende på om det är ett start eller slutmål och det nya start-/slutmålet läggs till i **ST** eller **SL**.

$$\mathbf{SL} = \begin{matrix} Namn \\ 3 \\ 4 \end{matrix} \begin{matrix} GPS\text{-}punkt\ för\ start\text{-}/slutmål \\ \begin{bmatrix} p_3 \\ p_4 \end{bmatrix} \end{matrix} \quad \text{där } p_3 = (lat_3, long_3)$$

Figur 3.7: Slutmålsvektorn **SL** utifrån *Figur 3.4*.

$$\mathbf{SE} = \begin{matrix} Namn \\ A \\ B \\ C \\ D \end{matrix} \begin{matrix} Segment \\ \begin{bmatrix} p_{A_1}, \ldots, p_{A_i} \\ p_{B_1}, \ldots, p_{B_j} \\ p_{C_1}, \ldots, p_{C_k} \\ p_{D_1}, \ldots, p_{D_l} \end{bmatrix} \end{matrix} \quad \text{där } p_{A_1} = (lat_{A_1}, long_{A_1}) \text{ och } i, j, k, l \in \mathbb{N}_1$$

Figur 3.8: Segmentmatrisen **SE** utifrån *Figur 3.4*.

Den första matrisen, segmentsmatrisen (**SE**) kommer vara uppbyggd med en mängd GPS-punkter i form av latitud och longitud. Dessa GPS-punkter är punkterna som är mellan start- och slutpunkterna, det vill säga punkterna som en buffertzon kommer läggas runt vid segmenteringen, se *Figur 3.8*. Vid skapandet av ett nytt segment kommer programmet gå igenom elementen i **SE** och kolla om punkterna för det nya segementet är inom en radie på för de GPS-punkter i **SE**. Om inte uppdateras **SE** med GPS-punkterna för det nya segmentet utan buffertzon.

Det är **ST**, **SL** och **SE** som kommer användas för att undersöka om en observerad punkt under prediktionen befinner sig inom den uppbyggda modellen. Genom att iterera igenom **ST**, **SL** och **SE** kan datamodellen hitta GPS-punkter som matchar den observerbara punkten. Exakt hur det går till beskrivs i avsnitt 3.5.1.

Tillståndsmatrisen (**S**) kommer vara den mest väsentliga och se annorlunda ut i jämförelse med **ST**, **SL** och **SE** som är beskrivna ovan. I denna matris kommer varje rad bestå av en rutt, representerad i form av en följd av tillstånd och antalet gånger varje rutt körts. Den första kolumnen visar hur många gånger föraren har kört rutten, den andra kolumnen innehåller start- och slutposition och sedan fylls de resterande kolumnerna på med de segment som passeras för att nå slutmålet och det aktuella slutmålet. Rutterna kan ha olika antal kolumner i **S**, det beror på att de antal segment som gås igenom för att nå en viss slutdestionation inte är samma för alla rutter. Tillstånden finns utförligt beskrivna i avsnitt 3.5.1. Exempelvis kan en rutt som startar i 1, därefter går genom segmenten A, E och B och slutar i 4 se ut som följande: ((1,4) (A,4) (E,4) (B,4) (4,4)), se *Figur 3.9*. **S** kommer användas istället för övergångs-matrisen som vanligtvis används i en HMM. Exakt hur den används beskrivs under avsnitt 3.5. Med hjälp av antalet gånger rutterna körts kan den matematiska modellen beräkna sannolikheter.

$$\mathbf{S} = \begin{array}{c} \\ X1 \\ X2 \end{array} \begin{array}{c} \begin{array}{cccccc} \textit{Rutt} & \textit{Antal} & \textit{Tillstånd} & & & \end{array} \\ \begin{bmatrix} 1 & (1,4) & (A,4) & (E,4) & (B,4) & (4,4) \\ 1 & (2,3) & (C,3) & (E,3) & (D,3) & (3,3) \end{bmatrix} \end{array}$$

Figur 3.9: Tillståndsmatrisen **S** utifrån *Figur 3.4*.

3.4.3 Maskininlärning

Plattformen ska som tidigare nämnts själv skapa **S**, **ST**, **SL**, **SE**. Segmenteringen och skapandet av start-/slutmål beskrivs i avsnitt 2.2.2 respektive 2.2.3. Varje gång ett nytt start- eller slutmål skapas läggs de till i **ST** eller **SL** beroende på om det är en start- eller slutdestination.

Vid första rutten som modellen får skapas ett klustrat startmål, ett segment för punkter mellan start- och slutmål och ett klustrat slutmål. När sedan de andra sträckorna körs genom programmet tar programmet de punkt för punkt och kollar om de ligger i redan existerande segment eller inte. För de punkter som ligger utanför eller mellan de existerande segmenten skapas nya segment. Medan de punkter som ligger inuti ett existerande segment sägs tillhöra det existerade segmentet, alternativt delar upp det existerande segmentet i flertalet delar. Hur många delar som krävs beror på hur den nya rutten korsar det gamla segmentet. Ett par olika delningar visas i bilderna nedan:

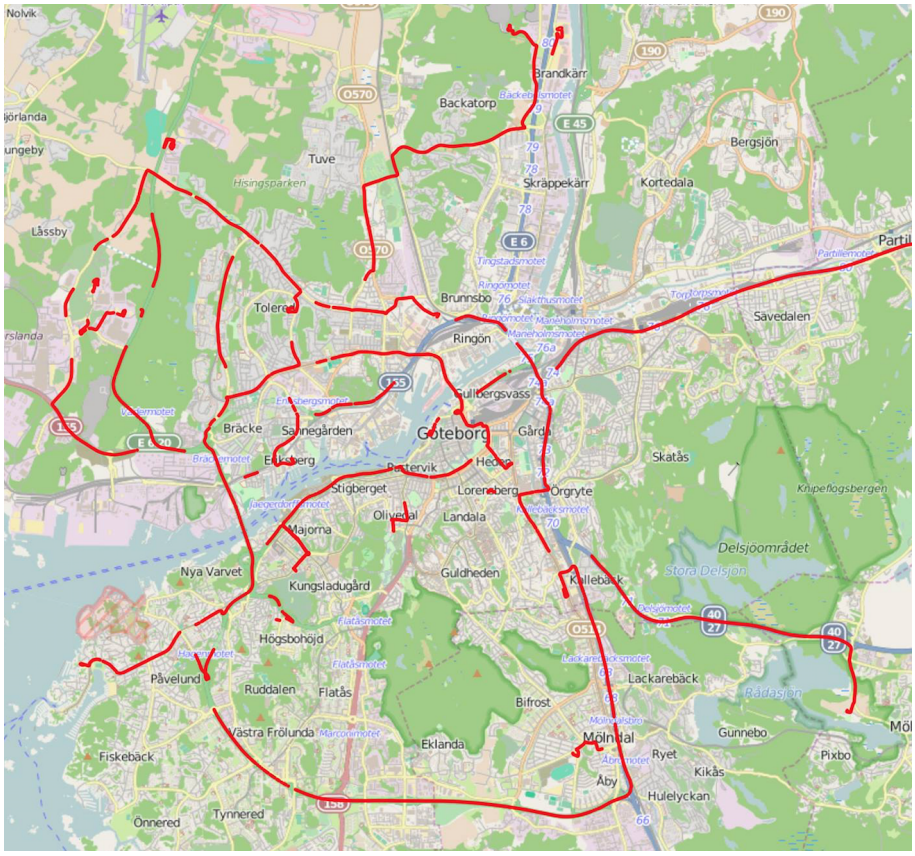
När samtliga rutter körts igenom av programmet har en fullständig karta byggts upp av väldigt många segment, se *Figur 3.10*. Den har även byggt upp de två vektorerna och de två matriserna som beskrivits tidigare. Hur dessa kommer användas för predikteringen beskrivs i kapitel 3.4

3.5 Matematiska modellen

3.5.1 Prediktering, steg för steg

För predikteringen över vart slutdestinationen för en rutt kommer ligga är det tillståndsmatrisen **S** som innehar den mest aktiva rollen. Plattformen kommer från när föraren påbörjat sin färd till att föraren har nått slutdestinationen få in observationer i form av GPS-punkter från fordonet eller en annan enhet. Applicerat som en dold Markovmodell är det även utifrån den första kolumnen i **S** som den stationära fördelningen π av alla starttillstånd erhålls från, detta genom att

$$\pi = \begin{bmatrix} \frac{s_{11}}{T} \\ \vdots \\ \frac{s_{R1}}{T} \end{bmatrix}$$



Figur 3.10: Fullständig karta med samtliga segment. ©OpenStreetMap Contributors

där R är antalet rader i $\mathbf{S}$ och $T = \sum_{i=1}^{R} s_{i1}$ är det totala antal alla rutter färdats.

Programmet kommer sedan att ta den första erhållna GPS-observationen och se om den observationen befinner sig i någon av areorna utifrån GPS-punkterna $p_{1,\ldots,K}$, där K är antalet element i $\mathbf{ST}$, som representerar startmålen i $\mathbf{ST}$. Om observationen inte befinner sig i någon av $st_{1,\ldots,K}$ är prediktion omöjlig att utföra då föraren aldrig har startat i det startmålet tidigare och därmed saknar statistik. Plattformen kommer då istället med hjälp av maskininlärning lära sig den färd som sker. Detta sker genom att $\mathbf{ST}$, $\mathbf{SL}$, $\mathbf{SE}$ och $\mathbf{S}$ uppdateras med den nya rutten.

Om observationen befinner sig i någon av $st_{1,\ldots,K}$ registreras startmålet, låt oss säga som st^*, i plattformen och programmet vänder sig mot tillståndsmatrisen $\mathbf{S}$. Där går den igenom det första tillståndet, det vill säga den andra kolumnen i $\mathbf{S}$ och noterar om startmålet i tillstånden, $\{s_{(1,\ldots,R)2}\}_1$ är densamma som det aktuella startmålet st^*. Ur ett HMM-perspektiv blir detta

$$\mathbf{B}_{1r} = b_{\{s_{r2}\}_1 \mathcal{O}_0} = P(\mathcal{O}_0 | \{s_{r2}\}_1)_r = \begin{cases} 1, & \text{om } \mathcal{O}_0 \in \{s_{r2}\}_1 \\ 0, & \text{om } \mathcal{O}_0 \notin \{s_{r2}\}_1 \end{cases} \quad \text{där } r = 1, \ldots, R$$

där observationssmatrisen $\mathbf{B}$ ej är stokastisk, vilket den ska vara i en HMM. Anledningen till detta är att vår modell inte är en HMM till fullo, utan bara har sin

grund i den.

För de rutter som delar st^* som startmål, $\{s_{(1,\dots,R)2}\}_1 = st^*$, erhålls de första sannolikheterna för vart slutmålet kommer vara, $P(\mathcal{O}_0|\mathbf{SL})$. Detta genom att den tar antalet gånger för rutter som delar startmålet st^* och ett slutmål från första kolumnen i tillståndsmatrisen $\mathbf{S}$, $T_{st^*}^{sl_j}$ $j = 1, \dots, J$, och delar med det totala antal alla rutter färdats som delar st^*, T_{st^*}

$$P(\mathcal{O}_0|\mathbf{SL}) = \begin{bmatrix} \frac{T_{st^*}^{sl_1}}{T_{st^*}} \\ \vdots \\ \frac{T_{st^*}^{sl_J}}{T_{st^*}} \end{bmatrix}.$$

Vidare i perspektivet utifrån en dold Markovmodell sker dessa uträkningar för samma ändamål

$$P(\mathcal{O}_0|\mathbf{SL}) = P(\mathcal{O}_0|\mathbf{SL})_j = \sum_{i=1}^{R} x_{ij} \quad j = 1, \dots, J, r = 1, \dots, R$$

där

$$x_{ij} = \begin{cases} P(\{\mathcal{O}_0|s_{r2}\}_1)_i, & \text{om } sl_j \in \{s_{i2}\}_2 \\ 0, & \text{om } sl_j \notin \{s_{i2}\}_2 \end{cases} \text{ och } P(\mathcal{O}_0|\{s_{r2}\}_1) = \pi b_{\{s_{r2}\}_1 \mathcal{O}_0} \quad j = 1, \dots, J, r = 1, \dots, R.$$

vilket multiplicerat med en normaliseringskonstant ger den normaliserade sannolikheten (det vill säga att summan av alla sannolikheter blir 1) för alla slutmål.

Om startmålet existerar och föraren därefter lämnar dess area tar plattformen nästkommande GPS-observation $\mathcal{O}_1$ och vänder sig till segmentmatrisen **SE**. Där undersöker den på samma sätt som med startmålsvektorn **ST** innan om observationen befinner sig inom någon av segmentets areor $se_{1,\dots,L}$, där L är det totala antalet segment, som byggs upp utifrån en serie av GPS-punkter. Om inte, blir det som tidigare att prediktering inte är möjlig att utföra. I det andra fallet, det vill säga att observationen existerar inom en av segmentens areor registreras segmentet, låt oss säga som se^*. Därefter går programmet igenom den tredje kolumnen i $\mathbf{S}$, det vill säga det andra tillståndet, för de rutter som i det första tillståndet innehöll startmålet st^* och undersöker om det innehåller se^*, $\{s_{(1,\dots,R)3}\}_1 = se^*$. Observationssannolikheten för modellen beräknas som tidigare fast utifrån tredje kolumnen i $\mathbf{S}$

$$\mathbf{B}_{2r} = b_{\{s_{3r}\}_1 \mathcal{O}_1} = P(\mathcal{O}_1|\{s_{r3}\}_1)_r = \begin{cases} 1, & \text{om } \mathcal{O}_1 \in \{s_{r3}\}_1 \\ 0, & \text{om } \mathcal{O}_1 \notin \{s_{r3}\}_1 \end{cases} \quad \text{där } r = 1, \dots, R.$$

I samband med detta kommer även övergånsmatrisen $\mathbf{A}$ beräknas som

$$\mathbf{A}_{1r} = a_{\{s_{r2}\}_1 \{s_{r3}\}_1} = P(\{s_{r3}\}_1|\{s_{r2}\}_1) = 1 \quad r = 1, \dots, R$$

där den ej heller kommer vara en stokastisk matris av samma anledningar som för $\mathbf{B}$. Anledningen till varför övergångssannolikheten alltid är lika med 1 är för att

tillståndsmatrisen **S** är uppbyggd så att sannolikheten är 1 för att ett tillstånd i en rutt går till nästkommande tillstånd, till exempel att $s_{12} \rightarrow s_{13}$.

I samband med segmentsbytet kommer sannolikheterna att uppdateras, då vissa av rutterna inte delar segmentet i tillståndet $\{s_{(1,\ldots,R)3}\}_1 = se^*$, men delade startmålet i det första tillståndet $\{s_{(1,\ldots,R)2}\}_1 = st^*$. Sannolikheterna för de olika slutmålen beräknas med hjälp av $P(\mathcal{O}_0|\mathbf{SL})$ genom att

$$P(\mathcal{O}_1|\mathbf{SL}) = P(\mathcal{O}_0|\mathbf{SL})^T \begin{bmatrix} \frac{T_{se^*}^{sl_1}}{T_{se^*}} & \ldots & \frac{T_{se^*}^{sl_J}}{T_{se^*}} \end{bmatrix}$$

där $T_{se^*}^{sl_j}$ är antalet rutter som delar slutmålet sl_j $j = 1, .., J$ och nuvarande segmentet i tillståndet $\{s_{(1,\ldots,R)3}\}_1 = se^*$ medan T_{se^*} är antalet rutter som delar $\{s_{(1,\ldots,R)3}\}_1 = se^*$. $P(\mathcal{O}_1|\mathbf{SL})$ är dock i behov att multipliceras med en normaliseringskonstant för att de korrekta sannolikheterna ska erhållas.

För att beräkna sannolikheterna utifrån en dold Markovmodell utförs följande beräkningar

$$P(\mathcal{O}_1|\mathbf{SL}) = P(\mathcal{O}_1|\mathbf{SL})_j = \sum_{i=1}^{R} x_{ij} \quad j = 1, \ldots, J, r = 1, \ldots, R$$

där

$$x_{ij} = \begin{cases} P(\{\mathcal{O}_1|s_{r3}\}_1)_i, & \text{om } sl_j \in \{s_{i3}\}_2 \\ 0, & \text{om } sl_j \notin \{s_{i3}\}_2 \end{cases} \text{ och } P(\mathcal{O}_1|\{s_{r3}\}_1) = \pi b_{\{s_{3r}\}_1\mathcal{O}_0} a_{\{s_{r2}\}_1\{s_{r3}\}_1} b_{\{s_{3r}\}_1\mathcal{O}_1}$$

$$j = 1, \ldots, J, r = 1, \ldots, R.$$

Därefter kommer observationer fortsätta mottas tills det att en observation inte är inom samma segmentet se^* som tidigare. När observationen sedan lämnar segmentet och $P(\mathcal{O}_v|se^*) = 0$ där $2 \leq v \in \mathbb{N}_1$ kommer programmet som tidigare se om observationen existerar inom någon av segmenten $se_{1,\ldots,L}$. Om den gör det, kommer den fortsätta till nästa kolumn i tillståndsmatrisen S och återupprepa samma procedur beskriven ovan. Detta kommer fortgå ända fram till att slutmålet för resan nås. Antalet möjliga slutmål minskar för varje segmentsbyte som sker, då rutter som inte delar första variabeln i tillståndet, det vill säga segmentet, avlägsnas som ett möjligt slutmål. Prediktionen kommer med detta att bibehålla eller öka i precision för varje segmentbyte som sker. Generellt beräknat som en dold Markovmodell sker följande

$$P(\mathcal{O}_v|\mathbf{SL}) = P(\mathcal{O}_v|\mathbf{SL})_j = \sum_{i=1}^{R} x_i \quad j = 1, \ldots, J, r = 1, \ldots, R$$

där

$$x_i = \begin{cases} P(\{\mathcal{O}_v|s_{r(v+2)}\}_1)_i, & \text{om } sl_j \in \{s_{i(v+2)}\}_2 \\ 0, & \text{om } sl_j \notin \{s_{i(v+2)}\}_2 \end{cases}$$

och

$$P(\mathcal{O}_1|\{s_{r(v+2}\}_1) = \pi b_{\{s_{r2}\}_1\mathcal{O}_0} \prod_{n=1}^{v} a_{\{s_{r(n+1)}\}_1\{s_{r(n+2)}\}_1} b_{\{s_{(n+2)r}\}_1\mathcal{O}_n} \quad j = 1, \ldots, J, r = 1, \ldots, R.$$

Värt att påpeka är att om en GPS-observation $\mathcal{O}_v$ inte befinner sig inom området för ett segment vid ett segmentsbyte mellan två tillstånd kommer predikteringen att avbrytas då modellen saknar information för att klara en prediktering, varav $P(\mathcal{O}_v|\mathbf{SL}) = 0$. Vid vissa fall kan det förekomma brus i GPS-signalerna som medför att en punkt ligger utanför ett segment, trots att föraren befinner sig i segmentet. Vid detta fall kommer predikteringen att avbrytas eftersom observationen inte ligger inom arean för något segment. Hela processen för prediktering kan tydligt överskådas i flödesschemat i *Figur 3.11*.

För att förtydliga predikteringsdelen följer ett exempel utifrån modellen som kan ses i *Figur 3.12*. Låt oss säga att en förare startar sitt fordon och därmed den inbyggda GPS:en. En första GPS-observation $\mathcal{O}$ går in i plattformen, varav programmet tar observationen och ser om den finns inom någon av areorna utifrån GPS-punkterna i startmålsvektorn $\mathbf{ST}$, *Figur 3.13*. Programmet svarar att den existerar inom startmålet 1, p_1, där den vänder sig till tillståndsmatrisen $\mathbf{S}$, *Figur 3.16*, med informationen. Där går den igenom den andra raden , dvs alla rutters första tillstånd, och noterar vilka tillstånd som innehåller samma startmål $\{s_{(1,\ldots,4)2}\}_1 = 1$, vilket är alla i det här fallet. Redan nu ger plattformen en första prediktion, där den går igenom första raden $s_{(1,\ldots,4)1}$ för de rutter som hade korrekt startmål, vilket var alla, och delar antalet gånger rutten färdats till ett visst slutmål $T^{sl_{1,\ldots,4}}$ med det totala antalet för alla rutter med startmålet st_1, T_{st_1}. Vilket ger följande prediktion

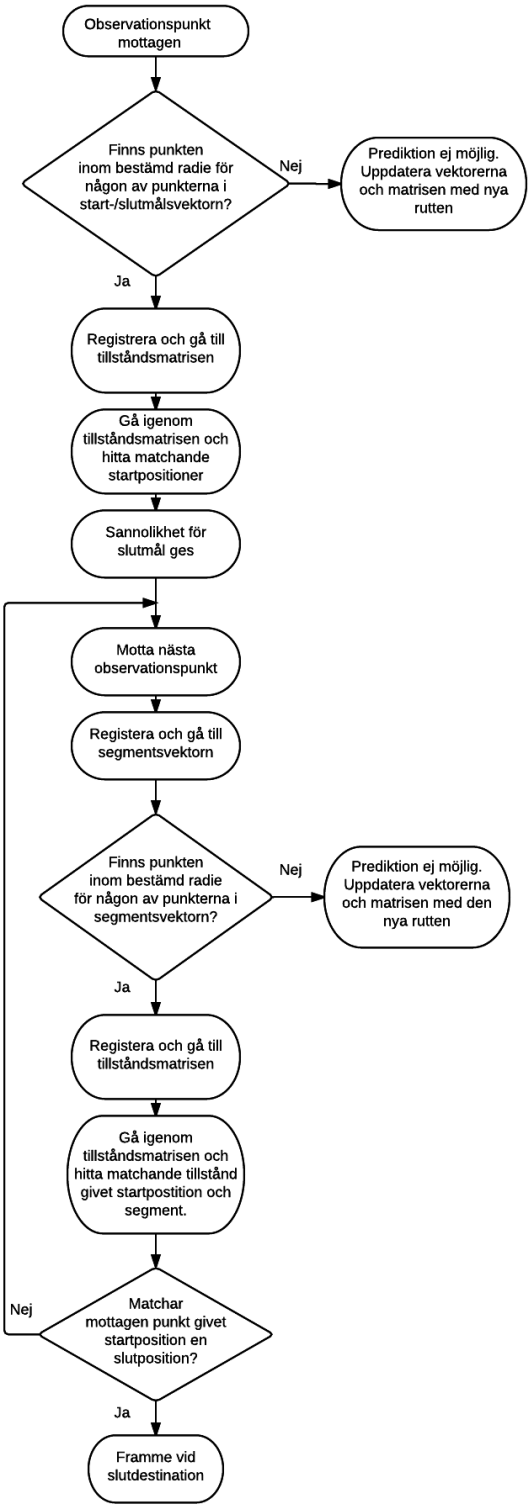
$$P(\mathcal{O}_0|\mathbf{SL}) = \begin{bmatrix} \frac{T_{st_1}^{sl_1}}{T_{st_1}} \\ \vdots \\ \frac{T_{st_1}^{sl_4}}{T_{st_1}} \end{bmatrix} = \begin{matrix} \textit{Slutmål} \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \quad \begin{bmatrix} \frac{2}{10} \\ \frac{4}{10} \\ \frac{1}{10} \\ \frac{3}{10} \end{bmatrix}$$

Detta kan även beräknas som en dold Markovmodell där den stationära fördelningen är

$$\pi = \begin{bmatrix} \frac{s_{11}}{T} \\ \vdots \\ \frac{s_{41}}{T} \end{bmatrix} = \begin{bmatrix} \frac{2}{10} \\ \frac{4}{10} \\ \frac{1}{10} \\ \frac{3}{10} \end{bmatrix}$$

där $T = \sum_{i=1}^{4} s_{i1}$ är det totala antal alla rutter färdats. Observationssannolikheterna för $\mathcal{O}_0$ beräknas

$$b_{\{s_{r2}\}_1\mathcal{O}_0} = P(\mathcal{O}_0|\{s_{r2}\}_1)_r = \begin{Bmatrix} 1, & \text{om } \mathcal{O}_0 \in \{s_{r2}\}_1 \\ 0, & \text{om } \mathcal{O}_0 \notin \{s_{r2}\}_1 \end{Bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{för } r = 1, \ldots, 4$$



Figur 3.11: Flödesschema för prediktering. Författarnas egna bild.

för att därefter bestämma sannoliheterna för de olika slutmålen i **SL**

$$P(\mathcal{O}_0|\mathbf{SL}) = P(\mathcal{O}_0|\mathbf{SL})_j = \sum_{i=1}^{R} x_{ij} = \begin{bmatrix} \frac{2}{10} \\ \frac{4}{10} \\ \frac{1}{10} \\ \frac{3}{10} \end{bmatrix} \quad j = 1, \ldots, 4, r = 1, \ldots, 4$$

där

$$x_{ij} = \begin{cases} P(\{\mathcal{O}_0|s_{r2}\}_1)_i, & \text{om } sl_j \in \{s_{i2}\}_2 \\ 0, & \text{om } sl_j \notin \{s_{i2}\}_2 \end{cases} \text{ och } P(\mathcal{O}_0|\{s_{r2}\}_1) = \pi b_{\{s_{r2}\}_1\mathcal{O}_0} = \begin{bmatrix} \frac{2}{10} \\ \frac{4}{10} \\ \frac{1}{10} \\ \frac{3}{10} \end{bmatrix}.$$

Då $P(\mathcal{O}_0|\mathbf{SL})$ i det här fallet redan är en stokastisk matris finns inget behov i användandet av en normalisering.

Plattformen mottar en ny observation $\mathcal{O}_1$ i och med att fordonet färdas och plattformen undersöker om den tillhör något segment i **SE**, *Figur 3.15* vilket den i vårt fall gör i form av segmentet A. Dock ger detta inget i syfte av predikteringen, då alla rutter passerar segmentet vilket innebär att sannolikheterna för de olika slutmålen ej förändras. Låt oss istället undersöka segmentbytet $B \rightarrow D$, där observationen mottagen är $\mathcal{O}_D$ och den innan var $\mathcal{O}_B$.

$$P(\mathcal{O}_D|\mathbf{SL}) = P(\mathcal{O}_{1,\ldots,B}|\mathbf{SL})^T \begin{bmatrix} \frac{T_{se_3}^{sl_1}}{T_{se_3}} & \ldots & \frac{T_{se_3}^{sl_4}}{T_{se_3}} \end{bmatrix} = \begin{bmatrix} \frac{2}{10} \\ \frac{4}{10} \\ \frac{1}{10} \\ \frac{3}{10} \end{bmatrix} \begin{bmatrix} \frac{0}{8} & \frac{4}{8} & \frac{1}{8} & \frac{3}{8} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{4}{80} \\ \frac{1}{80} \\ \frac{3}{80} \end{bmatrix}$$

vilket multipliceras med normaliseringskonsanten $\frac{1}{1-\sum_{i=1}^{J} P(\mathcal{O}_D|\mathbf{SL})_i} = \frac{1}{1-\frac{8}{80}}$ som ger

$$P(\mathcal{O}_D|\mathbf{SL}) = \begin{bmatrix} 0 \\ \frac{4}{8} \\ \frac{1}{8} \\ \frac{3}{8} \end{bmatrix}.$$

Om infallsvinkeln istället ändras till en dold Markovmodell sker beräkningarna som följande

$$b_{\{s_{(D+1)r}\}_1\mathcal{O}_D} = P(\mathcal{O}_1|\{s_{r(D+1)}\}_1)_r = \begin{Bmatrix} 1, & \text{om } \mathcal{O}_D \in \{s_{r(D+1)}\}_1 \\ 0, & \text{om } \mathcal{O}_D \notin \{s_{r(D+1)}\}_1 \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \quad \text{för } r = 1, \ldots, 4.$$

I samband med detta kommer även övergånsmatrisen **A** beräknas som

$$a_{\{s_{rD}\}_1\{s_{r(D+1)}\}_1} = P(\{s_{rD}\}_1|\{s_{r(D+1)}\}_1) = 1 \quad r = 1, \ldots, 4$$

$$P(\mathcal{O}_D|\mathbf{SL}) = P(\mathcal{O}_0|\mathbf{SL})_j = \sum_{i=1}^{4} x_{ij} = \begin{bmatrix} 0 \\ \frac{4}{10} \\ \frac{1}{10} \\ \frac{3}{10} \end{bmatrix} \quad j = 1, \ldots, 4, r = 1, \ldots, 4$$

där

$$x_{ij} = \begin{cases} P(\{\mathcal{O}_D|s_{r(D+1)}\}_1)_i, & \text{om } sl_j \in \{s_{i(D+1)}\}_2 \\ 0, & \text{om } sl_j \notin \{s_{i(D+1)}\}_2 \end{cases}$$

och

$$P(\mathcal{O}_D|\{s_{r(D+1)}\}_1) = \pi b_{\{s_{r2}\}_1 \mathcal{O}_0} \prod_{n=1}^{D} a_{\{s_{r(n+1)}\}_1 \{s_{r(n+2)}\}_1} \{s_{(n+2)r}\}_1 \mathcal{O}_n = \begin{bmatrix} \frac{2}{10} \\ \frac{4}{10} \\ \frac{1}{10} \\ \frac{3}{10} \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{4}{10} \\ \frac{1}{10} \\ \frac{3}{10} \end{bmatrix}.$$

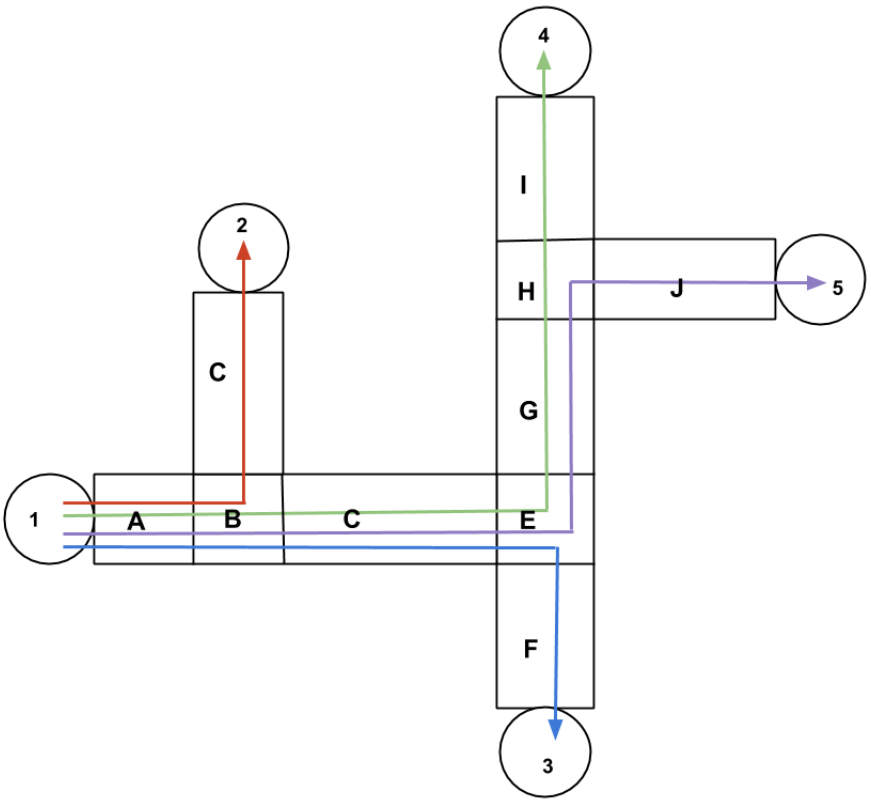
$\mathcal{O}_D|\mathbf{SL}$) är som i det tidigare fallet i behov av att multipliseras med en normaliseringskonstant, vilket är $\frac{1}{1-\sum_{i=1}^{J} P(\mathcal{O}_D|\mathbf{SL})_i} = \frac{1}{1-\frac{8}{10}}$ som ger

$$P(\mathcal{O}_D|\mathbf{SL}) = \begin{bmatrix} 0 \\ \frac{4}{8} \\ \frac{1}{8} \\ \frac{3}{8} \end{bmatrix}.$$

I och med avlägsnandet av möjliga slutmål ökar precisionen av prediktionen, vilket innebär att en säkrare prediktion innehas desto längre föraren färdas. Plattformen fortsätter att arbeta på samma sätt tills en observationen inte längre hamnar inom någon av segmenten i tillstånden av de rutter som hittills uppfyllt alla tidigare tillstånd. När observationen gör det, ska den först se om den finns inom någon av areorna i slutmålsevektorn **SL**. Om observationen är det, och det slutmålet tillhör det sista tillståndet i rutten, är sannolikheten $\frac{1}{1}$ att förarens slutdestinationen är där. Om den inte är det, kan ingen vidare prediktion utföras på grund av att modellen saknar information för vidare prediktering.

3.6 Utvärdering av modellen

För att få en tydlig bild över hur korrekt modellen predikterar används samma metod som beskrivits av Alvarez-Garcia et al (2010). I det fallet genereras resultat genom att ett visst antal rutter körs i modellen för att prediktera vart dess slutmål är. Slutmålen är kända sedan tidigare, varav de efter att 25 %, 50 %, 75 % och 90 % av rutten har körts noterat sannolikheten för att korrekt slutdestination är slutmålet. Som syns i Alvarez-Garcia et al (2010) resultat, *Figur 3.17*, kommer sannolikheten öka för att rätt slutmål predikteras allt eftersom föraren närmar sig sitt



Figur 3.12: En modell skapad av fyra rutter med start-/slutmål samt segment utritade. Författarnas egna bild.

$$\mathbf{ST} = \begin{matrix} \textit{Namn} \\ 1 \end{matrix} \begin{bmatrix} \textit{GPS-punkt för startmål} \\ p_1 \end{bmatrix} \quad \text{där } p_1 = (lat_1, long_1)$$

Figur 3.13: Startmålsvektorn **ST** utifrån *Figur 3.12*.

slutmål. Värt att nämnas är att användare 1,4 och 6 endast färdats med bil medan de resterande användarna även har färdats med cykel och/eller till fots. Användare 1 har kört 18 gånger till 3 olika destinationer, användare 4 har kört 81 gånger till 4 olika destinationer och användare 6 har kört 231 rutter till 11 olika destinationer.

Modellen kommer byggas upp på 198 stycken rutter. På grund av att modellen byggs upp av väldigt få rutter till många olika slutmål, kommer 20 av de rutter som modellen byggs upp av slumpmässigt tas ut och användas som testrutter. För att beräkna sannolikheten för att slutmålet efter 25%, 50%, 75% och 90% kommer den procentsatsen beräknas för de antal GPS-punkter som bygger upp rutten och sedan predikteras. Antalet punkter efter procentberäkningen kommer alltid att avrundas nedåt. Resultatet kommer därefter att genereras som beskrivits ovan, det vill säga med samma metod som Alvarez-Garcia et al (2010) använde, för att därefter jämföras med deras resultat.

$$\mathbf{SL} = \begin{matrix} Namn \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \begin{matrix} GPS\text{-}punkt\ för\ slutmål \\ \left[\begin{matrix} p_2 \\ p_3 \\ p_4 \\ p_5 \end{matrix}\right] \end{matrix} \quad \text{där } p_2 = (lat_2, long_2)$$

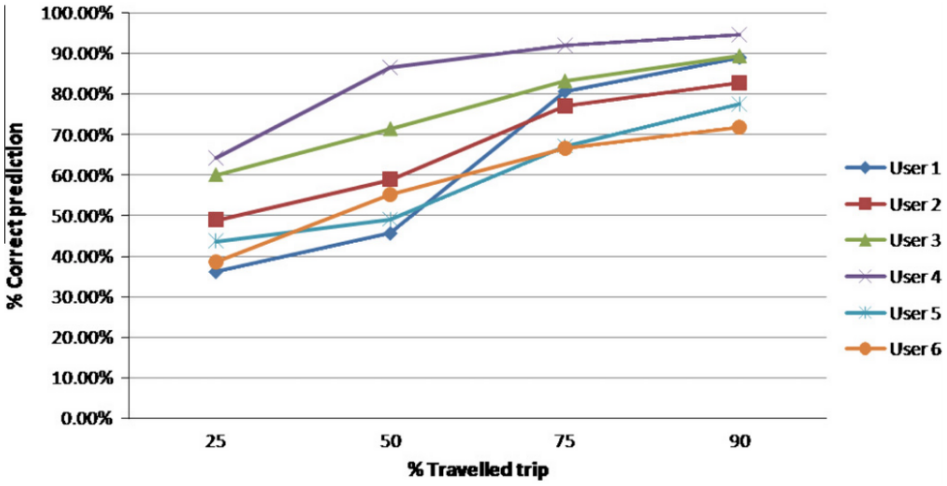
Figur 3.14: Slutmålsvektorn **SL** utifrån *Figur 3.12*.

$$\mathbf{SE} = \begin{matrix} Namn \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \end{matrix} \begin{matrix} Area\ i\ GPS\text{-}punkter \\ \left[\begin{matrix} p_{A_1}, \ldots, p_{A_i} \\ p_{B_1}, \ldots, p_{B_j} \\ p_{C_1}, \ldots, p_{C_k} \\ p_{D_1}, \ldots, p_{D_l} \\ p_{E_1}, \ldots, p_{E_m} \\ p_{F_1}, \ldots, p_{F_n} \\ p_{G_1}, \ldots, p_{G_o} \\ p_{H_1}, \ldots, p_{H_p} \\ p_{I_1}, \ldots, p_{I_q} \\ p_{J_1}, \ldots, p_{J_r} \end{matrix}\right] \end{matrix} \quad \text{där } p_{A_1} = (lat_{A_1}, long_{A_1}) \text{ och } i, j, k, l, m, n, o, p, q, r \in \mathbb{N}_1$$

Figur 3.15: Segmentmatrisen **SE** utifrån *Figur 3.12*.

$$\mathbf{S} = \begin{matrix} Rutt \\ X1 \\ X2 \\ X3 \\ X4 \end{matrix} \begin{matrix} Antal & Tillstånd & & & & & & & & \\ \left[\begin{matrix} 2 & (1,2) & (A,2) & (B,2) & (C,2) & (2,2) & & & & \\ 4 & (1,3) & (A,3) & (B,3) & (D,3) & (E,3) & (F,3) & (3,3) & & \\ 1 & (1,4) & (A,4) & (B,4) & (D,4) & (E,4) & (G,4) & (H,4) & (I,4) & (4,4) \\ 3 & (1,5) & (A,5) & (B,5) & (D,5) & (E,5) & (G,5) & (H,5) & (J,5) & (5,5) \end{matrix}\right] \end{matrix}$$

Figur 3.16: Tillståndsmatrisen **S** utifrån *Figur 3.12*.



Figur 3.17: Resultat för korrekt prediktion av slutmål för Alvarez-Garcia et al. plattform (s.5, Alvarez-Garcia et al., 2010). Återgiven med tillstånd.

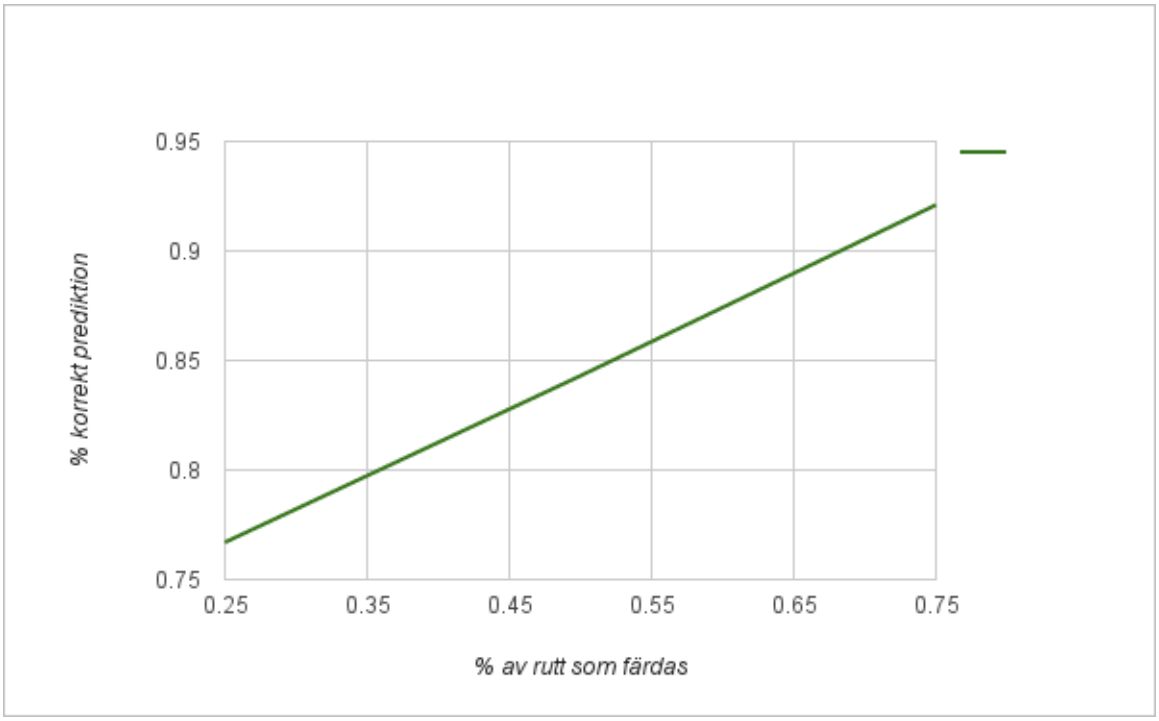
4
Resultat

Programet har körts med 20 olika rutter och resultaten presenteras i *Tabell 4.1*. För varje rutt testas det med hur stor sannolikhet plattformen predikterar rätt slutmål efter 25 %, 50 % och 75 % av rutten. Värt att notera i *Tabell 4.1* är att många rutter kan predikteras med 100 % sannolikhet redan efter 25 %. Sedan har genomsnittet för dessa lagts in i *Figur 4.1*. Grafen visar att redan efter 25 % av rutten predikteras rätt mål med över 75 % sannolikhet och att efter 75 % av rutten predikteras rätt mål med över 90 % sannolikhet.

Programmet har körts upp till 90 % av rutten för de 20 rutterna, men har då i de flesta fallen misslyckats med prediktionen. Vad det beror på diskuteras i avsnitt 5.

25 %	50 %	75 %
0.33	1	1
0.2	0.25	1
1	1	1
1	1	1
0.5	0.5	0.6667
0.7143	0.7143	1
1	1	1
1	1	1
0.6667	0.6667	1
0.2	1	1
0.0909	0.0909	0.1111
1	1	1
0.8	0.8	0.8
1	1	1
1	1	1
1	1	1
1	1	1
1	1	1
0.833	0.833	1
1	1	1

Tabell 4.1: Sannolikhet för korrekt slutmål efter att 25%, 50% och 75% av rutten körts.



Figur 4.1: Grafen visar medelvärdet av det procentuella värdet för korrekt prediktion för när en bil har färdats 25-75 % av en rutt. Resultatet är baserat på 20 ruttter. Författarnas egna bild.

5
Diskussion

I detta kapitel kommer vi börja med att diskutera det resultat kandidatarbetet mynnat ut i, där en viktig del är att jämföra det resultatet som vi åstadkommit med resultatet från Alvarez-Garcia et al. Vidare diskuteras det hur effektivt datamodellen fungerar och vad som hade kunnat förändrats till det bättre. Därefter sker samma procedur för den matematiska modell som applicerats och till sist ges förslag på hur plattformen kan förbättras och vidareutvecklas.

5.1 Resultatdiskussion

Vi har en nästan fungerande modell. I de flesta fallen predikterar den upp till över 75 % av rutten, men efter det misslyckas den. Det är därför vi i resultatet inte predikterar hela vägen upp till 90 %. Detta kan bero på en rad av orsaker som diskuteras i utvärderingarna av modellerna.

Resultatet för de 20 slumpmässigt utvalda rutter som gick att prediktera är väldigt bra. Vid 75 % av utförd rutt visar 17 av de 20 testrutterna ett resultat på 100 %. I grafen i *figur 4.1* kan man tydligt se hur precisionen ökar ju närmre man når sitt slutmål. På grund av att vi inte kan prediktera ända upp till 100 % av färdad rutt kan man heller inte se den procentuella ökningen för korrekt prediktion ända fram till 100 % av färdad rutt. Skulle man optimera koden i modellen utefter förslagen som ges i avsnitt 5.4 hade man troligtvis kunna se ett resultat som vid 100 % av en färdad rutt troligtvis närmar sig 100 % korrekt prediktion.

I *Tabell 4.1* kan det ses att många rutter kunde predikteras med 100 % sannolikhet väldigt tidigt in i rutten. Detta är en konsekvens av att modellen är uppbyggd av väldigt få rutter och att många av dem är helt unika från de andra. Detta kommer försvåra utvärderingen av den matematiska delen av arbetet.

5.1.1 Jämförelse med Alvarez-Garcia et al.

Vid jämförelsen mellan vårt resultat och Alvares-Garcia et als bör man ha i åtanke att måtten som används är väldigt data- och situationsberoende. Detta innebär att om modellen endast består av två rutter som körs mellan två olika start-/slutdestinationer som ligger på helt skilda håll från varandra är en korrekt prediktering väldigt simpel att utföra.

För de 20 rutter i vår modell som klarade predikteringen togs ett medelvärde av det procentuella värdet för korrekt prediktion. För 25 % av gången rutt gavs resultatet 67 % korrekt prediktion. Detta kan jämföras med den användare med bäst resultat efter 25 % av gången rutt i Alvarez-Garcia et als, då procentuell korrekt prediktion låg på cirka 63 %. Samma rutt har efter 76 % av resan en procentuell korrekt prediktion på cirka 92 %, vilket är samma resultat som medelvärdet ger i vår modell. De övriga resultat som visas för Alvarez-Garcia et al är betydligt sämre än de resultat vår plattform predikterar. Eftersom vår modell inte kan prediktera samtliga rutter är det svårt att göra en rättvis jämförelse. Dock visar resultatet för de rutter som går att prediktera ett lovande resultat, där resultatet är minst lika bra som resultatet i Alvarez-Garcia et als fall, om inte bättre. Skulle vår modell vidareutvecklas och förbättras enligt förslagen som ges under avsnitt 5.4 skulle troligen modellen fungera korrekt för samtliga rutter och man skulle därmed kunna göra en mer korrekt jämförelse. Av samma anledning är det även svårt att dra slutsatser om skapande av segment eller att använda stödpunkter är den metod som ger bäst resultat. I avsnitt 5.2.3 kommer fördelarna med att använda stödpunkter diskuteras.

5.2 Utvärdering av datamodellen

Efter att modellen körts igenom konstaterades det att vi hade till stor del en fungerande datamodell. Den skapar startmål, slutmål, segment, tillstånd och rutter. Dock misslyckas predikteringsdelen mot slutet, och i vissa fall går det inte att prediktera alls. Anledningent till att vår modell inte kan prediktera alla rutter fullt ut skulle kunna vara att arean för buffertzonen som bygger upp segmenten är för liten och det skapar ett glapp mellan två segment, och programmet kommer därmed att avbrytas när en rutt lämnar ett segment. Hur segmenten kunnat förändras diskuteras i stycket 5.2.2.

Tanken vid projektets start var att lägga undan en del av de rutter vi hade erhållit från Volvo Personvagnar och använda som testrutter. Olyckligtvis var antalet rutter efter filtreringen inte tillräckligt många för att bygga upp den storlek på modellen som var önskvärt. Med detta i åtanke valde vi istället att använda rutter som byggt upp modellen för att få reslutat. Då modellen ändå inte kan eller ska ha möjlighet att prediktera rutter som aldrig tidigare körts behöver dessa ej testas. Den låga mängden rutter betydde även att idéer som vi först haft i åtanke att applicera i modellen ej gick att genomföra

5.2.1 Datamängd

Det är svårt att fullständigt utvärdera modellen då antalet rutter för uppbyggandet av modellen ej varit av en önskvärd storlek. Efter filtrering erhölls 198 korrekta rutter att bygga upp modellen med, detta kan jämföras med modellen av Froehlich & Krumm som efter filtrering byggdes upp av närmare 14 500 rutter. Hade mängden data i form av rutter varit i en större storlek än de 198 arbetsbara hade möjligheten funnits för att bygga upp en mer exakt modell. Ett annat problem med den låga datamängden är att de flesta rutter enbart körts en gång. (Eftersom modellen vi har

byggt upp avbryts när den inte hittar segment som matchar den mottagna GPS-punkten är den matematiska modellen svårutvärderad. Programmet lyckas hitta de möjliga rutter och slutmål som finns, men sannolikheterna för dessa blir ofta identiska då rutterna bara körts en gång var. Med hjälp av den erhållna datan kan vi lätt se att datamodellen lyckas ta fram de möjliga rutterna och slutmålen.

5.2.2 Antal GPS-punkter och storlek på segment

Genom att ändra avståndet mellan punkterna i modellen till antingen fler eller färre punkter kan datamängden ökas och minskas. Dock får inte avstånden mellan punkterna vara allt för stor, då detta skulle innebära att punkter som exempelvis ligger i en sväng försvinner och rutten får en helt annan form vid segmentering av rutten.

Genom att göra segmenten större kan ruttdelar som i annat fall ses som olika segment nu ses som samma segment. Detta kan leda till att rutter som annars anses olika grupperas som en enda rutt. Detta är positivt om inte segmenten blir för stora och får med GPS-punkter som ligger på en parallell väg.

Vid testkörning med en väldigt liten storlek på segmenten fann vi att rutterna blev unika då ingen av de 198 rutterna som programmet arbetade med ansågs vara samma rutt. På samma sätt kunde vi se att stora segment grupperade flera rutter under vissa geografiska avsnitt där de låg nära varandra, men att rutternas differenser inte registrerades - vilket i sin tur gav oss en oanvändbar tillståndsmodell. Svårigheten har legat i att hitta en balans där man i valet av storlek på segmentet bibehåller ruttens unika delar samtidigt som vi lyckas gruppera gemensamma delar för att få en snabbare och mindre störningskänslig modell.

5.2.3 Stödpunkter

Vid valet av att använda stödpunkter eller koppla punkterna till segment beslutades det att segment var det mest välanpassade alternativ till vår modell. Dock finns det givetvis fördelar med att använda stödpunkter. Vår plattform är relativt långsam med att bygga upp modellen på grund av den stora mängden data som måste sparas och genomgås. Detta hade kunnat minskas med hjälp av stödpunkter istället för att spara hela rutter och segment. Trots detta är vi nöjda med valet, då vi kunnat lägga mer fokus på att utveckla den matematiska modellen. Dessutom påverkar inte tiden för att bygga upp modellen hur snabb predikteringen arbetar. Huvudsaken är att själva predikteringen är effektiv då modellen måste hinna bearbeta den mängd GPS-punkter som plattformen mottar under bilens färd. Prediktionsprogrammet hade även varit effektivare tidsmässigt och tagit mindre plats med stödpunkter. Ifall modellen exempelvis hade applicerats i en smartphone skulle detta varit önskvärt, men då detta är en plattform för vidareutveckling anser vi att prediktionen är tillräckligt snabb.

5.2.4 Tidsaspekt

Tanken vid projektets start var att tillstånden skulle innehålla en tidsaspekt, där det togs hänsyn till om rutten körts under förmiddag, eftermiddag eller kväll/natt. Detta hade kunnat bidra till att förbättra prediktionen eftersom vissa rutter endast körts under vissa timmar på dygnet, exempelvis att de flesta åker till jobbet under förmiddagen. Genom att ha tidsaspekten som en parameter i tillstånden hade programmet kunnat avlägsna vissa slutmål i avseende vilken tid på dygnet som rutten körs. Dock möttes vi av problemet att antalet tillhandahållna rutter inte var i storlek nog för att dra nytta av tidsparametertillägget. Varav tidsaspekten inte längre ansågs relevant att använda som en del av ett tillstånd. Dock är denna modell uppbyggd på ett sätt som gör det lättapplicerbart att lägga till fler parametrar i tillstånden, och skulle fler rutter tillkomma kan tidsaspekten tillämpas för att öka precisionen. Oavsett om programmet använder sig av en tidsaspekt eller inte kommer vi att se att precisionen ökar när föraren närmar sig sitt slutmål. Det beror på att antal möjliga slutdestinationer minskar alltnärmre föraren färdas sin slutdestination

5.3 Utvärdering av den matematiska modellen

Den matematiska modellen är lite svårutvärderad på grund av att modellen inte är uppbyggd på tillräckligt många rutter och att datamodellen inte fungerar fullständigt som den ska. Den beräknar korrekta sannolikheter när datamodellen ger den rätt observationer och inte eliminerar rutter som inte bör elimineras.

5.3.1 HMM

Den modell som skapats under kandidarbetets gång är på många sätt lik en HMM där skillnaderna till stor del består av definitionen av de applicerade matriserna. Då vår tillståndsmatris **S** innehåller information över hur alla rutter körts med de passerade tillstånden i kronlogisk ordning samt antalet rutterna körts i första kolonnen. Det kan ses som en sammanslagning av övergångsmatrisen **A**, den stationära fördelningen π samt alla segment **S** i en traditionell dold Markovmodell. Den observationsmatrisen **B** som vanligtvis används i en HMM kan i vårt fall beskrivas som applicerandet av de erhållna GPS-observationerna **O** i startmålsvektorn **ST**, slutmålsvektorn **SL** och segmentmatrisen **SE**. I och med det val applicerande som gjorts, där startpunkten och förarens val av rutt tar en mer betydande roll än med en vanligt dold Markovmodell vilket innebär att predikteringen ej kommer vara identisk med applicerandet av typisk HMM. Dock kan även vår modell ses som en HMM, trots att den ej är identisk med en. Detta kan ses i kapitel 3.5 om den matematiska modellen där **A** och **B** skiljer sig mot de i en HMM då de ej är stokastiska matriser. Det går även väl att se ovan beskrivna skillnader och likheter om man jämför kapitel 2.3 om Markovmodeller med kapitel 3.5.

Vidare skillnader med den matematiska modellen som applicerats i jämförelse mot en vanlig HMM är att vår tillståndsmatris är ruttbaserad vilket innebär att vår modell avlägsnar samtliga rutter och dess slutmål som skiljer sig från den rutt som

färdas som möjliga slutmål. Detta innebär att en rutt och dess slutmål aldrig har en möjlighet att återkomma som möjligt alternativ i prediktionen, även om föraren passerar ett tillstånd som finns i den avlägsnade rutten. Det påverkar inte vilka slutmål som är möjliga men prediktionen för att hamna i någon av dessa. Om exempelvis två rutter som startar vid olika startpunkter, men har samma slutmål, sammanfogas efter halva färdens kommer de därefter ha en eller flera gemensamma tillstånd. En HMM är bara intresserad av vilka tillstånd den kan vara i och vilka som kan komma härnäst, detta leder till att HMM:en tar hänsyn till bägge rutterna och på så vis ökar sannolikheten till deras gemensamma slutmål. Anledningen till att vi valt att inte använda en vanlig HMM är för att vi anser att sannolikheten för till exempel att du är på väg hem från jobbet ej ska påverkas av det antal du åkt från en matbutik hem. Tyvärr skulle vi behöva testa med samma data både på vår modell och på en modell uppbyggd av en HMM för att undersöka vilket som ger bäst resultat.

5.4 Vidareutveckling och förbättringar

Denna modell är på många vis en väl fungerande modell, dock finns det en del förbättringar som kan genomföras och saker som bör tänkas på vid EN vidareutveckling av modellen. Flera av dessa punkter är saker vi diskuterat, men begränsat bort.

Först och främst skulle modellen kunna byggas upp och bli större och mer givande om antalet rutter hade av en större storlek. Antaligen hade det då funnits rutter som enligt modellen skulle varit identiska och därmed hade den matematiska modellen kunnat beräkna ut sannolikheter baserade på antal gånger rutten hade körts. Nu fanns det som tidigare nämnts inga rutter som ansågs vara lika och en sannolikhet för att nå ett mål som föraren inte varit vid kunde ej beräknas.

Hade fler rutter erhållits och byggt upp kartan hade det eventuellt behövts ses över hur data i form av GPS-punkter sparas. Redan vid de 198 rutter vi nu hade efter filtreringen som byggde vår modell blev tungt för datorn att modellera och det gick väldigt långsamt på grund av mängden data. En förbättring utan att implementera stödpunkter och behålla de segment som vi använder skulle varit att minska antalet sparade punkter per rutt. Just nu sparar vi en stor mängd punkter för att kunna vara så precis som möjligt, detta kan minskas om ett behov av att effektivisera programmet existerar. Ett annat alternativ att betänka vid behov av effektivisering är att optimera koden ytterligare vilket inte har utförts till fullo på grund av tidsbrist.

Det finns eventuellt ett behov att förbättra programmet i och med att en observation i form av en GPS-punkt ej ses tillhöra ett segment då dess punkt ligger utanför ett segments area vilket är uppbyggd från segmentmatrisens SE GPS-punkter. Anledningen är att brus kan förekomma bland GPS-signalerna som gör att en punkt hamnar utanför ett segmentet som den egentligen tillhör, för att den med nästkommande punkt ligger inom segmentet. En förbättring skulle kunna vara att programmet gör

ett undantag i form av att den väntar och noterar om nästkommande punkt ligger inom segmentet. Om den gör det kan programmet anta att det endast är störningar i GPS-signalen. Då vi efter filtreringen har punkter med ett avstånd på 30 m från varandra är det inte troligt att fordonet ska ha kört en annan väg och därefter återkommit på vägen den tidigare kört.

Om rutten som predikteras är en ny rutt, det vill säga att rutten avviker från samtliga tidigare rutter avslutas prediktionen i denna modell. En vidareutveckling av modellen skulle kunna vara att hantera denna situation på ett bättre sätt. Exempelvis skulle programmet då bilen lämnar den sista möjliga rutten se om bilen trots allt fortfarande är i inom ett existerande segment i $\mathbf{S}$. I sådana fall skulle prediktionen startat om från det segmentet. Ytterligare en funktion som hade kunnat implementeras är om rutten inte startar i ett startmål, men att observationen finns inom ett segment, varav den hade satt det segmentet som startmål och påbörjat dess prediktering över möjliga slutmål därifrån.

Litteraturförteckning

[1] Alvarez-Garcia, J.A., Ortega, J.A., Gonzalez-Abril, L. & Velasco, F. (2010). Trip destination prediction based on past GPS log using a Hidden Markov Model. *Expert Systems with Applications.*

[2] Duda, R. O. Hart, P.E. Stork & D. G. (2001). *Pattern Classification second edition.* New York: Wiley Cop.

[3] Ecma International. (2013). *The JSON Data Interchange Format.* Geneva

[4] Froehlich, J. & Krumm, J. (2008). Route Prediction from Trip Observations. *Society of automotive engineers (SAE) world congress.*

[5] Krumm, J. (2008). A Markov model for driver turn prediction. *Society of automotive engineers (SAE) world congress.*

[6] Lantmäteriet. (u,å). *Felkällor vid GNSS-mätning* Hämtad 2015-05-15, från https://www.lantmateriet.se/sv/Kartor-och-geografisk-information/GPS-och-geodetisk-matning/GPS-och-satellitpositionering/Metoder-for-GNSS-matning/Felkallor-vid-GNSS-matning/

[7] Lantmäteriet. (u,å). *GPS* Hämtad 2015-05-15, från https://www.lantmateriet.se/sv/Kartor-och-geografisk-information/GPS-och-geodetisk-matning/GPS-och-satellitpositionering/GPS-och-andra-GNSS/GPS/

[8] Lantmäteriet. (u,å). *GPS och satellitpositionering* Hämtad 2015-05-15, från https://www.lantmateriet.se/sv/Kartor-och-geografisk-information/GPS-och-geodetisk-matning/GPS-och-satellitpositionering/

[9] Nagaraj, U. & N. Kadam, N. (2011). Study Of Statistical Models For Route Prediction Algorithms In VANET. *The International Institute for Science, Technology and Education (IISTE), Vol 1, No.4.*

[10] Naturvårdsverket. (2015). *Utsläpp av växthusgaser från inrikes transporter 1990-2013.* Hämtad 2015-05-06, från https://www.naturvardsverket.se/Samar-miljon/Statistik-A-O/Vaxthusgaser-utslapp-fran-inrikes-transporter/?visuallyDisabledSeries=0

[11] Simmons, R., Browning, B., Zhang, Y & Sadekar, V. (2006). Learning to Predict Driver Route and Destination Intent. *Institute of Electrical and Electronics Engineers (IEEE) Intelligent Transportation Systems Conference.*

[12] Stamp, M. (2012). A Revealing Introduction to Hidden Markov Models.

[13] The MathWorks, Inc. (1994-2015). *Matlab Overview.* Hämtad 2015-05-05, från http://se.mathworks.com/products/matlab/?refresh=true

[14] Trafikverket. (2014). *Transportsektorns utsläpp.* Hämtad 2015-05-06, från http://www.trafikverket.se/Privat/Miljo-och-halsa/Klimat/Transportsektorns-utslapp/

[15] Waze Mobile. (2015) Hämtad 2015-05-06, från https://www.waze.com/

Bildförteckning

[1] Alvarez-Garcia, J.A., Ortega, J.A., Gonzalez-Abril, L. & Velasco, F. (2010). Trip destination prediction based on past GPS log using a Hidden Markov Model. *Expert Systems with Applications.*