Technical University of Denmark

Analyse og prædiktion af menneskelige bevægelsesmønstre på baggrund af spatiotemporal data

Analysis and prediction of human mobility based on spatiotemporal data

Christian Raaby

Master thesis, 30-09-2011

DTU Informatik
Institut for Informatik og Matematisk Modellering

Analyse og prædiktion af menneskelige bevægelsesmønstre på baggrund af spatiotemporal data

Author(s):
Christian Raaby

Supervisor(s):
Michael Kai Petersen
Sune Lehman Jørgensen

Technical University of Denmark
Informatics and Mathematic Modelling (IMM)
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone. +45 4525 3351, Fax +45 4588 2673
E-mail: reception@imm.dtu.dk
www.imm.dtu.dk

Release date:	30-09-2011
Class:	1. (Public)
Edition:	1. edition
Comments:	This report is a part of the requirements to achieve Masters degree in Digital Media Engineering at Technical University of Denmark. The report represents 30 ECTS points.

Indholdsfortegnelse

1. Abstract

In this report modern methods for analysis and modeling of human mobility is studied. Further a model is developed, which based on spatiotemporal data can predict human mobility. The model is fitted and evaluated on a spatiotemporal dataset consisting of observations of 4367 DTU students and employees, collected in the spring semester of 2011.

2. Resumé

I denne rapport studeres moderne metoder til analyse og modellering af menneskelige bevægelsesmønstre. Desuden udvikles en model der ud fra spatiotemporal data kan modellere og prædiktere disse bevægelsesmønstre. Modellen tilpasses og evalueres på et spatiotemporalt datasæt bestående af observationer af 4367 DTU ansatte og studerende, indsamlet i forårssemestret 2011.

3. Indledning

Hver dag bevæger hver eneste af de mere end 6 milliarder mennesker på jorden sig. Disse bevægelser kan foregå over afstande fra ganske få meter til mange tusinde kilometer. Ved første øjekast kan bevægelserne virke kaotiske og tilfældige, men virkeligheden er at langt de fleste mennesker har et mere eller mindre periodisk bevægelsesmønster. Dette er ligeså sandt for forretningsmanden der pendler hundredvis af kilometer hver dag, som for pensionisten der sjældent bevæger sig mere end et par kilometer fra sit hjem. At menneskelige bevægelsesmønstre ikke er tilfældige er et vigtigt observation der betyder at det enkelte individs bevægelsesmønster har personlige karakteristika der gør det interessant at analysere og modellere på individuelt niveau. Studiet af individuelle menneskelige bevægelsesmønstre er ikke bare interessant fra et akademisk synspunkt, men har vidtrækkende anvendelses muligheder indenfor en lang række kommercielle felter.

Ved bedre at forstå individuelle bevægelsesmønstre og samtidig være i stand til at modellere disse vil det med stor sandsynlighed være muligt at forbedre nuværende modeller for massebevægelse. Modeller som bl.a. anvendes i forbindelse trafikplanlægning, køsimulering, information- og sygdomsspreding, ad-hoc netværk (29. André Panisson. et al.) m.fl. Men hvad der er mindst ligeså relevant i disse personaliserings tider er mulighederne for at målrette information og reklamer til den individuelle bruger, på baggrund af karakteristika i bevægelsesmønster og prædiktion af fremtidig position. Udover de indlysende kommercielle muligheder for målrettet reklame kunne en interessant anvendelse være en service til forudsigelse af hvor og hvornår du kan forvente at møde dine sociale relationer eller en mobil applikation der automatisk kan give dig information om det sted du er på vej hen.

Et konkret kommercielt eksempel på analyse og modellering af menneskelige bevægelsesmønstre findes i Kastrup Lufthavn, hvor der i skrivende stund arbejdes udviklingen af et system til registrering og analyse af bevægelsesmønstre for lufthavnens passagerer. Systemets primære formål er at overvåge og analysere passagerflow gennem lufthavnens forskellige områder, ventetid ved check-in skranker og lignende, samt identificere og analysere typiske 'paths' eller bevægelsesmønstre gennem lufthavnen. Alt dette sker med henblik på kommerciel og logistisk optimering af lufthavnens opbygning og dimensionering (22. Negroni). Læs mere om passager tracking i Kastrup Lufthavn i appendix A.

Der er således klart at det ikke kun er af akademisk, men at der er store kommercielle perspektiver i analyse og modellering af individuelle menneskelige bevægelsesmønstre. Formålet med dette projekt er derfor at udvikle en metode til modellering af individuelle menneskelige bevægelsesmønstre. Projektet tager udgangspunkt i den subpopulation der

udgøres af DTUs ansatte og studerende og deres bevægelsesmønstre indenfor DTUs campus i Lyngby. Datagrundlaget er observationer gjort af 4367 brugere på DTUs campus i perioden fra Mandag d. 24/01-2011 til mandag 9/5-2011. Datasættets factsheet følger herunder og yderligere information kan findes i appendix B.

Factsheet for DTU-datasæt

- Sampling frekvens: ca. 1 sampling pr. 10 min
- Antal accesspoints: 430
- Antal brugere: 4367
- Observations periode: Mandag d. 24/01-2011 til mandag 9/5-2011
- Trådløsteknologi: WiFi
- Observations format: observations tid i sekunder siden 1. Januar 1970 og position i form af accesspoint SSID
- Data format: Comma Seperated Values (CSV)
- Gennemsnitligt antal check-ins pr. bruger i observationsperioden: 599
- Gennemsnitligt antal besøgte accesspoints pr. bruger i observationsperioden: 28

Nærværende rapport er struktureret på følgende måde. Først følger afsnitter Teoretisk grundlag i hvilket metoder til analyse og modellering af menneskelige bevægelsesmønstre beskrives og efterprøves på DTU-datasættet og det teoretiske overblik emnet menneskelige bevægelsesmønstre dannes. Herefter følger afsnittet Prædiktions modeller PMM og PSMM tilpasset DTU der er en beskrivelse af et konkret eksempel på modellering af DTU-populationen med udgangspunkt i modellerne PMM og PSMM. Der følges op på resultaterne af dette modellerings forsøg i afsnittet Evaluering af resultater og endelig afrundes opgaven med en samlet konklusion i afsnittet Konklusion. Ud over rapportens ordinære afsnit følger appendix A, B, C og D, der indeholder beskrivelse af henholdsvis studierne af menneskelige bevægelsesmønstre i Kastrup Lufthavn, DTU-datasættet og indsamling af spatiotemporal data på DTU via mobilapplikation, samt kildekoden til det primære PSMM-modellerings script (Python).

4. Teoretisk grundlag

Alle mennesker bevæger sig som tidligere nævnt rundt i større eller mindre omfang hvad enten det foregår ved egen kraft i gang, løb eller på cykel eller ved brug af transporthjælpemidler som biler, tog eller fly. Dette afsnit er dedikeret til studiet af spatiotemporale menneskelige bevægelsesmønstre, altså individets position i rummet som funktion af tid. Det endelige mål i nærværende projekt er som allerede nævnt at blive i stand til at modellere menneskelige bevægelsesmønstre og herfra prædiktere menneskelig bevægelse, hvorfor alle studier vil blive betragtet i dette perspektiv.

For at kunne skabe en model til forudsigelse af menneskelig bevægelse er det nødvendigt at undersøge hvilke faktorer der har indflydelse på disse bevægelser og hvordan. Lad os starte med at inddele menneskelige bevægelser i to kategorier mikro- og makro-bevægelser. Mikro-bevægelser omfatter kort sagt korte bevægelser i tid og rum, men kan karakteriseres ved ofte at være direkte reaktioner på det øjeblikkelige nærmiljø. Eksempelvis betragtes et kort stop for at vente på at en bil passerer før en vej krydses eller en meter afvigelse fra den direkte vej mellem to punkter for at undvige en regnpyt, som mikro-bevægelser. Forudsigelse af denne type bevægelser er udenfor målet med denne opgave og vil blive derfor ikke blive studeret yderligere i denne opgave. Makro-bevægelser er modsat defineret som bevægelser af længere udtrækning i tid og rum og kan desuden karakteriseres ved have et specifikt formål ud over optimere den øjeblikkelige interaktion med nærmiljøet. Et eksempel kunne være at gå til et super marked nede af vejen for at købe dagligvare mandag eftermiddag eller at cykle til en restaurant fredag aften, for spise med en ven. Bevægelser af denne type af interessante af to grunde for det for de første fordi det er forudsigelse af bevægelser i dette størrelsesforhold der har en værdi i forbindelse med by- og trafikplanlægning, mund til mund informationsspredning, sygdomsspredning og meget mere. For det andet gør den formålsdrevne karakter af disse bevægelser mulighederne for forudsigelse langt større.

Med indsnævringen til makro-bevægelser kan vi nu begynde at overveje hvilke faktorer der påvirker disse formålsdrevne bevægelser og hvordan. Der er almindelig enighed om at langt de fleste mennesker har stærke spatiotemporale regelmæssigheder (11. Marta C. Gonza´lez. et al.),(10. Juyong Park. et al.), (7. Eunjoon Cho. et al.). De fleste kan eksempelvis nikke genkendende til befinde sig samme sted de fleste mandag morgener kl. 9 (på arbejdet) eller tirsdag nat kl. 3 (hjemme), hvilket tyder på at tidspunktet på dagen og eller ugen har afgørende indflydelse på et individs position og at der derudover findes et begrænset antal lokationer der frekventeres meget ofte. Ud over de spatiotemporale faktorer et de fleste enige om at et menneskes bevægelser påvirkes af andre mennesker og i særlig grad af et begrænset antal personer der på den ene eller anden måde har en social eller professionel relation til individet. Der findes således allerede en del studier af sammenhængen mellem spatiotemporale sammenfald og venskab,

samt undersøgelser sociale relationers indflydelse på et individs bevægelses mønster (9. Justin Cranshaw. et al.),(7. Eunjoon Cho. et al.),(5. Dashun Wang. et al.)(13. Nathan Eagle. et al.)(6. David J. Crandalla. et al.)(4. Chiara Boldrini. et al.)

Jeg vil i det følgende gå i dybten med et udvalg af disse studier, herunder spatiotemporale rutiner, sammenhængen mellem spatiotemporal data og sociale relationer og sociale relationers indflydelse på menneskelige bevægelsesmønstre.
Derudover vil jeg se nærmere på hvilke metoder der på nuværende tidspunkt anvendes til analyse og modellering af menneskelig bevægelse. Endelig studeres en metode til bestemmelse af den teoretisk maksimale forudsigelsespræcision, af en brugers bevægelser, der er mulig med et givent datagrundlag.

4.1. Bevægelsesmønstre og sociale relationer

Hvis man spørger sig selv – Har mine sociale relationer indflydelse på hvordan jeg bevæger mig rundt? Er det indlysende svar – Ja! Overvej blot sandsynligheden for at du skulle befinde dig på en persons privat adresse hvis denne ikke er din ven, et familiemedlem, kæreste eller på anden måde en social relation – dette vil i de fleste tilfælde være usandsynligt. Tænk yderligere over hvor ofte du egentlig befinder dig på en social relations adresse uden beboeren er til stede – Igen vil de fleste nå frem til at dette ikke sker ret ofte. Intuitivt virker det således indlysende at der kan være en sammenhæng mellem ikke bare hvor, men også hvornår et individ befinder sig på en given lokation og individets sociale relationer. Spørgsmålet er så blot om dette kan vises kvantitativt og hvordan det kan udnyttes i forudsigelse af en persons geografiske placering. Derudover foreligger endnu en udfordring i og med DTU-datasættet ikke indeholder nogen eksplicit information om det underliggende sociale netværk. Det er således nødvendigt at konstruere det ønskede sociale netværk fra den tilgængelige spatiotemporale data, for at kunne gennemføre der ønskede studier. Herunder vil jeg præsentere en række studier af sammenhængen mellem spatiotemporal data, sociale relationer og bevægelsesmønstre.

4.1.1. Sociale relationer og Co-occurences

Hver dag, udledes antagelser om den sociale verden fra ufuldstændige observationer af begivenheder omkring os. En særlig kategori af disse antagelser er baseret co-occurences I tid og rum (også kaldet spatiotemporale co-occurences) – hvor sociale relationer mellem to individer baseres på at disse har befundet sig på samme location på ca. samme tidspunkt. Dette er udover at være intuitivt fornuftigt, en almindeligt anvendt antagelse I psykologiske studier af byliv (25. Milgram) og retslig analyse af farerne ved 'guilt by association' (26)(27). På trods af den hyppige anvendelse af sammenhæng mellem co-occurrence og sociale relationer er det ikke før (6. David J. Crandalla. et al.) (2010) at denne sammenhæng er blevet undersøgt kvantitativt baseret på empirisk data. Jeg vil her gennemgå de store linjer undersøgelsesn fremgangsmåde og de vigtigste resultater. Den udførte kvantitative undersøgelse

tager således udgangspunkt i data indsamlet fra det sociale medie, Flickr (28), en online service til deling af privat billedmateriale. Det spatiotemporale element i Flickr introduceres i form at geotagging, en funktion der giver brugerne mulighed for at angive hvor og hvornår et billede er blevet taget. Flikcr har desuden en social komponent der muliggør eksplicit registrering af venskaber mellem brugere.

En co-occurrence defineres således som den situation hvor to brugere tager et billede ca. same sted på ca. samme tidspunkt. I praksis inddeles jorden i et gitter af celler hver med en side længde på *s* længde- og *s* breddegrader. To individer A og B har således en co-occurrence i celle C hvis de begge har taget et billede her inden for den temporale distance af *t* dage. (6. David J. Crandalla. et al.) søger nu at belyse spørgsmålet: Hvad er sandsynligheden for at der er en social relation mellem to individer, givet at der observeres co-occurences mellem dem i *k* unikke celler inden for den temporale periode *t*. Svaret herpå afhænger at tre parametre antallet af unikke co-occurrence-lokationer *k*, der giver mængden af bevis materiale for en social relation, sammen med størrelsen af *s* og *t*, der bestemmer præcisionen af dette bevismateriale.

Undersøgelserne påviser den forventede sammenhæng mellem co-occurrences og sociale relationer nemlig at sandsynligheden for en social relation vokser som funktion af antallet af co-occurences. Det overraskende element her, er hvor hurtigt denne sandsynlighed vokser. To tilfældigt udvalgte Flickr-brugere har 0.0134% sandsynlighed for at have en social relation, denne sandsynlighed vokser signifikant når to brugere har et antal spatiotemporale co-occurences. Eksempelvis har to brugere næsten 60% sandsynlighed (over 5000 gange baseline sandsynligheden) for at have en social relation på Flickr når de har 5 co-occurences indenfor en temporal periode på en dag, I 5 unikke celler med side længder på henholdsvis 1 bredde- og 1 længdegrad (svarende til ca. 80 km, i nærheden af ækvator). Og selv med bare 3 co-occurrences med samme værdier af s og t, er sandsynligheden for en social relation ca. 5% hvilket er mere en 300 gange højere end datasættets baseline.

(6. David J. Crandalla. et al.) fortsætter med at forfine metoden ved at justere for cellernes varierende størrelse over jordens overflade og ved at introducere forskellige metoder til optælling af unikke co-occurences. De finder at der med passende justeringer kan fastslås en proportionalitet mellem størrelsen af *s* og *t* og signifikansen af antallet af co-occurrences. Konklusionen forbliver imidlertid den samme nemlig at der er en stærk proportionalitet mellem co-occurrences og sociale relationer.

4.1.2. Signifikans af co-occurences

(9. Justin Cranshaw. et al.) beskriver i deres paper *Bridging the gap between physical location and online social networks* en metode hvormed det er muligt at tage højde for at ikke alle co-occurrences har samme signifikans, i spatiotemporale datasæt med højere observations opløsning end det var tilfældet for Flickr, i forhold til at antyde et venskab mellem to individer.

Eksempelvis er en observation af to individer på en privat adresse uden andre personers tilstedeværelse en langt mere betydningsfuld indikator af et venskab end en observation af de samme to personer i kantinen på universitetet. De anvender entropien af den lokation hvor en co-occurence finder sted som et mål for signifikansen af denne co-occurrence. Hvor en lav entropi af lokationen hvor en co-occurence finder sted angiver en signifikant co-occurrence, mens en høj entropi angiver en usignifikant co-occurrence. En lokations entropi er givet ved.

Equation 1

$$-\sum_{u \in U_l} P_l(u) \log_2\left(P_l(u)\right)$$

hvor $P_l(u)$ er den totale andel af alle check-ins på lokation l, foretaget af bruger u, og U_l sættet af samtlige brugeres check-ins på lokation l. På Figure 1 ses det hvordan lokations entropien for et geografisk område omkring et universitet fordeler sig. Figuren er lavet på baggrund af deres observationer af 489 brugere observeret i en gennemsnitlig periode på 74 dage.

(9. Justin Cranshaw. et al.) observerer en signifikant forbedring I deres models evne til at forudsige venskab mellem de observerede personer når denne trænes på et datasæt af co-occurrence data vægtet i forhold til locations entropi, frem for træning baseret på de rå co-occurrence data.

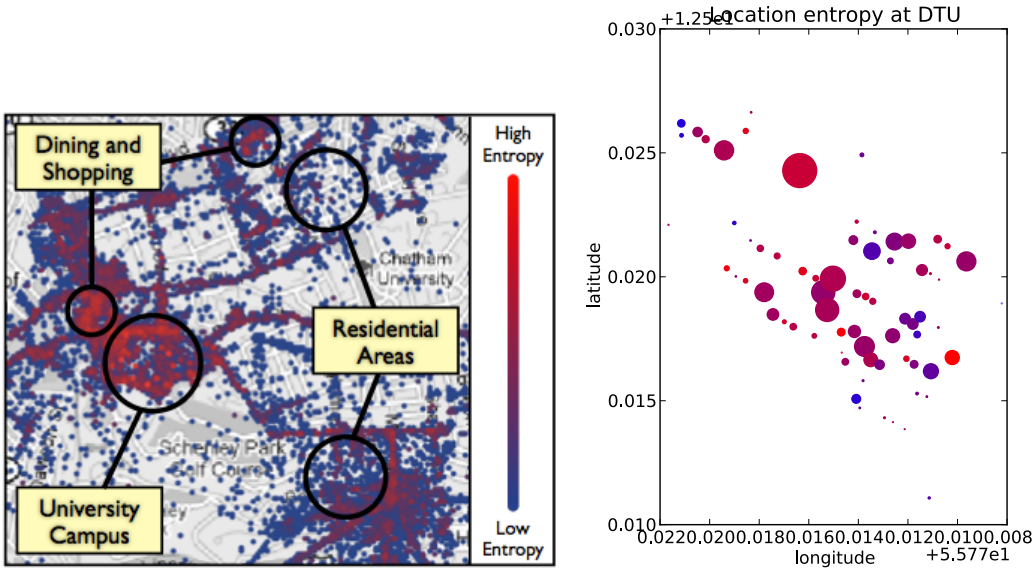


Figure 1: (figuren til venstre er lånt fra (9. Justin Cranshaw. et al.)) Til venstre ses et kort over et amerikansk universitetscampus og de omkringliggende indkøbs- og boligområder. Et områdes farve angiver dets entropi. Af figuren fremgår det at campus og indkøbsområdet har høj entropi (rød), mens bl.a. boligområdet har lav entropi (blå). Til højre ses et tilsvarende plot for DTU her er indikere en cirkel samlingen af accesspoints på en given geografisk lokation. Cirklens størrelse angiver check-in frekvensen på lokationen (store cirkler betyder mange check-ins), mens farven ligesom på plottet til venstre angiver lokationens entropi. Da den data figuren til venstre er baseret er indsamlet fra brugernes synspunkt, mens DTU-datasættet er indsamlet fra netværkets synspunkt er der nogle afgørende forskelle. For det første er antallet af mulige positioner lang mere begrænset i DTU-sættet derudover indeholder DTU-sættet kun data fra campus. Ikke desto mindre er de

observerede entropi forskelle indenfor DTU's campus stadig interessante og det er tydeligt at forskellige lokationer har forskellig entropi og dermed at der er grund til at inkludere dette mål når et social netværk skal udledes fra de spatiotemporale data.Lav evt.

4.1.3. Sociale relationer og mobiltelefoner

The Reality Mining Study (13. Nathan Eagle. et al.) er en undersøgelse udført på MIT, hvor 94 personer blev udstyret med mobiltelefoner med en pre-installeret applikation til indsamling af data. Applikationen indsamlede call logs, synlige Bluetooth devices indenfor ca. 5 meters afstand, cell tower ID, applikations brug, og telefon status, over en periode på 9 måneder. De undersøgte individer var alle fra samme universitet men fulgte 2 forskellige studie retninger og inkluderede både studerende og ansatte. De deltagende personer blev desuden bedt om at udfylde skemaer med information om hvem iblandt de andre deltagere de opfattede som venner, hvor og hvornår de befandt sig i nærheden af disse, samt deres tilfredshed med deres arbejdsgrupper.

Der blev således foretaget tre analyser af disse data. For det første blev forholdet mellem deltagernes selv-rapporterede data og den kvalitative data indsamlet ved hjælp af mobiltelefonerne. For det andet blev det undersøgt om venskab kan udledes af de kvalitative data og endelig sammenhængen mellem observationerne I den kvalitative data og den individuelle deltagers tilfredshed.

Jeg vil her kun se nærmere på resultaterne af analyse nummer to, nemlig hvorvidt det er muligt at udlede venskab mellem personer på baggrund af den indsamlede data. (13. Nathan Eagle. et al.) indleder deres undersøgelse med at slå fast at observation af kognitive forhold, som venskab og kærlighed er en fundamentalt anderledes udfordring end at observere om to personer befinder sig I nærheden af hinanden. Eksempelvis kan to personer sagtens være venner uden at der er nogen (med den anvendte applikation) observerbare interaktioner mellem dem I en længerevarende periode. Ikke desto mindre er kontekst , og især den spatiotemporale kontekst sandsynligvis en god indikator for forskellige typer af sociale relationer. Hvis observationer af spatiotemporale sammenfald mellem forsøgsdeltager deles op i *on campus* (observationer gjort mellem 8.00 og 20.00) og *off campus* (observationer gjort mellem 20.00 og 8.00) viser det sig at det er muligt at udlede 95% af alle venskaber og 96% af alle ikke-venskaber på baggrund af blot *off campus* sammenfald. Det observeres desuden at datasættet indeholder store mængder redundant data og således at det er muligt med stor præcision at reproducere resultaterne baseret på 9 måneder data ved hjælp af blot to ugers data.

En anden undersøgelse foretaget af Dashun Wang. et al. (5. Dashun Wang. et al.) viser ligeledes at sammenfald i to individers bevægelsesmønstre giver en stærkt forøget sandsynlighed for en social relation mellem de to. Her er undersøgelserne baseret på spatiotemporal data fra 6 milliioner mobiltelefon-brugere

4.1.4. Bevægelsesmønstre og sociale relationer

Det ligger nu fast at vi med et solidt videnskabeligt grundlag kan forvente at kunne tilnærme det underliggende sociale netværk på DTU med den rette udnyttelse af informationen i den tilgængelige spatiotemporale data, men hvordan kan dette udnyttes i prædiktion af en brugers fremtidige bevægelser. Eunjoon Cho. et al. (7. Eunjoon Cho. et al.) undersøger 2 forskellige måder hvorpå sociale relationer påvirker et individs mobilitet. For det første ser de på sandsynligheden for at besøge en social relation's hjemme-lokation som funktion af rejseafstanden (den afstand en person bevæger sig mellem to check-ins). Dernæst undersøger de hvordan det påvirker sandsynligheden for at foretage et check-in på en given lokation at en social relation har checket ind på lokation først. Den empiriske data der ligger til grund for disse undersøgelser er spatiotemporal-, samt eksplicit social netværksdata fra henholdsvis 196,591 og 58,228 brugere af de to lokations baserede netværk Gowalla (23.) og BrightKite (24.)

Det viser sig mod forventning at den relative sandsynlighed for at besøge en social relations hjemme-lokation vokser i takt med den rejste afstand, når man tager i betragtning det voksende antal mulige besøgslokationer. Det viser sig at der i det benyttede datasæt er 10 gange større relativ sandsynlighed for at en bruger besøger en social relation ved rejseafstand på 1000 km i forhold til en rejseafstand på 40 km.

Når det kommer til undersøgelsen af hvordan en social relations besøg til en lokation påvirker brugerens sandsynlighed for at rejse til denne, så observere (7. Eunjoon Cho. et al.), i deres to datasæt, kun hendholdsvis 4,1% og 9,6% tilfælde hvor en social relation havde besøgt en lokation der umiddelbart herefter blev besøgt af brugeren. De fortsætter med at undersøge sammenhængen mellem en høj grad af sammenfald i besøgte lokationer og vensskab og finder at et 40% rummeligt sammenfald i check-ins (altså uafhængigt af hvornår disse check-ins har fundet sted), resultere i en en 30% sandsynlighed for et venskab. På trods af den påviste stærke rummelige sammenhæng mellem besøgte lokationer og venskab er det også nødvendigt at der er en hvis temporal overenstemmelse for sociale relationer kan bruges til at give meningsfuld information om en bruger's øjblikkelige lokation. De undersøger derfor for hver bruger hvordan antallet af tilfælde hvor en ven besøger en lokation umiddelbart før brugeren fordeler sig. Det viser sig at 84% af brugerne, i 20% af deres besøg til en lokation, følger umiddelbart efter et besøg fra en ven. Derudover blev det observeret at 52% slet ikke havde nogle registrerede tilfælde hvor de besøgte en lokation umiddelbart efter en ven.

På baggrund af de præsenterede data er det svært at drage andre universelt gældende konklusioner end den at sociale relationer til en person der befinder sig tæt på denne i tid og rum lader til at have en positiv tiltrækningskraft på personen. Den eksakte beskrivelse af denne tiltrækningskraft for sociale relationer må i stedet udledes for den konkrete undersøgte population.

4.1.5. Konklusion på bevægelsesmønstre og sociale relationer

Det kan konkluderes at der findes et stærkt videnskabeligt grundlag for at udlede sociale relationer fra spatiotemporal data. Det kan samtidig konstateres at viden om sociale relationer i nogen grad kan bruges til at bestemme en brugers position, om end de eksakte retningslinjer for hvordan dette gøres er en smule usikre og må forventes at variere fra datasæt til datasæt. Yderligere undersøgelser at dette emne følger derfor i beskrivelsen af den konkrete implementerede modellerings og prædiktionsmodel.

4.2. Individuelle menneskelige bevægelsesmønstre

På trods af at viden om menneskelige bevægelsesmønstre som tidligere nævnt er essentiel indenfor en lang række videnskabelige felter er det først inden for de sidste par år der er blevet gjort signifikante opdagelser indenfor feltet. Dette skyldes ikke manglende interesse indenfor feltet men snarere at det indtil nu ikke har været muligt at studere et stort antal individers bevægelse kontinuerligt over et større område. Med den stigende popularitet og brug af personlige elektroniske devices med trådløs netværksforbindelse og/eller GPS, som mobiltelefoner, smartphones, PDA'er og laptops er det gennem de sidste par år blevet muligt at generere store datasæt med den fornødne spatiotemporale data.

Det er således nu blevet muligt at teste validiteten af en række af de modeller som klassisk er blevet anvendt i beskrivelsen af menneskelige bevægelsesmønstre så som Lèvy flight og random walk (15. H. Eugene Stanley. et al.). Mens disse modeller der sædvanligvis bruges til modellering af partikelbevægelser og succesfuldt kan bruges til at beskrive bl.a. gas diffusion i rummet og mange andre fysiske systemer, så viser (11. Marta C. Gonza´lez. et al.) at disse modeller ikke har samme success når det kommer til beskrivelsen af menneskelig bevægelse. Undersøgelserne er baseret på to datasæt et bestående af positionsdata fra 100.000 mobiltelefon brugere i en periode på 6 måneder det andet af data fra 206 brugere over en periode på en uge. For brugerne i det første datasæt blev deres position hver gang der blev foretaget eller modtaget et opkald eller blev sendt eller modtaget en sms. I det andet datasæt blev brugernes position registreret regelmæssigt med 2 timers mellemrum.

Jeg vil her gennemgå de essentielle undersøgelser og beregninger der leder (11. Marta C. Gonza´lez. et al.) til at afvise Lèvy flight og random walk som passende modeller til beskrivelse af individuel menneskelig bevægelse. En interessant og essentiel opdagelse gjort af (11. Marta C. Gonza´lez. et al.) er at størrelsen på det område et individ bevæger sig indenfor lader til af være begrænset eller i hvert fald kun vokser ekstremt langsomt som funktion af tiden *t*. Den opdagelse er vigtig i det den står i skarp kontrast til Lèvy flight og random walk modellerne, hvor den forventede forskydning eller rejseafstand af en partikel vokser ubegrænset som polynomier af tiden *t* (15. H. Eugene Stanley. et al.). For at efterprøve om de nærværende DTU trajectories kan forventes at være fuldstændig tilfældige og derfor bedst beskrives ved en af ovenstående modeller eller om de indeholder en spatiotemporal regelmæssighed der kan udnyttes i en

mere avanceret prædiktionsmodel, udregnes radius of gyration og rejseafstandsfordelingen herunder.

Radius of gyration er defineret som den gennemsnitlige lineære afstand r_g et device bevæger sig fra observationernes tyngdepunkt r_{cm}^a til observations step t

Equation 2

$$r_g^a(t) = \sqrt{\frac{1}{n_c^a(t)} \sum_{i=1}^{n_c^a} (r_i^a - r_{cm}^a)^2}$$

hvor r_i^a repræsentere den $i = \{1, \cdots, n_c^a(t)\}$ position registreret for device a og observationernes tygndepunkt r_{cm}^a er givet ved

Equation 3

$$r_{cm}^a(t) = \frac{1}{n_c^a(t)} \sum_{i=1}^{n_c^a} r_i^a$$

På de to plots herunder ses hvordan radius of gyration udvikler sig som funktion af tiden for hendholdsvis 10 og 1000 DTU devices.

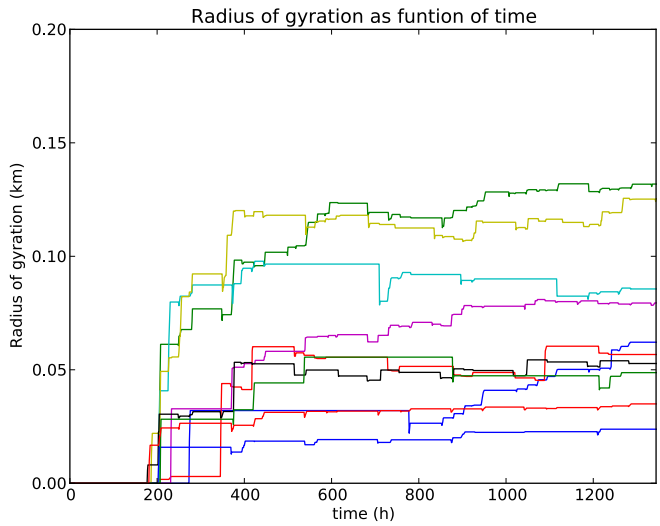


Figure 2: Plottet viser 10 DTU brugeres radius of gyration som funktion af tiden i timer. Plottet viser at radius of gyration for de fleste brugere er stabiliseret allerede efter ca. 200 timer eller lidt over en uge.

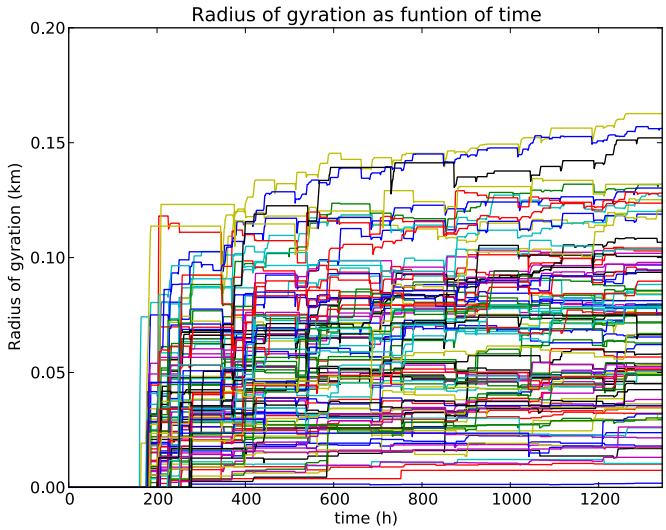


Figure 3: Med 1000 brugere ser vi samme tendens til hurtigt stabilisering om end det svært at skelne de enkelte brugere fra hinanden i plottet.

På baggrund af den hurtige stigning efterfulgt af næsten fuldstændig udfladning observeret i udviklingen af radius of gyration som funktion af tiden t (samme tendens kan ses for DTU individer på Figure 2 og Figure 3), samt konstateringen meget høje besøgsfrekvenser af lille antal lokationer, så som hjem og arbejder konkluderer (11. Marta C. Gonza´lez. et al.) at menneskelige bevægelsesmønstre indeholder en høj grad af regelmæssighed.

(11. Marta C. Gonza´lez. et al.) undersøger herefter hvordan rejseafstanden Δr givet ved afstanden mellem to på hinanden følgende bruger registreringer fordeler sig over hele populationen. Det viser sig at populationens rejseafstande tilnærmelsesvis kan beskrives ved en trunkeret potens-lov (se Figure 4 for et histogram over DTU populationens rejseafstande, samt en approksimation til denne fordeling ved en trunkeret potens-lov)

Equation 4

$$P(\Delta r) = (\Delta r + \Delta r_0)^{-\beta} \exp(-\Delta r / k)$$

hvilket peger i retning af at individerne heri bevæger sig efter en truncated Lèvy flight model på trods af den netop anviste regelmæssighed hos de enkelte individer. Denne opførsel kan dog forklares ved at forskydningsdistributionen for den samlede population i virkeligheden er resultatet af en foldning af de individuelle trajectories karakteristika og den samlede populations heterogenitet.

Endelig undersøges ensartetheden af menneskelige bevægelsesmønstre for at belyse mulighederne lave en generel model for prædiktion af et individs position på baggrund af historisk spatiotemporalt bevægelsesdata. De enkelt brugere bevæger sig selvsagt i forskellige områder og disse områder størrelse, givet ved radius og gyration, variere ligeledes. Den interessante observation de gør sig her at anisotropien, altså retningsbestemtheden af en bruger bevægelser er proportional med dennes radius of gyration. Det vil sige at individer der rejser

lange afstande almindeligvis gør det i en bestemt retning mens individer der bevæger sig inden for et mindre område bevæger sig mere omnidirektionelt. De viser sig at en reskalering i forhold anisotropi, og radius og gyration gør sandsynlighedsfordelingerne stort set ens på tværs af det meget omfattende datasæt, hvilket yderligere forbedrer mulighederne i en generel positions prædiktionsmodel. DTU's relativt begrænsede geografiske udstrækning betyder at vi kan se bort fra forskelle i anisotropi i den implementerede prædiktionsmodel.

Konklusionen på (11. Marta C. Gonza´lez. et al.) undersøgelse af individuel menneskelig bevægelse bliver således at menneskelige bevægelsesmønstre indeholder en høj grad af spatial og temporal regelmæssighed. Som et resultat heraf er klassiske partikel forskydningsmodeller som Lèvy flight og random walk der antager at det modellerede objekt bevæger sig tilfældigt, ikke egnede til at beskrive individuel menneskelig bevægelse.

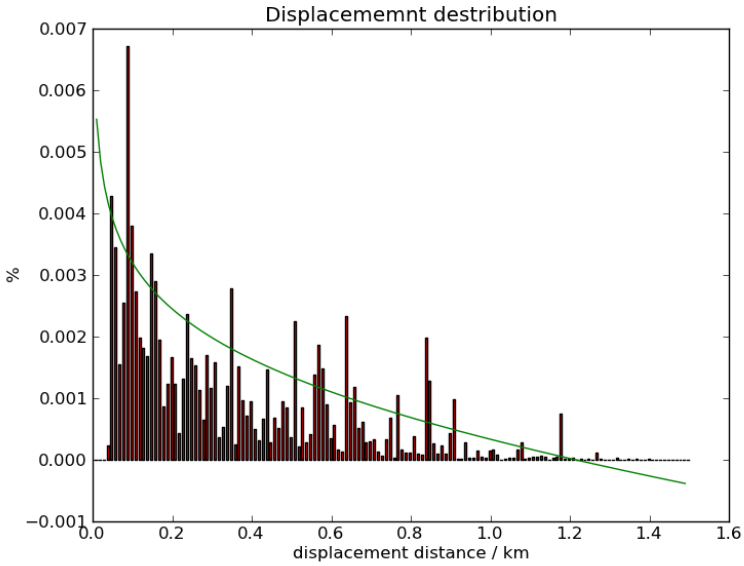


Figure 4: Histogram over rejseafstande Δr, givet ved afstanden mellem to på hinanden følgende registreringer, for alle devices i DTU populationen, samt en approksimation til fordelingen af formen givet i Equation 4.

4.2.1. Eigenmode analyse af menneskelig bevægelse

Vi nu ved at individuel menneskelig bevægelse indeholder en høj grad af spatiotemporal regelmæssighed og at klassiske metoder til beskrivelse af partikel bevægelse netop af denne grund ikke er egnede til at beskrive menneskelig bevægelse. Det er således nødvendigt at finde mere egnede metoder til analyse og og beskrivelse af menneskelig bevægelse. En sådan metode er eigenmode analyse der med udgangspunkt i Markov processer der for det første forsøger at bestemme clusters i spatiotemporal data og dernæst analysere transitions tendenser mellem disse clusters.

I denne tilgang til analyse af menneskelige bevægelsesmønstre først præsenteret af (10. Juyong Park. et al.) tages som sagt udgangspunkt i Markov Processen (18.

Weisstein), en fundamental og uundværlig metode anvendt i studier af stokastiske processer indenfor en lang række videnskabelige felter så som lingvistik, bioinformatik, grafteori, og informationsteori m.fl. En Markov-chain er en tidsdiskret stokastisk proces med Markovegenskaben. Det vil sige, at processens forløb kan bestemmes ud fra dens nuværende tilstand uden kendskab til tidligere tilstande. En Markov process er en tidskontinuerlig Markov-chain. Jeg vil i det følgende gennemgå og efterprøve de centrale dele af de anvendte analyse metoder på DTU datasættet og sammenligne og evaluere i forhold til resultaterne præsenteret af (10. Juyong Park. et al.). Først konstrueres den Markov transitions matrix fra datasættet, hvis eigenmodes danner grundlag for de efterfølgende analyser.

Markov eller transitions matricen M, er givet ved $M = \{m_{ij} \equiv m_{i \leftarrow j} | 1 \leq i, j \leq K\}$, hvor $K \leq N$, antallet af check-ins for den givne bruger og m_{ij} er sandsynligheden for at brugeren checker ind ved accesspoint *i* umiddelbart efter at have checket ind ved accesspoint *j* uden andre check-ins der imellem. m_{ij} udregnes således ved n_{ij} / n_j hvor n_{ij} er antallet af disse *i* på *j* følgende check-ins og n_j det samlede antal check-ins på accesspoint *j* i obervations perioden. Matricen M, konstrueret på denne måde udviser følgende matematiske egenskaber. For det første summere hver af matricens søjler til 1 idet $\sum_i m_{ij} = \sum_i n_{ij} / n_j = 1$. Det samme gør sig ikke gældende for matricens rækker da M er asymmetrisk. For det andet gælder det ifølge Perron-Frobenius teoremet (19. Weisstein), at M altid har en førende eigenværdi $\lambda_0=1$, hvis tilhørende eigenvektor er $e_0=p$, hvor $p=\{p_1,p_2,...,p_k,\}$ repræsenterer den stationære sandsynlighed p_i, for at finde brugeren på et givent accesspoint *i*, $p_i=n_i/N$. Dette kan forklares ved definitionen af den stationære sandsynlighed der er givet ved $p_i = \sum_j m_{ij} p_j$, hvilket også kan skrives som $p=\lambda Mp$ med $\lambda=1$. Endelig opfylder alle eigenværdier $|\lambda_i| \leq 1$, eller med andre ord alle eigenværdier befinder sig inden for enhedscirklen i det komplekse plan.

Ved at opstille et udtryk (Equation 5) for udviklingen af en brugers radius of gyration som funktion af transitionsskridtet *s*, lykkes det (10. Juyong Park. et al.) at påvise at udviklingen af radius of gyration bestemt ved dette udtryk kommer tæt den empirisk fundne radius of gyration og samtidig langt fra en Bernoulli simulation anvendt som sammenligningsgrundlag i undersøgelsen (se Figure 5). Herfra laver (10. Juyong Park. et al.) slutningen af historisk data er vigtig i beskrivelsen af menneskelig bevægelse idet en Bernoulli Matrix er givet ved de stationære sandsynligheder for at befinde sig på en lokation uden hensynstagen til brugerens nuværende position. Den anvendte version af Markov matricen tager imidlertid højde for netop dette, hvilket kan betragtes som at medtage et historisk skridt.

Af Equation 5 fremgår det at den kvadrerede radius of gyration er givet ved en sum af led der hver især kun afhænger af en enkelt eigenmode *k*. Det undersøges nu hvilke eigenværdier der giver den simulering af radius of gyration der kommer tættest på den impiriske data og det viser sig at eigenværdier hvis reelle

værdi er tæt på 1 og komplekse værdi tæt på 0 giver det bedste resultat (se Figure 5).

4.2.2. Radius of gyration og transitions matricen

En brugers radius of gyration kan udregnes som funktion af skridt s, i transitions matrix M ved udtrykket.

Equation 5

$$\overline{r_g^2(s)} = \sum_{k=1}^{K} \rho_k F(\lambda_k, s)$$

hvor den såkaldte mode weight ρ_k er defineret ved $\rho_k \equiv (a_x^k b_x^k + a_y^k b_y^k)$. ρ_k er invariant i s, eller med andre ord uafhængig af hvilken transitions state brugeren befinder sig i. Det gælder desuden at

$$R^{-1}Px = \{a_k^x\} \qquad and \qquad x^T L^{-T} = \{b_k^x\}^T$$

$$R^{-1}Py = \{a_k^y\} \qquad and \qquad y^T L^{-T} = \{b_k^y\}^T$$

her er L og R de venstre og højre eigenvektor-matricer til transitionsmatricen M, normaliseret sådan at $RL^T = I$ og $x = \{x_1, \ldots, x_K\}$ og $y = \{y_1, \ldots, y_K\}$ er koordinaterne på de K unikke accesspoints besøgt af brugeren. $F(\lambda_k, s)$ er givet ved

$$1 - \left(\frac{1+\lambda_k}{1-\lambda_k}\right)\frac{1}{s} + \left(\frac{2\lambda_k(1-\lambda_k^s)}{(1-\lambda_k)^2}\right)\frac{1}{s^2}$$

hvor λ_k er eigenværdi *k* til transistionsmatricen M.

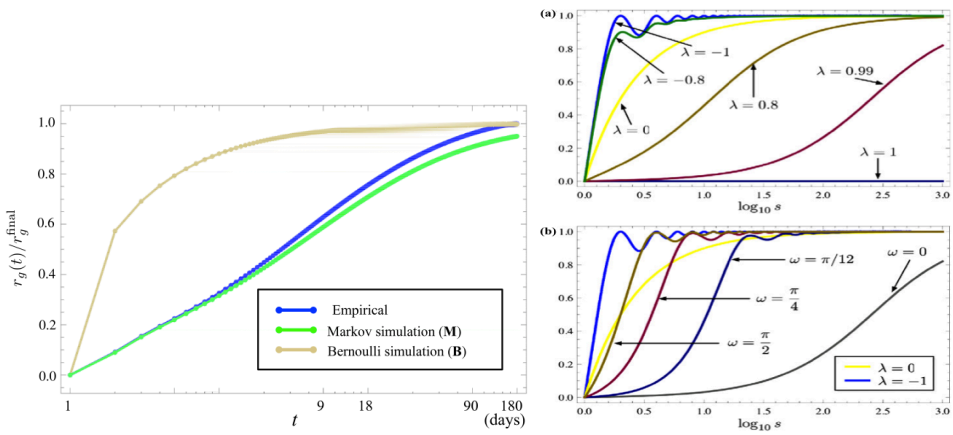


Figure 5: De tre plot herover er lånt fra (10. Juyong Park. et al.). På plottet til venstre ses en sammenligning af udviklingen af radii of gyration for den empiriske data (blå) med Markov simulationen (grøn) og Bernoulli simulationen (beige). I de to plots til højre ser vi at den ønskede

radius of gyration udvikling stammer fra eigenmodes med eigenværdier med reel del tæt på 1 og imaginær del tæt på 0.

Efter at have fastslået at eigenmodes med eigenværdier hvis reelle del er tæt på 1 og imaginær del tæt på 0, undersøger (10. Juyong Park. et al.) hvordan informationen i netop disse eigenmodes kan bruges til at beskrive en brugers bevægelsesmønster. Der ses bort fra den førende eigenværdi λ_0 =1, der ifølge Perron-Frobenius teoremet altid vil være at finde i blandt eigenværdierne for en Markov transitions matrix (da alle elementer er ikke negative) da den tilhørende eigenvektor indeholder de stationære sandsynligheder der i denne sammenhæng er uinteressante. Undersøgelsen af eigenmodes deles op i reelle og komplekse eigenværdier.

Reelle eigenmodes

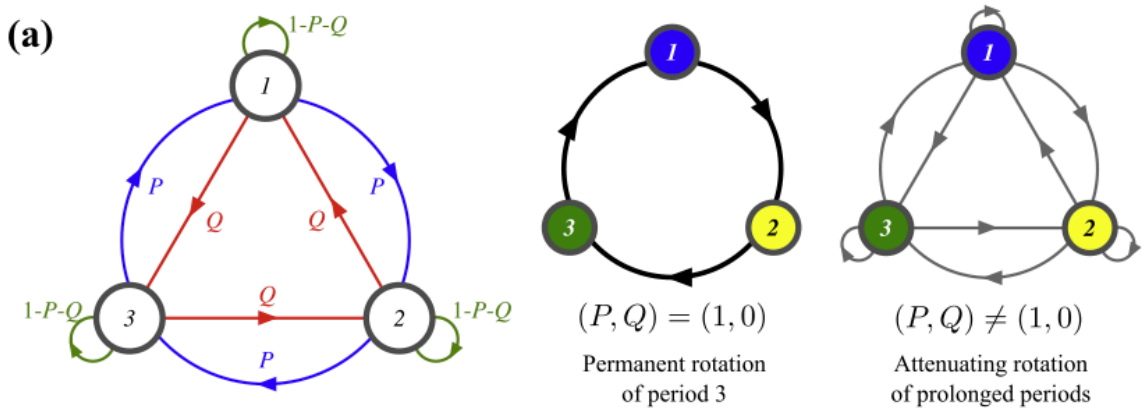
En eigenmode består af et sammenhørende par af eigenværdi og eigenvektor. For reelle eigenmodes kan elementerne I en eigenvektor antage værdier der er enten positive eller negative svarende til at have imaginære dele lig 0 eller π. I en eigenmode analyse betyder det at elementerne og de tilsvarende accesspoints kan indeles I to clusters en for positive og en for negative. Forskellige eigenmodes kan og vil ofte give forskellige clusters hvorfor det kun er eigenmodes med de højeste reelle eigenværdi der tages i betragtning

(10. Juyong Park. et al.) viser at hvis transitioner inddeles I intra- og intercluster transitioner, er det eigenværdien tilhørende den eigenmode der ligger bag cluster-inddelingen der afgør hvilken af disse der er mest sandsynlig. Det gælder således at der for eigenvædier λ_k >0 er størst sandsynlighed for at brugeren vil forblive inden for den nuværende cluster. For λ_k >0 gælder det modsat at brugerne har større sandsynlighed for at skifte cluster i den kommende transition.

Eigenværdiers indflydelse på cluster transition har desuden den side effekt at clusters dannet på baggrund af eigenmodes med store positive eigenværdier har stor sandsynlighed for at indeholde brugeren igennem et større antal på hinanden følgende transitioner. I det store positive eigenværdier jo betyder at skift mellem de definerede clusters er sjældne og dermed at disse clusters er temporalt veldefinerede.

Komplekse eigenmodes

I det komplekse tilfælde gælder overordnet set samme eigenmode tolkning dog kan de imaginære dele nu antage alle værdier I intervallet [0, 2π[. Dette betyder at accesspointene nu kan indeles I mere end 2 clusters. Dette demonstreres på et simpelt eksempel med tre celler se Figure 6. Herefter undersøger (10. Juyong Park. et al.) om denne 3-periode struktur kan genfindes i virkelige data se Figure 7. Dette viser sig at være tilfældet i deres data, men det er ikke muligt påvise samme periodiske opførsel i DTU datasættet.



Eigenmode 0 : $\lambda_0 = 1, \qquad \mathbf{e}_0 = \frac{1}{3}(1,1,1)$

Eigenmode 1 : $\lambda_1 = \frac{1}{2}((2-3(P+Q)) + \mathrm{i}\sqrt{3}(P-Q)),$

$\mathbf{e}_1 = (\mathrm{e}^{-\mathrm{i}2\pi/3}, \mathrm{e}^{\mathrm{i}2\pi/3}, 1)$

Eigenmode 2 : $\lambda_2 = \lambda_1^*, \quad \mathbf{e}_2 = \mathbf{e}_1^*.$

Figure 6: Eksempel lånt fra (10. Juyong Park. et al.). Det antages at P>Q . Vi kan se at e_1 og e_2 inddeler systemets tre celler (1,2 og 3) i tre separate clusters (blå, gul og grøn) uafhængigt af P og Q.

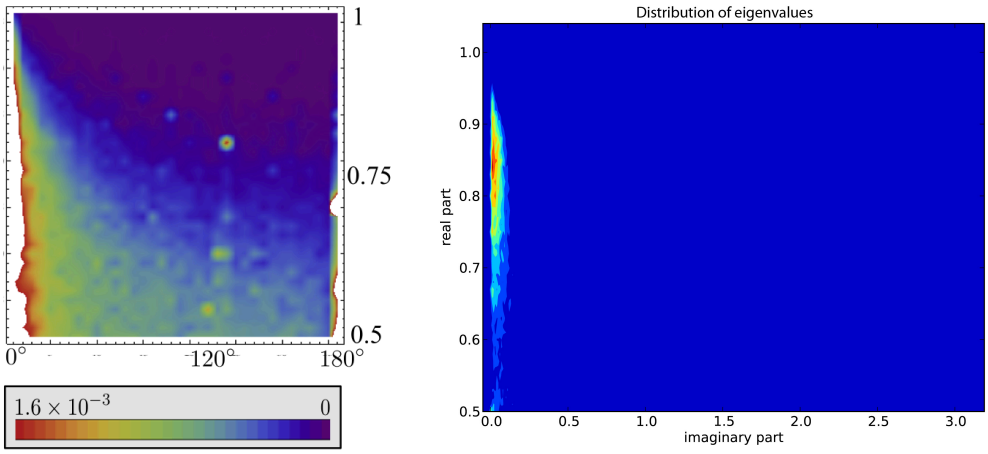


Figure 7: Plottet til venstre er lånt fra (10. Juyong Park. et al.) mens plottet til højre er den tilsvarende fordeling for DTU datasættet. Begge plot indeholder fordelingen over de respektive datasæts eigenværdier i det komplekserum, x-aksen i plottet til venstre er i grader mens x-aksen til højre er i radianer . På plottet til venstre ser vi en høj koncentration af eigenværdier med imaginære værdier tæt på nul det samme gør sig gældende for DTU sættet til højre. Plottet til højre snyder lidt idet at det ikke lader til at indeholde en eneste eigenværdi med imaginære del over 0.2 dette er ikke tilfældet, de er blot så få og så spredte over det komplekse spekrum at de ingen steder opnår en koncetration der er høj nok til at fremgå af konturplottet. Det kan dog med sikkerhed konkluderes at den høje koncentration af eigenværdier med imaginære værdier omkring 120 der indikerer en 3-state cyklisk periode i plottet til venstre ikke er til stede i plottet til højre. Det kan således slås fast af de registrerede DTU individer ikke udviser samme 3-periode opførsel som (10. Juyong Park. et al.) observerer i deres datasæt.

Eigenmode analyse viser meget interessante og lovende takter i forhold til analyse og karakterisering af brugere i mindre komplekse systemer, som det præsenteret i Figure 6. I den type systemer er det muligt at inddele en brugers

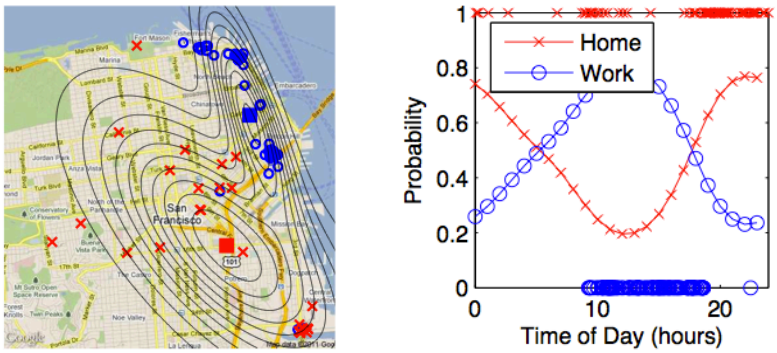
spationtremporale data i clusters med veldefinerede transitionssandsynligheder indenfor og mellem clusters. Metoderne kan desværre ikke umiddelbart generaliseres til mere komplekse systemer og kan derfor ikke anvendes på DTU-datasættet. I mere kompleks data er det sommetider muligt at identificere grundlæggende perioder i bevægelsesmønstret (se Figure 7), men anvendt på DTU-datasættet er det imidlertid ikke muligt at påvise nogen sådanne periodiske tendenser. Det spatiotemporale datasæt fra DTU har altså tilsyneladende ikke et hyppigt tilbagevendende antal states eller check-in clusters dets brugere bevæger sig periodisk imellem og det er derfor nødvendigt at finde mere fleksible metoder til at beskrive denne population.

4.3. Friendship and Mobility: User Movement In Location-Based Social Networks

(7. Eunjoon Cho. et al.) beskriver i deres paper *User Movement In Location-Based Social Network* en model til at prædiktere bevægelse for brugere af de to lokations baserede netværk Gowalla (23.) og BrightKite (24.). Modellen kaldes *Periodic and Social Mobility Model (PSMM)* og består af tre primære komponenter en temporal og en spatial der til sammen udgør hvad de kalder den periodiske model, samt en social komponent. Tilsammen giver disse komponenter altså en model der benytter informationerne i både den temporale, den rummelige og den sociale dimension tilgængeligt i et lokationsbaseret socialt netværk. Jeg vil her gennemgå tanker og teori bag hver af disse komponenter.

4.3.1. Periodisk mobilitets model (PMM)

Den periodiske mobilitets model har til opgave at indfange regelmæssigheden i en brugers spatiotemporale bevægelsesmønster og herfra konstruere en model for fremtidige periodiske bevægelser. Modellen bygger på en antagelse om at menneskelige bevægelser foregår periodisk mellem et relativt lille antal states herefter kaldet lokationer. Modellen kan håndtere et vilkårligt antal af disse lokationer, men i denne præsentation betragtes blot to. Disse to kaldes hjem og arbejde (home og work i figurene lånt fra *User Movement In Location-Based Social Network*). Afhængig tiden på dagen vil en bruger bevægelser være centreret omkring enten hjem, arbejde eller et sted der i mellem når brugeren pendler mellem de to lokationer. På Figure 8 (a) ses det hvordan en brugers hjemme og arbejds lokation (henholdsvis blå og rød) bestemmes som centroiden af alle check-ins hørende til den givne lokation og modelleres med en gaussfordeling. Den temporale fordeling af en brugerens check-ins på hjemme- og arbejdslokationen kan ses i Figure 8 (b). Denne fordeling giver sandsynligheden for at finde en bruger på en given lokation til et givent tidspunkt og illustrerer samtidig brugerens overgange mellem lokationer.



(a) Spatial model ; (b) Home/work state

Figure 8: (figuren er lånt fra (7. Eunjoon Cho. et al.))(a) En brugers spatiale fordeling af check-ins med røde krydser for arbejds- og blå cirkler for hjemme-check-ins. De udledte centroider af de to lokationer er markeret med firkanter. (b) temporal fordeling af samme check-ins.

Formelt er PMM givet ved,

Equation 6

$$P[x(t)=x]=P[x_u(t)=x\,|\,c_u(t)=H]\cdot P[c_u(t)=H]+P[x_u(t)=x\,|\,c_u(t)=W]\cdot P[c_u(t)=W]$$

hvor t er tid på dagen, $x_u(t)$ er bruger u's geografiske position til tiden t og $C_u(t)$ er den lokation brugeren befinder sig på til tiden t. Den spatiale position af en bruger's check-in er således styret fordelingen af brugerens check-ins over de to lokationer $P[x(t)=x\,|\,c_u(t)]$. Den samlede sandsynligheds fordeling over en brugers spatiale position er således blot et blanding af hjemme- og arbejdslokationen hvor blandingsfoholdet er styret af den temporale model.

4.3.2. Temporal komponent af PMM

$P[c_u(t)]$ modelleres ved en trunkeret Gaussfordeling med parameter t.

Equation 7

$$P[c_u(t)=H]=\frac{N_H(t)}{N_H(t)+N_W(t)}$$
$$P[c_u(t)=W]=\frac{N_W(t)}{N_H(t)+N_W(t)}$$

hvor

$$N_H(t)=\frac{P_{c_H}}{\sqrt{2\pi\sigma_H^2}}\exp\left[-\left(\frac{\pi}{12}\right)^2\frac{(t-\tau_H)^2}{2\sigma_H^2}\right]$$
$$N_W(t)=\frac{P_{c_W}}{\sqrt{2\pi\sigma_W^2}}\exp\left[-\left(\frac{\pi}{12}\right)^2\frac{(t-\tau_W)^2}{2\sigma_W^2}\right]$$

Her er τ_H det gennemsnitlige tidspunkt på dagen hvor brugeren befinder sig på hjemme-lokationen, og σ_H er den tilhørende spredning. P_{c_H} er den

tidsuafhængige sandsynlighed for at et givent check-in stammer fra hjemme-lokationen.

4.3.3. Spatial komponent af PMM

Den rummelige komponent modelleres som en 2-dimensionel Gaussfordeling

Equation 8

$$P[x_u(t) = x_i \mid c_u(t)] = \begin{cases} \mathrm{N}(\mu_H, \Sigma_H) & if \quad c_u(t) = H \\ \mathrm{N}(\mu_W, \Sigma_W) & if \quad c_u(t) = W \end{cases}$$

hvor Σ_H, Σ_W er de respektive covarians matricer til "hjem" og "arbejde" og μ_H og μ_W de spatiale gennemsnit.

Vi har således en model der i praksis er et mix af to Gaussfordelinger med hver deres tidsafhængige a priori sandsynlighed. Hvor den temporale komponent styrer transitioner mellem de to lokationer mens en brugers spatiale position bestemmes af den 2-dimensionelle Gaussfordeling(17. Weisstein).

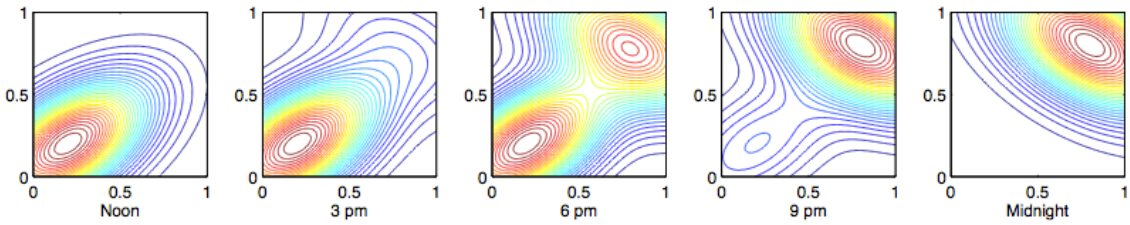


Figure 9: (figuren er lånt fra (7. Eunjoon Cho. et al.)). En brugers spatiale sandsynlighedsfordeling over tid, mens brugeren transitionerer fra "arbejde" til "hjem"

4.3.4. Social komponent af PSMM

Antagelsen der ligger til grund for den spatiotemporale komponent af PSM-modellen er at menneskelig bevægelse er rutine præget og derved indeholder nogle sptiotemporale regelmæssigheder. Mens denne antagelse langt hen ad vejen lader til at være velbegrundet så er det også indlysende at der fra tid til anden vil foretage afvigelser fra disse rutiner disse afvigelser er meget svære for ikke sige umulige at forudsige med en rutine baseret model. Derfor tilføjes en social komponent til modellen til at tage sig af netop disse situationer. Den udvidede model kaldes Periodisk og Social Mobilitetsmodel (PSMM) og er givet ved

Equation 9

$$P_u[x(t) = x] = P[x(t) = x \mid z_u(t) = 1] \cdot P[z_u(t) = 1] + P[x(t) = x \mid z_u(t) = 0] \cdot P[z_u(t) = 0]$$

hvor $z_u(t) = 1$ indikerer en social check-in og $z_u(t) = 0$ en periodisk check-in. Således er PMM givet ved $P[x(t) = x \mid z_u(t) = 0]$. Givet at bruger u foretager et

socialt check-in, er sandsynligheden for at bruger *u* checker ind på en given position x_i bestemt af to faktorer, tiden siden en ven *v,* har checket ind og afstanden fra fra *v*'s check-in til x_i. Af Figure 10 fremgår det at sandsynligheden for at en bruger foretager et check-in på en lokation hvor en ven tidligere har foretaget et check-in aftager som en potens lov af både den temporale og spatiale afstand til en bruger. Derfor modelleres den sociale komponent således

Equation 10

$$P[x_u(t) = x_i \mid z(t) = 1] \approx \sum_{(t_j, x_j) \in J_u} \left| t_j - t \right|^{-\alpha} \cdot \left\| x_i - x_j \right\|^{-\beta}$$

hvor J_u er samlingen af alle check-ins foretaget af bruger *u*'s venner på den givne dag. t_j er tiden og x_j for check-in *j,* foretaget af *u's* venner.

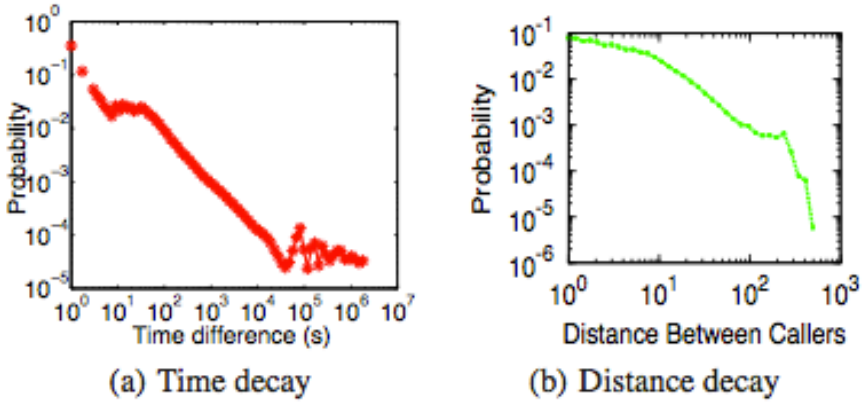


Figure 10:(figuren er lånt fra (7. Eunjoon Cho. et al.)). (a) tiden siden sidste ven foretog en chek-in på brugerens nuværende position. (b) afstanden mellem to brugere når et opkald initialiseres

PSMM modellen viser sig at give rigtig pæne prædiktions resultater og er i stand til at bestemme eksakte bruger positioner i op mod 40% af alle forsøg, hvilket er ganske godt i betragtning af de relativt støjfyldte datasæt. Da modellen samtidig udnytter information fra de tre dimensioner af information tilgængelige i DTU-datasættet (spatial, temporal og social), virker modellen som et godt grundlag for en prædiktionsmodel optimeret til netop dette datasæt. I beskrivelsen af hvordan PSM-modellen kan tilpasses til DTU-datasættet vil der blive gået yderligere i detaljer med mange af de ideer og koncepter kort præsenteret i dette afsnit, men først følger en undersøgelse af hvilken prædiktionspræcision der kan forventes på baggrund af den tilgængelige spatiotemporale information i DTU-datasættet.

4.4. Øvre grænser for forudsigelighed af menneskelig bevægelse

Ved at følge metoden først introduceret af (3. Chaoming Song. et al.) vil vi nu undersøge hvor nøjagtigt vi kan forvente at kunne forudsige et individs position frem i tiden baseret på dennes historiske bevægelsesdata. Metoden basere sig på entropien i et individs bevægelsesdata og kan herfra bestemme den øvre grænse

for forudsigeligheden af dette individ given en algoritme der udnytter informationen i den historiske data optimalt.

4.4.1. Entropi

Først introduceres tre entropi mål til beskrivelse af forudsigeligheden af en tidsserie som de behandlede device trajecctories fra DTU.

Tilfældig entropi, S_i^{rand}

Equation 11

$$S_i^{rand} = \log_2 N_i$$

hvor N_i er antallet af unikke lokationer besøgt af bruger *i.* Tilfældig entropi beskriver forudsigeligheden af brugerens position i det tilfælde hvor alle lokationer besøges med lige stor sandsynlighed.

Temporal ukorreleret entropi, S_i^{unc}

Equation 12

$$S_i^{unc} = -\sum_{i=1}^{N_i} P_i(j) \cdot \log_2 P_i(j)$$

hvor $P_i(j)$ er den historiske sandsynlighed for at lokation *j* blev besøgt af bruger i. Temporal-uncorrelated entropy karakterisere brugerens spatiale lokations-besøgsmønster.

Og endelig den faktiske entropi, S_i, der afhænger ikke bare af frekvensen af en brugers besøg til en lokation, men også rækkefølgen med hvilken disse er blevet besøgt og mængden af tid brugt på hver lokation (der ligeledes kan betragtes som besøgsrækkefølge hvis to på hinanden følgende besøg til samme lokation bruges til at beskrive længerevarende besøg på en lokation), hvorved den fulde spatiotemporale regelmæssighed i en brugers bevægelsesmønster indfanges.

Faktisk entropi, S_i

Equation 13

$$S_i = -\sum_{T_i^{'} \subset T_i} P\left(T_i^{'}\right) \log_2\left(P\left(T_i^{'}\right)\right)$$

her er $P\left(T_i^{'}\right)$ sandsynligheden for at finde en tidsordnet subserie $T_i^{'}$ i bruger *i's* trajectory, $T_i = \left\{X_1, X_2, \cdots, X_L\right\}$.

Det forholder sig naturligvis sådan at entropien falder i takt med at først den spatiale- (i S_i^{unc}) og dernæst også den temporale regelmæssighed (i S_i) indfanges i entropi beregningerne. Derfor gælder at

Equation 14

$$S_i \leq S_i^{unc} \leq S_i^{rand}$$

Da udregningen af $P(T_i')$ kan være relativt beregningstung anvendes en algoritme til approksimeret beregning af S_i , baseret på Lempel-Ziv data kompression [4](2. Chaoming Song. et al. (support)). Algoritmen er kendt for hurtigt at konvergere mod den faktiske entropi i en tids serier med et relativt stort antal obervationer. For en tids serie med *n* trin ser entropi beregningen sådan ud

Equation 15

$$S^{est} = \left(\frac{1}{n} \sum_i \Lambda_i \right)^{-1} \ln(n)$$

her er Λ_i den korteste subserie startende i position *i* der ikke allerede er fundet i position *1* til *i-1*. Det kan bevises at S^{est} går mod den faktiske entropi for *n* gående mod uendelige (2. Chaoming Song. et al. (support)).

For at konceptet i de tidsordnede subserier T_i' skal give mening skal den behandlede tidsserie være samplet i regelmæssige intervaller. Dette skyldes at den vigtige information der ligger i den temporale distance mellem to observationer ellers vil gå tabt. For at kunne transformere de behandlede bruger trajectories til uniforme tidsserier skal to udfordringer overkommes. For det første findes der ingen information om brugers position når denne befinder sig unden for DTU's campus og for det andet samplingen af brugernes position uregelmæssig når de befinder sig på campus. Løsningen på den manglende information om brugere der befinder sig uden for campus løses ved at indføre en lokation kaldet *off campus*, hvor brugeren antages at befinde sig hver gang denne ikke er at finde på campus. Den uregelmæssige sampling frekvens løses ved at tælle alle check-ins der falder inden for en given klokke time og give brugeren en samlet position i hele denne time der svare til den check-in position der registreret oftest i den givne time. Hvis ingen check-ins er registreret indenfor en given klokketime betragtes brugenren's position som værende *off campus*. Som det fremgår af Figure 11 og Figure 12 så falder den faktiske og ukorrelerede entropi ganske betragteligt, mens den tilfældige entropi er næsten uændret når *off campus* registreringer medtages i entropi beregningerne. Denne opførsel giver god mening idet den tilfældige entropi kun afhænger af antallet af unikke besøgte lokationer der for de fleste brugere ligger mellem 10 og 100 hvorfor tilføjelsen af en enkelt ekstra lokation efter logaritmen af dette antal er blevet beregnet kun giver en meget lille forøgelse. Når det kommer til den temporalt ukorrelerede betyder tilføjelsen af en ekstra lokation med så høj besøgsfrekvens

som *off campus* en lavere entropi. *Off campus* har ikke bare en høj besøgsfrekvens, men også en meget regelmæssig temporal opførsel idet *off campus* registrering oftest vil falde i lange ubrudte serier i aften og natte timerne, hvilket forklarer den lavere faktiske entropi når *off campus* registreringer medtages.

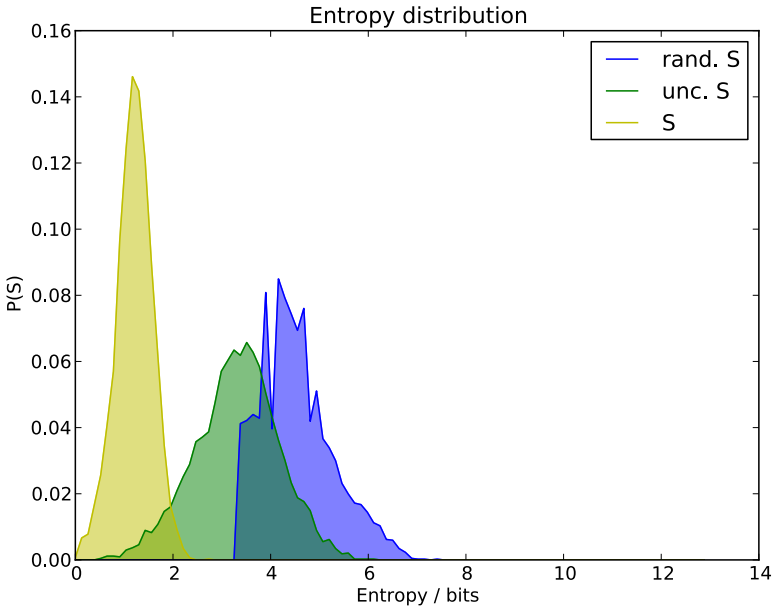


Figure 11: De tre entropi-mål S_i, S_i^{unc} *og* S_i^{rand} som de fordeler sig for de 4367 brugere i datasættet. Her er *off campus* registreringer udeladt på trods af den lille fejlfaktor der introduceres ved at den første registeringer tirsdag morgen betragtes som følgende lige efter den sidste registrering mandag eftermiddag.

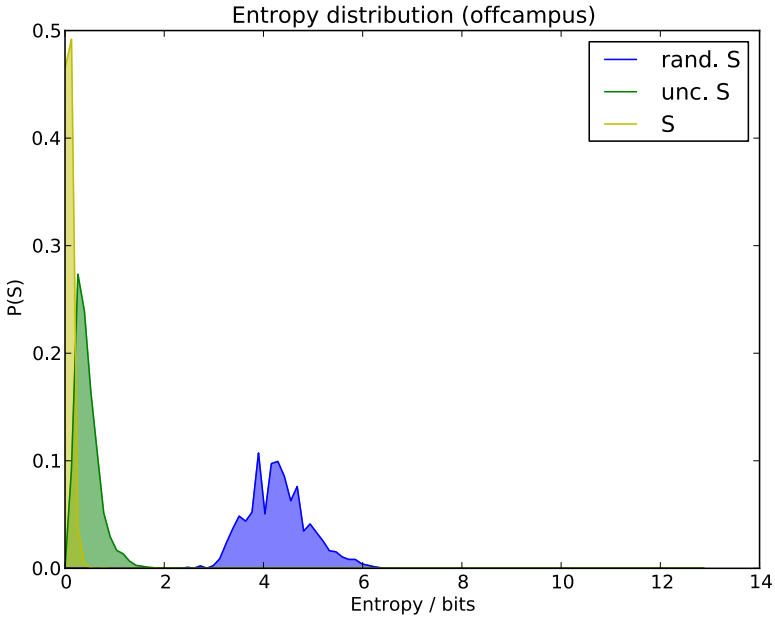


Figure 12: Her er fordelingen af de tre entropi mål for hele datasættet igen afbilledet, men denne gang er *off campus* registreringer også inkluderet.

4.4.2. Fundamentale grænser for forudsigelighed

Med entropi beregningerne vel udført bevæger vi os nu videre til det egentlige mål nemlig at fastslå de øvre grænser for forudsigeligheden af en tidsserie. (12. N. Navet. et al.)Den nedre entropigrænse er per definition *S=0* denne entropi tilstand kan tolkes som fuldstændig orden, hvilket i tilfældet hvor vi forsøger at forudsige en brugers position betyder fuld forudsigelighed. I den anden ende toppes skalaen af $S = S^{rand} = \log_2(N)$ der beskriver tilfældet når en brugers bane er fuldstændig tilfældig mellem *N* lokationer. I dette tilfælde er den maksimale forudsigelsespræcision på *1/N*. Det er mellem disse to ekstremer langt de fleste (sandsynligvis alle) brugere skal findes hvilket indikerer at de behandlede bevægelsesmønstre indeholder elementer af tilfældighed, men også regelmæssighed der kan udnyttes til at forudsige en brugers position. I det følgende vil jeg præsentere en metode til at kvantificere grænserne for forudsigeligheden af en brugers bevægelsesmønster, på baggrund af entropien i dette mønster. Metoden stammer fra (3. Chaoming Song. et al.) og er baseret på Fano's ulighed (12. N. Navet. et al.)(14. Fano). Metoden giver et mål, Π^{Max}, for den øvre grænse for forudsigeligheden af en brugers position på baggrund af dennes historiske bevægelsesmønster givet en algoritme der optimalt kan udnytte information i dette mønster. $\Pi^{Max}(S,N)$ er givet ved løsningen på ligningen

Equation 16

$$S = -\left[\Pi^{Max}\log_2(\Pi^{Max}) + (1-\Pi^{Max})\log_2(1-\Pi^{Max})\right] + (1-\Pi^{Max})\log_2(N-1)$$

hvor *S* er entropien i brugerens historiske bevægelsesmønster og *N* er antallet af unikke accesspoints besøgt af denne bruger. Equation 16 kan ikke løses analytisk, hvorfor det er nødvendigt numerisk at udregne værdierne af Π^{Max} for hver af de tre entropimål S^{rand},S^{unc} *og S.* Da de numeriske udregningerne Π^{Max} ikke blot skal udregnes en enkelt gang for hver bruger og hver entropi type og ikke gentagende gange i den endelige prædiktionsmetode valgte jeg selv at implementere en numerisk løsnings algoritme baseret på Euler metoden(1. Ascher. et al. & Petzold) . Af Figure 13 og Figure 14 fremgår det at det med den rette algoritme vil være muligt korrekt forudsige en DTU brugers position i gennemsnitligt 85% af tiden når denne befinder sig på campus. Hvis *off campus* medtages bliver dette tal i den høje ende af halvfemserne. Det betyder ikke at det bliver nemmere at forudsige en brugers position på campus hvis *off campus* registreringer medtages med blot den samlede sandsynlighed for korrekt at forudsige en brugers position bliver højere i kraft af at en bruger med næsten 100% nøjagtighed kan forudsiges at være *off campus* i aften- og nattetimerne.

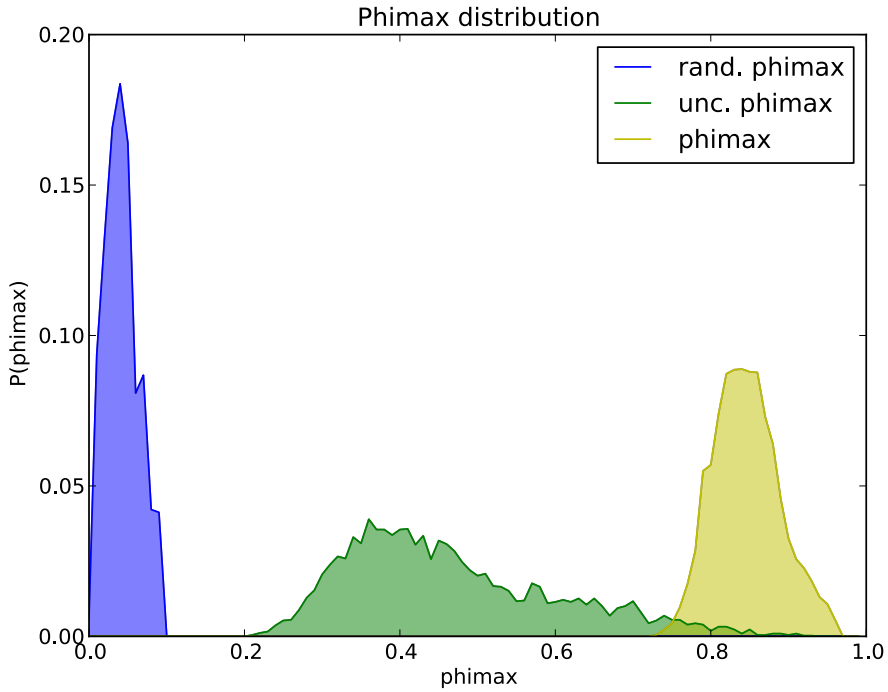


Figure 13: Det ses tydeligt at der er stor forskel på den maksimale forudsigelighed afhængig af om en bruger bevægelsesmønster betragtes som tilfældige bevægelse mellem et antal accesspoints (blå), om spatiale regelmæssigheder tages med (grøn) eller både spatiale og temporale regelmæssigheder inkluderes (gul).

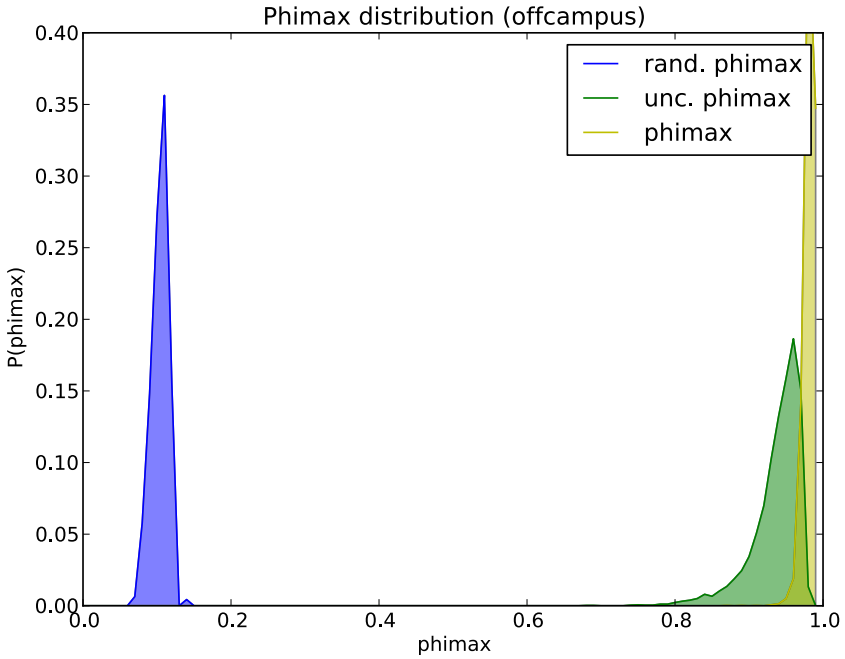


Figure 14: Af plottet kan vi se at forudsigeligheden af brugerne, beregnet ud fra den temporalt ukorrelerede entropi og den faktiske entropi, stiger betragteligt når *off campus* registreringer tages med i beregningerne. Igen ser vi at inkluderingen af *off campus* registreringer ikke har nogen bemærkelsesværdig effekt på den tilfældige entropi og derfor heller ikke på π^{Max} beregnet på baggrund heraf.

5. Prædiktions modeller PMM og PSMM tilpasset DTU

I dette afsnit beskrives hvordan PSMM som beskrevet i afsnittet Friendship and Mobility: User Movement In Location-Based Social Networks, ændres, tilpasses og optimeres til at kunne modellere DTU-datasættet. Afsnittet indeholder både teoretisk og praktisk beskrivelse af den implementerede version af PSMM, samt evaluering af modellens performance i praksis. Afsnittet er bygget op på følgende måde først præsenteres den rent periodiske model PMM, dernæst følger en præsentation og diskussion af hvordan den sociale komponent konstrueres og tilføjes til PMM for at danne den samlede Periodiske og sociale mobilitets model. Herefter følger en beskrivelse af parameter optimering for PSMM og endelig følge ren evaluering af modellens performance i praksis.

5.1. Den periodiske mobilitets model (PMM)

Den periodiske mobilitets model er som nævnt i afsnittet Friendship and Mobility: User Movement In Location-Based Social Networks, en metode til at konstruere sandsynlighedsfordelingen $P[x_u(t) = x]$ der til tiden *t*, giver sandsynligheden for at finde bruger *u* på en given position *x*. Modellen er baseret på *N* states (states vil herefter også blive omtalt som lokationer) der hver især har en position og udstrækning i tid og rum. Eksempelvis kunne en state være området omkring kantinen på DTU med midten af spisesalen som sin spatiale position, mens det temporale udtrækning kunne være 2 timer omkring den temporale position kl. 12 (mere om definition og konstruktion af states senere). Modellen består således af en spatial og en temporal komponent, hvor den temporale komponent bestemmer den spatiale sammensætning af states til et givent tidspunkt. Sandsynlighedsfordelingen $P[x_u(t) = x]$ er givet ved,

Equation 17

$$P[x_u(t) = x] = \sum_{i=1}^{N} P[x_u(t) = x \mid c_u(t) = L_i] \cdot P[c_u(t) = L_i]$$

hvor $c(t)$ er bidraget fra state L_i til tiden *t* og *N* er antallet af states. Equation 17 er en modifikation Equation 6 generaliseret fra 2 til *N* states.

5.1.1. Tilpasning og optimering af PMM i forhold til oprindelig form

Da jeg ønsker så detaljeret modellering af en brugers bevægelsesmønster på campus som muligt har jeg valgt at generalisere fra de oprindelige 2 til et vilkårligt antal modellerede states. Udover ønsket om øget detaljegrad skyldes generaliseringen at det ikke umiddelbart er muligt at argumentere for et bestemt antal states der vil være naturligt at modellere DTU-populationen over. I

afsnittet Træning af model – Expectation Maximization findes en berskrivelse af hvordan antallet af modellerede states bestemmes for den enkelte bruger.

I kraft af den ugentlige periodicitet der intuitivt findes hos de fleste mennesker der har fast arbejde eller følger et fuldtidsstudie (se Figure 35) og for den sags skyld de fleste andre mennesker, har jeg desuden valgt at konstruere en temporal model der modellere en fuld uge frem for det ene døgn der var tilfældet i den oprindelige model.

5.1.2. Temporal komponent af PMM

til tiden t befinder en bruger sig i state L_i med sandsynligheden $P[c_u(t) = L_i]$ den temporale fordeling $P[c_u(t)]$ der beskriver den tidsafhængige sandsynlighed for at befinde sig i hver af de definerede states er givet ved

Equation 18

$$P[c_u(t) = L_x] = \frac{N_{L_x}(t)}{\sum_{i=1}^{N} N_{L_i}(t)}$$

hvor $N_{L_x}(t)$ er givet ved

Equation 19

$$N_{L_x}(t) = \sum_{i=1}^{7} \frac{D_{L_x i} P_{L_x}}{\sqrt{2\pi\left(\frac{\sigma_{L_x i}}{v}\right)^2}} \exp\left[-\left(\frac{\pi}{12}\right)^2 \frac{(t-\tau_{L_x i})^2}{2\left(\frac{\sigma_{L_x i}}{v}\right)^2}\right]$$

som det fremgår af ovenstående definition af $N_{L_x}(t)$ er denne givet ved en sammensat funktion af 7 unikke gaussfordelinger, en for hver af ugens 7 dage. Hver daglig gaussfordeling afhænger af tre parametre der er unikke for den aktuelle ugedag, nemlig τ_{L_x}, σ_{L_x} og $D_{L_x i}$, samt to parametre P_{L_x} og v der er uafhængig af ugedagen. τ_{L_x} er det gennemsnitlige tidspunkt på dagen hvor bruger u befinder sig i state L_i, mens σ_{L_x} er den tilhørende spredning. I udregninger af temporalt gennemsnit og spredning betragtes hver dag som cyklisk, hvilket betyder at der mellem kl. 23.59 og 00.01 kun er to minutters tidsforskel. Gennemsnit og spredning bliver således udregnet som var check-in tiderne vinkler i en enheds cirkel. Denne behandling er nødvendig for at kunne håndtere gennemsnit der ligger i nærheden af skiftet mellem to dage (kl. 24.00). Eksempelvis vil en ikke-cyklisk behandling af en bruger med regelmæssige check-ins på en given lokation mellem kl. 20.00 og 06.00 (et tidsrum hvor de fleste vil befinde sig i deres hjem) ved almindelig udregning give et gennemsnit omkring kl. 13.00, hvilket tydeligvis ikke stemmer overens med det opfattede gennemsnitstidspunkt for ophold i hjemmet. Ved at betragte dagen som cyklisk

og udregne gennemsnit i overensstemmelse hermed bliver gennemsnittet i stedet kl. 01.00, hvilket passer bedre med vores intuition. $D_{L_x i}$ er en parameter der bruges til at vægte den enkelte dag i forhold til antallet af check-ins på lokation L_x der falder på netop denne ugedag i forhold til det samlede antal check-ins på L_x. P_{L_x} er en vægtning af den enkelte lokationen L_x, givet ved antallet check-ins til denne lokation i forhold til det samlede antal check-ins på alle lokationer. Endelig er der parameter v som er en konstant til justering fordelingens spredning efter skiftet til den 24 timers cykliske periode.

Equation 18 og Equation 19 er baseret på Equation 7. Equation 18 er generaliseret fra 2 til N states, mens Equation 19 er modificeret fra at i den oprindelige form (Equation 7) kun at modellere en brugers temporale fordeling over et enkelt døgn til at modellere en uge.

Da det benyttede datasæt ikke indeholder information om brugernes bevægelser uden for DTU's campus har det som tidligere nævnt været nødvendigt at indføre en '*off campus*'-lokation hvor brugeren antages at være positioneret når en hvis tærskel er overskredet for tiden siden sidste check-in. Denne lokation har i kraft af den manglende information ingen geografisk position, men fungerer udelukkende som en temporal state. De kunstigt genererede '*off campus*' check-ins er markeret med lilla på Figure 15.

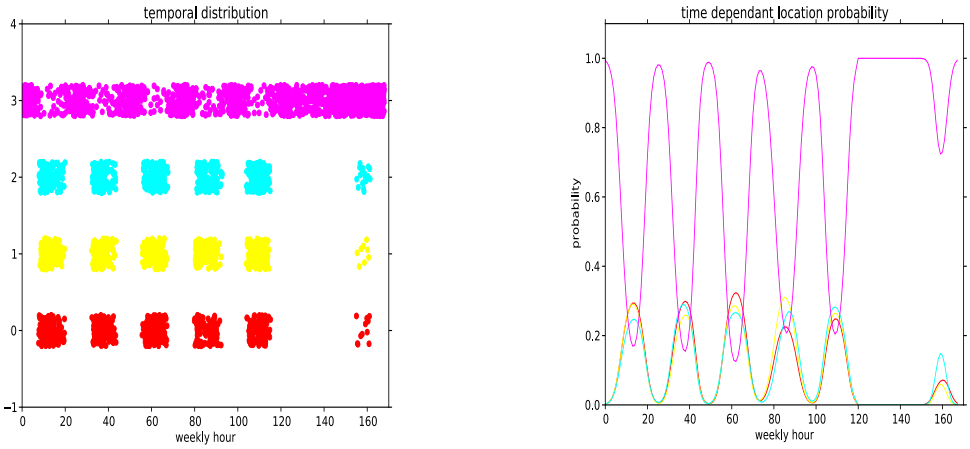


Figure 15: Til venstre ses den temporale komponent af en brugers check-ins som de fordeler sig over ugens 168 timer. Her er de fordelt tilfældigt mellem tre lokationer angivet med rød, gul og tyrkis. Den specielle off campus lokation er angivet med lilla. På plottet til højre ses hvordan Equation 18 transformerer den temporale data til den tidsafhængige sandsynligheds fordeling der giver sandsynligheden for at befinde sig på en given lokation til tiden *t*.

5.1.3. Off campus lokationen

Da DTU-datasættet kun indeholder observationer af DTU-brugerne når disse befinder sig på campus er det nødvendigt at tage stilling til behandlingen af de længerevarende tidsrum mellem observationer der opstår i bl.a. aften- og nattetimerne når brugerne ikke befinder sig på campus. Jeg har således valgt at definere en state kaldet *off campus*, hvori brugeren forestilles indeholdt når

denne ikke kan observeres på campus. Da der ikke findes nogen tilgængelig information om brugernes privat adresser, er der ikke noget grundlag for at definere denne *off campus* state i rummet. Derfor har jeg valgt at betragte *off campus* som en udelukkende temporal lokation uden hverken position eller udtrækning i rummet. Hermed undgås det at *off campus* lokationen påvirker den spatiale *on campus*-sandsynlighedsfordeling.

Off campus lokationen er vigtigt element i den temporale sandsynlighedsfordeling og er afgørende for hvornår PMM og PSMM bestemmer en brugers position som værende on eller off campus. Da *off campus* lokationen skal indgå i et datasæt der består af diskrete observationer af en brugers position i tid og rum er det nødvendigt at efterligne denne struktur i de simulerede *off campus* observationer. I praksis bliver en bruger således betragtet som værende *off campus* hvis denne i mere 90 minutter ikke er blevet observeret på campus. Herefter registreres en ny *off campus* observation hvert 90. minut i den almindelige studie- og arbejdstid mellem kl. 8.00 og kl. 20.00. og hvert 30 minut udenfor denne periode. Denne skelnen mellem almindelig studie- og arbejdstid er indført for at nedtone effekten af situationer hvor en bruger af den ene eller anden grund befinder sig på campus, men ikke har et aktivt WiFi device i nærheden.

5.1.4. Spatial komponent af PMM

Den tidsuafhængige rummelige komponent af PMM modelleres ved en 2-dimensionel Gaussfordeling da dette er en almindelig brugt metode til at modellere menneskelig bevægelse omkring et punkt i rummet (30. D. Brockmann. et al.)(31. M. C. Gonzalez et al.),

Equation 20 (17. Weisstein)

$$P[x_u(t) = x_i \mid c_u(t)] = \frac{1}{2\pi\sigma_{x1}\sigma_{x2}\sqrt{1-\rho^2}} \exp\left[-\frac{z}{2(1-\rho)}\right]$$

hvor

$$z = \frac{(x_{i1} - \mu_{L_x 1})^2}{\sigma_{L_x 1}^2} - \frac{2\rho_{L_x}(x_{i1} - \mu_{L_x 1})(x_{i2} - \mu_{L_x 2})}{\sigma_{L_x 1}\sigma_{L_x 2}} + \frac{(x_{L_x 2} - \mu_{L_x 2})^2}{\sigma_{L_x 2}^2}$$

og

$$\rho = \frac{\mathrm{cov}(X_{L_x 1}, X_{L_x 2})}{\sigma_{L_x 1}\sigma_{L_x 2}}$$

startende fra bunden har vi $X_{L_x 1}$ og $X_{L_x 2}$ der repræsentere første og anden koordinaterne på alle check-ins fra bruger u på lokation L_x. $\sigma_{L_x 1}$ og $\sigma_{L_x 2}$ er spredningen af disse check-ins første og anden koordinat, mens $\mu_{L_x 1}$ og $\mu_{L_x 2}$ er de korresponderende middelværdier.

På Figure 16 ses et eksempel på den resulterende spatiale sandsynlighedsfordeling til et givent tidspunkt *t,* samt den tilhørende temporale fordeling.

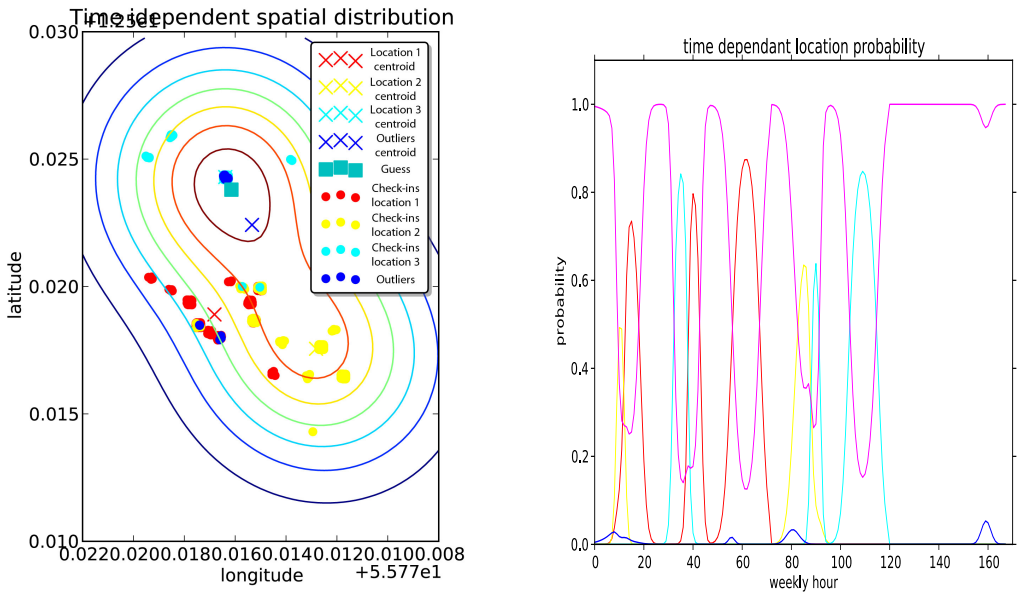


Figure 16: Til venstre ses et contourplot af sandsynlighedsfordelingen for en brugers position på campus torsdag lige omkring frokost. Hvis vi sammenligner med den temporale sandsynlighedsfordeling fremgår det at den spatiale sandsynlighedsfordelings tyngdepunkt er ved at flytte sig fra lokation 2 (gul) til lokation 3 (tyrkis). Af contourplottet (venstre) kan det ses hvordan den spatiale sandsynlighedsfordeling til det givne tidspunkt er sammensat af to 2-dimensionelle gauss-fordelinger en centreret i lokation 2's centroide (gult kryds) og en centreret i lokation 3's centroid (tyrkis kryds) i overensstemmelse med den temporale fordeling (højre).

5.2. Social model

Som nævnt i afsnittet Periodisk mobilitets model (PMM) udnytter den Periodiske mobilitets model PMM de spatiale og temporale rutiner der findes i stort set alle menneskers hverdagsbevægelsesmønstre. At basere en model på disse rutiner giver på samme tid et meget stærkt grundlag for forudsigelse af en brugers rutine-bevægelser og et meget ringe grundlag for at forudsige "ikke-rutine"-bevægelser. I et forsøg på at gøre modellen mere robust overfor bevægelser der falder uden for de almindelige rutiner vender vi blikket mod andre rutiner denne gang ikke i tid og rum men snarere i den sociale dimension. For lige så vel som de fleste mennesker har en tendens til at besøge de samme geografiske lokationer samme tid på dagen eller ugen har vi almindeligvis også en tendens til at opholde og bevæge os med en begrænset gruppe af sociale relationer. Hvad enten disse er venner, familie eller kollegaer kan vi bruge disse sammenfald til at forudsige brugerens bevægelser ud fra de sociale relationers' bevægelse. Inddragelsen af sociale relationer i modellen forudsætter tre kriterier er opfyldt for at kunne anvendes succesfuldt i modellen

K1. Information om det sociale netværk en bruger indgår i
K2. Information om de sociale relationers position i tid og rum i perioden hvor brugerens bevægelser skal forudsiges
K3. Brugerne, modellen arbejder med, skal foretage en anselig del af deres "ikke-rutine" bevægelser sammen med deres sociale relationer

Information om det sociale netværk og dermed opfyldelsen af kriterium 1 kan opfyldes på to måder enten explicit hvor brugeren selv rapporterer sine sociale relationer som det kendes fra online sociale/profesionelle netværk som Facebook, Google+, LinkedIn, m.fl. Eller implicit fra den tilgængelige information og struktur i det behandlede data. I DTU tilfældet findes der ingen explicit information om sociale relationer hvorfor et implicit approach er valgt. Mens en implicit generering af det sociale netværk baseret på spatiotemporale data kan have svært ved at identificere relationer der primært dyrkes uden for DTU eller opretholdes gennem distance kommunikation så har det den fordel at sociale relationer bliver identificeret med formålet at bidrage til bestemmelsen af en brugers position for øje. Dette fjerner behovet for udvælge de mest relevante sociale relationer sammen med behovet for at skelne mellem forskellige former for sociale relationer som f.eks. venner og kollegaer.

Hvis man forestiller sig at den anvendte periodiske og sociale mobilitets model (PSMM) skal anvendes til at forudsige bruger bevægelse frem i tiden vil der af gode grunde ikke være spatiotemporal information tilgængelig for forudsigelsesperioden. Man kunne forestille sig et approach hvor de sociale relationers position ligeledes blev estimeret frem i tiden, men i kraft af at denne estimering nødvendigvis ville være genereret med PMM model vil alle "ikke-rutine" bevægelser forsvinde fra modellen og dermed hele ideen med at introducere den sociale dimension. Hvis man derimod ønsker at modellen bruges til at bestemme en brugers position i real time hvor dennes position ikke er kendt, enten p.g.a. mangel på aktivt WiFi device eller manglende WiFi-dækning kan dette lade sig gøre i og med de sociale relationers position er kendte op til bestemmelsestidspunktet. I det præsenterede tilfælde er det foreliggende DTU-datasæt blevet delt i to en del til træning af modellen og en til evaluering, hvilket betyder at spatiotemporal data for en brugers sociale relationer er tilgængelige i evaluerings/forudsigelses perioden og dermed at kriterium 2 er opfyldt.

Lad os nu tænke over rimeligheden i at kriterium 3 skulle være opfyldt ved at overveje hvilke situationer der kan anses for sandsynlige årsager til at afvige fra hverdagens spatiale og temporale rutiner som studerende eller ansat på DTU.

- Ekstraordinært socialt eller fagligt event så som messe, fest, udstilling, ekstraordinær forelæsning.
- Aflysning eller ændring af forelæsning eller anden undervisning.
- Frokost på ny lokation sammen med en social relation
- Overarbejde og ekstra gruppearbejde op til afleveringsdeadlines og eksaminer

Hver af de ovenstående begivenheder vil med rimelig sandsynlighed foregå sammen med en social relation. Mens dette på ingen måde er en udtømmende liste over mulige årsager til rutinebrud så er alle, plausible situationer involverende sociale relationer som kan forsage rutinebrud. I det følgende vil vi se nærmere på hvordan sociale relationer kan bestemmes fra et spatiotemporalt datasæt og hvordan disse relationer kan anvendes til at bestemme en brugers position under disse rutinebrud.

5.3. Implicit bestemmelse af sociale relationer på baggrund af spatietemporalt datasæt

Den første basale antagelse der er nødvendig for implicit at kunne bestemme sociale relationer fra et spatial-temporalt datasæt er at der er sammenhæng mellem de sociale relationer mellem to individer og sammenfald i deres placering i tid og rum. Dette er en almindelig anerkendt antagelse der er blevet påvist i en række videnskabelige artikler i de seneste år (9. Justin Cranshaw. et al.), (6. David J. Crandalla. et al.) og som beskrevet i afsnittet Sociale relationer og Co-occurences. I vores moderne verden findes der naturligvis talrige kommunikationsredskaber så som telefon, mail, chat, online sociale netværk, m.fl. der gør det muligt at fungere i sociale og professionelle relationer på afstand, men disse relationer vil i denne opgave blive set bort fra. Desuden gør den geografiske afgrænsning af det observerede område (DTU's campus i Lyngby), at langdistance relationer ikke bliver relevante i forhold til dette datasæt.

5.3.1. Co-occurences

Grundlaget for at bestemme en social relation mellem to brugere implicit, fra spatio temporal data, er placeringssammenfald i tid og rum mellem disse to. Et sådan sammenfald betegnes herefter co-occurence (beskrevet i afsnittet Sociale relationer og Co-occurences), og defineres for DTU-datasættet som observationen af to brugere på samme accesspoint med mindre end 60 minutters mellemrum. *Off campus* co-occurrences medregnes ikke.

5.3.2. Co-occurence signifikans og Friendscore

I erkendelse af at en co-occurence mellem to brugere kl. 2 torsdag nat i en databar er mere signifikant i forhold til en eventuel social relation mellem de to end en co-occurence mandag kl. 12.30 i kantinen, vil hver registreret co-occurence blive tildelt en vægt baseret på et tidsafhængigt entropimål $Entropi(A,t)$, tilhørende det involverede accesspoint. *Entropi(A,t)* er en tidsafhængig version af det lokationsentropi udtryk præsenteret af (9. Justin Cranshaw. et al.)(se afsnit Signifikans af co-occurences), der ikke blot tager højde for hvor en co-occurrence finder sted men også hvornår. $Entropi(A,t)$ er givet ved,

Equation 21

$$-\sum_{u \in U_A} P_A(u,t) \cdot \log_2\left(P_A(u,t)\right)$$

hvor $P_A(u,t)$ er den totale andel af alle check-ins på accesspoint A foretaget i tidsrummet t, af bruger u og U_A sættet af alle device check-ins på accesspoint A

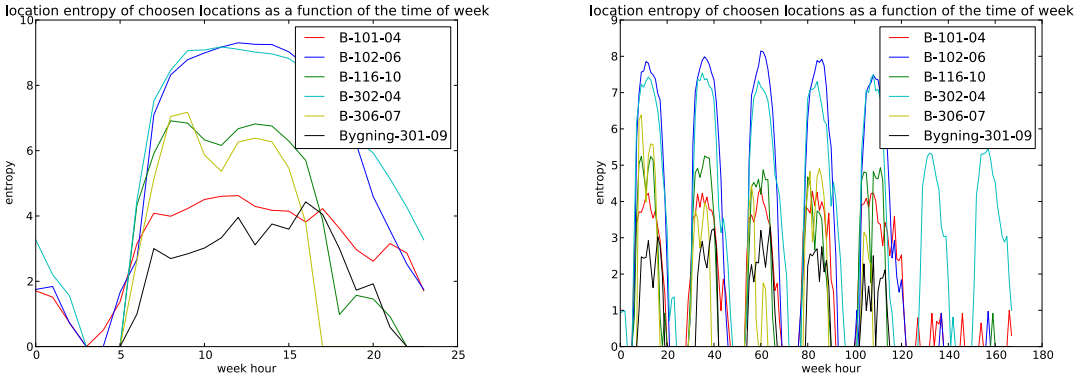


Figure 17: på plottet til venstre ses accesspoint entropien for 6 udvalgte accesspoints over en periode på 24 timer. Det ses tydeligt hvordan entropien er højest i de travle timer midt på dagen mens den er lavest i de sene aften- og tidlige morgentimer. På plottet til højre ses accesspoint entropien for de samme 6 accesspoint, men her over en periode på 168 timer (1 uge). Her ses igen høje entropi værdier i de almindelige arbejds-/undervisningstimer for dem 5 hverdage og næsten forsvindende entropi for 5 af de 6 accesspoints i weekenden, på det sidste accesspoint er der fortsat relativt høj aktivitet og dermed høj entropi lørdag og søndag.

Friendscore bliver således summen af disse tidsafhængige accesspoint entropier divideret med antallet af co-occurrences eller bare det gennemsnitlige co-occurrence entropi.

Equation 22

$$FS(f) \frac{|CO|}{\sum_{(A,t) \in CO} Entropi(A,t)}$$

her er CO samlingen af alle co-occurrences mellem brugeren og social relation f givet ved tidspunkt og accesspoint for hændelsen. I det konkrete tilfælde, hvor pointen med at identificere sociale relationer er at kunne bestemme en brugers position ud fra positionen af dennes sociale relationer. Kan man argumentere for at det er mest hensigtsmæssigt at lade brugeren påvirke kraftigst af den relation der har den højeste rå sandsynlighed (antallet af co-occurrences) for at befinde sig sammen med brugeren. At jeg alligevel vælger at benytte friendscore til at styre styrken af en social relation skyldes følgende ræsonnement baseret på de to hypoteser,

H1: Sociale relationer med mange co-occurrences, men med lav friendscore, *s1*, er personer eksempelvis fra samme studieretning, som følger samme kurser men ellers ikke har nogen social relation til brugeren. Sociale relationer med høj

friendscore, *s2,* er derimod personer som brugeren eksplicit vil angive som venner.

H2: Rutine brud af typen beskrevet under kriterium K3 har større sandsynlighed for at ske med en relation af typen *s2* end med en relation af typen *s1* der blot har mange kurser til fælles med brugeren.

(lignende eksempler kan gives for personer der er fastansatte på DTU, men disse er udeladt p.g.a. kortfattedhed)

Givet at H1 og H2 er sande har vi at på trods af at en given bruger muligvis har et større antal co-occurrences til fælles med relationer af typen *s1*, så er sandsynligheden større for finde brugeren sammen med relationer af typen *s2* under rutinebrud. Og da det er i netop disse situationer vi ønsker at den sociale komponent supplerer den periodiske, vælges friendscore frem for antallet af co-occurrences som den styrende faktor for styrken af sociale relationer.

5.3.3. Tærskler og begrænsninger

For at acceptere en social relation mellem to individer skal antallet af registrerde co-occurences svare til mindst 20% af det samlede antal registreringer for den af de to der har færrest registreringer. Fastsættelsen af denne grænse skete efter en vurdering af at en gennemsnitlig DTU-bruger har en omgangskreds (andre DTU brugere der ses på daglig eller ugentlig basis) på mellem 10 og 20 personer. Efter 'trial and error'-princippet kom jeg frem til at netop dette interval af gennemsntilige relationer blev opnået ved en co-occurence tærskel på mellem ca. 15%-30%. Jeg valgte således 20% fordi det svare til at dele 1 ud af de 5 hverdage med brugeren og gav et gennemsnitligt antal venner på ca. 12. For at undgå situationer hvor der bestemmes en social relation mellem to individer der eksempelvis følger kurser i to tilstødende foredragslokaler der deler accesspoint, men derudover ikke har noget til fælles, må højst 80% co-occurences mellem to individer stamme fra samme accesspoint.

5.4. Inkludering af sociale relationer i den Periodiske mobilitets model

For at kunne udnytte den sociale dimension i en brugers bevægelsesmønster til at forbedre præcisionen af PMM er det nødvendigt at gøre sig en række overvejelser i forbindelse med denne udvidelse. De vigtigste af disse overvejelser kan opsummeres således

- Hvordan defineres og genkendes en social begrundet bevægelse
- Hvordan modelleres og optimeres den sociale sandsynlighedsfordeling
- Og endelig hvordan vægtes den sociale sandsynlighedsfordeling i forhold til den periodiske

5.4.1. Hvordan defineres og genkendes en social begrundet bevægelse - Outliers

Hvis vi vender tilbage til motivationen for at introducere den sociale dimension i prædiktionsmodellen var denne at forbedre forudsigelser der falder uden for de periodiske bevægelser. Fra et statistisk synspunkt svarer dette til at finde outliers til den periodiske model. Ifølge (8. Grubbs) er outliers defineret således

> *"An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs"*

Der findes mange forskellige matematiske metoder til at konkret at bestemme outliers , jeg har valgt at benytte Chauvenet's kriterium (16. Taylor) da dette er intuitivt og kan udregnes fra værdier allerede kendt i modellen.

Equation 23

$$outlier(X) = 1 \quad if \quad f(X) \cdot N < 0.5 \wedge x < \mu_{outlierThreshold} \quad else \quad outlier(X) = 0$$

hvor X er den observation i tid og rum der testes, N er det totale antal af observationer og f en normalfordelingen herunder, med $\mu_{outlierThreshold}$ og $\sigma_{outlierThreshold}$ værende henholdsvis gennemsnit og spredning af observationerne.

Equation 24

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2_{outlierThreshold}}} \exp\left(-\frac{(x - \mu_{outlierThreshold})^2}{2\sigma^2_{outlierThreshold}} \right)$$

Konkret bestemmes outliers ved at lade den færdigtrænede model gennemløbe træningssættet igen og registrere modellens sandsynligheds output for hver af de kendte check-ins. Herefter beregnes gennemsnit og spredning af de registrerede outputs. Disse to værdier udgør henholdsvis $\mu_{outlierThreshold}$, $\sigma_{outlierThreshold}$ i Equation 23, mens antallet af check-ins i træningssættet giver $N_{outlierThreshold}$. Det anden begrænsning i Equation 23 sikrer at det kun er exceptionelt små og værdier af x der klassificeres som outliers og ikke de ualmindeligt sikre og dermed store værdier af x.

5.4.2. Hvordan modelleres den sociale sandsynlighedsfordeling

For at bestemme hvordan de sociale relationers påvirkning af brugerne bedst modelleres i tid og rum undersøges et temporalt og et spatialt mål for de sociale relationers bevægelser i forhold til brugeren for 100 tilfældigt udvalgte brugere. Det temporale mål skal bestemme hvilken indflydelse tiden siden en social relation sidst foretog et check-in på et accesspoint har på sandsynligheden for at en co-occurrence finder sted på netop dette accesspoint. Dette mål defineres således som den temporale afstand mellem en co-occurrence og den involverede sociale relations sidste check-in på samme accesspoint. Det spatiale mål har til formål at bestemme hvordan afstanden mellem brugeren og en social relation har til indflydelse på sandsynligheden for en co-occurrence og defineres som

afstanden mellem bruger og social relation ved deres respektive check-ins umiddelbart før en co-occurrence finder sted.

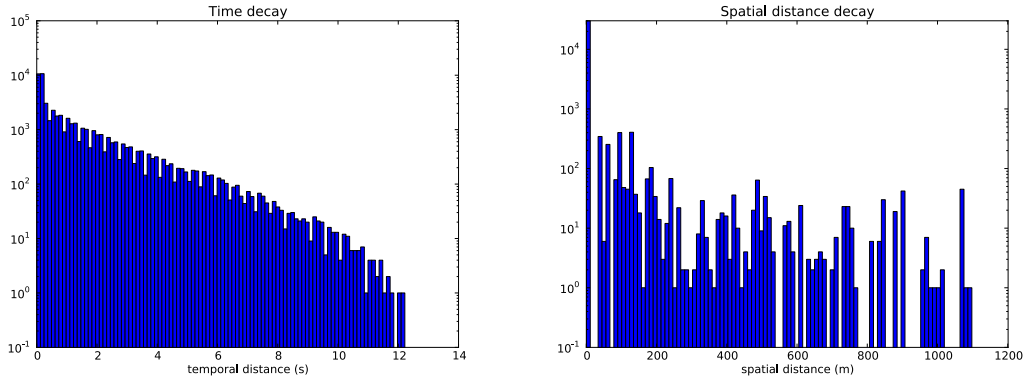


Figure 18: Plottet til venstre viser antallet af co-occurrences mellem bruger og sociale relationer som funktion af tiden siden den involverede sociale relation sidst foretog et check-in på det accesspoint hvor co-occurrencen fandt sted. Plottets y-akse er logaritmisk. Til højre ses et histogram over antallet af co-occurrences som funktion af afstanden mellem bruger og social relation ved deres respektive sidste check-ins før en co-occurrence. y-aksen er igen logaritmisk.

Af Figure 18 kan det ses at histogrammet til venstre tilnærmelsesvist kan beskrives ved en aftagende ret linje i det semilogaritmiske plot hvilket betyder at sandsynligheden for en co-occurence falder som en potensfunktion af tiden siden den sociale relation sidst checkede ind på det relevante accesspoint. I plottet til højre er der sværere at se en direkte sammenhæng mellem afstanden mellem bruger og social relation og sandsynligheden for en co-occurrence det er dog bemærkelsesværdigt at der ved en afstand på 0 meter er ca. en faktor hundrede større sandsynlighed for en efterfølgende co-occurence end ved nogen andre afstande. Jeg vælger derfor at modellere den spatiale afstands indflydelse på sandsynligheden for en co-occurence som en delta funktion med værdi 1 når ven og social relation befinder sig på samme rummelige position og 0 i alle andre tilfælde. Mens det er en vigtig observation at sandsynligheden for en co-occurrence er betydelige højere umiddelbart efter en bruger er observeret på samme accesspoint som en social relation er det dog på ingen måde overraskende. At man har større sandsynlighed for at blive sammen med en social relation man allerede er i kontakt i stedet for spontant at bevæge sig rundt mellem sociale relationer på forskellige positioner i rummet stemmer fint overens med hvad mad ville forvente af en almindelig menneskelig adfærd.

Af Figure 18 har vi således at sociale relationers påvirkning af en bruger bedst modelleres som potensfunktion i tid og en delta funktion i rummet. Det samlede udtryk for de sociale relationers påvirkning af en bruger bliver således.

Equation 25

$$\sum_{(t_i, x_i) \in J_f} \left| t_j - t \right|^{-\alpha} \cdot \delta_{x_i x_j}$$

hvor

$$\delta_{x_i x_j} = \begin{cases} 1, & if\ x_i = x_j \\ 0, & if\ x_i \neq x_j \end{cases}$$

I Praksis betyder denne modellering af sociale check-ins at en bruger har større sandsynlighed for at checke ind på et accesspoint der befinder sig tæt, i tid og rum, på en ven.

Modellen for sociale relationer afviger i den spatiale komponent fra modellen anvendt i (7. Eunjoon Cho. et al.). Denne afvigelse sker på baggrund af observationerne i det spatiale plot Figure 18 og kan have flere årsager. De to mest nærliggende er for det første at det geografiske område (DTU's campus) hvor dataindsamlingen er begrænset til er så relativt lille at alle sociale relationer der befinder sig inden for det observerede område er inden for en overskuelig afstand hvilket betyder at de kan og ifølge observationer i Figure 18, bliver opsøgt uafhængigt af afstanden til dem. Den anden nærliggende årsag til den observerede afvigelse ligger i det sociale netværk. Hvor der hos (7. Eunjoon Cho. et al.) er tale om eksplicit defineret socialt netværk der må forventes primært at indeholde sociale relationer af typen man vil betragte som venner, er det i DTU tilfældet et implicit genereret netværk hvor de fleste identificerede relationer må forventes at være studiekammerater/kollegaer. Det er ikke utænkeligt at bevægelsemønstre motiveret af venner er forskellige fra bevægelsesmønstre motiveret at studiekammerater/kollegaer.

5.5. Hvordan vægtes den sociale sandsynlighedsfordeling i forhold til den periodiske

Med både en periodisk- og en social-komponent til at bestemme en brugers position er det af afgørende betydning at bestemme om de to modeller skal bruges parallelt og dermed hvornår den peridodiske- og hvornår den sociale-komponent skal anvendes. Eller om de to komponenter skal bruges serielt og dermed hvordan de to modeller skal vægtes i forhold til hinanden. Kort sagt kan man sige at den serielle sammensætning af den periodiske- og sociale model giver en flydende kontinuert overgang mellem de to komponenter, mens den parallelle sammensætning giver en overgang der kan beskrives ved en deltafunktion. Begge fremgangsmåder har sine fordele og ulemper, hvorfor begge vil blive forfulgt teoretisk og implementeret i modellen for på den måde at undersøge hvilken af de to der giver PSMM den bedste performance.

5.5.1. Parallel kombination af periodisk og social komponent

Hvis vi igen vender tilbage til motivation for at introducere den sociale dimension hvor ideen var at finde en alternativ model til at beskrive ikke-periodiske bevægelser er det intuitivt at vælge en løsning hvor ikke periodiske bevægelser identificeres, udvælges og modelleres udelukkende med den sociale komponent, altså en parallel løsning. Dermed kommer man også uden om

udfordringen der ligger i at, videnskabeligt, skulle bestemme et vægt forhold mellem den periodiske og sociale komponent. Som det er blevet gjort klart gennem hele model beskrivelsen så er de periodiske bevægelser og dermed den periodiske komponent det primære metode til beskrive en brugers bevægelser. Derfor er det også som udgangspunkt den periodiske model som bruges til at bestemme en brugers position. Til at bestemme hvornår en positions bestemmelse skal betragtes som social anvendes igen outlier konceptet. Det vil sige at såfremt en positionsbestemmelse ved brug af den periodiske komponent vurderes til at være så usikker at den betragtes som en outlier, vil den sociale komponent blive anvendt til bestemmelsen. Grænsen for hvornår et positionsgæt betragtes som en outlier antages at være den samme som den grænse der blev udregnet under identifikationen af outliers i træningssættet.

5.5.2. Seriel kombination af periodisk og social komponent

På den anden side har den serielle løsning kan også sine fordele for det første fordi man kan undgå at skulle finde et kriterium der kan ligge til grund for en skarp vurdering af om det periodiske gæt er en outlier. Chauvenet's kriterium er blot en metode til bestemmelse af outliers der måske og måske ikke kan bestemme meningsfulde outliers i forhold til PSM-modellens behov. For det andet fordi at man i hvert positionsbestemmelse får et bidrag fra både den periodiske og sociale komponent hvilket kan være en fordel i en række situationer. Eksempelvis når både det periodiske og det sociale bidrag er meget usikkert, eller når det periodiske bidrag er relativt usikkert, men akkurat ikke nok til at have krydset outlier-tærsklen, mens det sociale bidrag er meget sikkert (sikre og usikre referere til mener jeg henholdsvis høje og lave maksimale sandsynlighedsværdier se Positionsestimering fra PSMM-sandsynlighed).

For at bestemme et passende vægtforhold mellem den sociale og periodiske komponent i PSM-modellen, vender vi igen tilbage til formålet med at indføre en den sociale komponent. Ideen var at den sociale model skulle være dominerende i de situationer hvor den periodiske model var usikker, og det er med netop dette mål for øje at vægtforholdet mellem de to komponenter bestemmes. Vi skal altså finde et vægtforhold der på den ene side gør den sociale komponent stærk nok til at være den styrende komponent ved outliers men, samtidig ikke forstyrre den periodiske komponent under bestemmelsen af periodiske bevægelser. I praksis bliver vægtforholdet indført som faktoren W_{social} der ganges på bidraget fra den sociale komponent.

Outliers til den periodiske model er tidligere blevet bestemt for træningssættet og den uvægtede ratio $r_{outlier}$ mellem bidraget fra den periodiske og sociale komponent kan nu udregnes for hver outlier. Det samlede sæt af $r_{outlier}$ kaldes $R_{outlier}$ og giver os størrelsesforholdet mellem bidragene fra den periodiske og den sociale komponent i de tilfælde hvor vi ønsker at have den sociale komponent som den styrende.

Equation 26

$$R_{outlier} = \sum_{c_{outliers} \in C_{outlier}} \frac{P_{periodisk}[x_u(t) = c_{outlier}]}{P_{social}[x_u(t) = c_{outlier}]}$$

Hvor $C_{outlier}$ er sættet af alle check-ins ,$c_{outlier}$,klassificeret som outliers. Samme ratio udregnes for de periodisk velbeskrevne punkter, og benævnes $r_{periodisk}$ og sættet af periodiske ratii $R_{periodisk}$.

Equation 27

$$R = \sum_{c \in C} \frac{P_{periodisk}[x_u(t) = c]}{P_{social}[x_u(t) = c]}$$

Hvor C er sættet af alle periodisk velbeskrevne check-ins ,c. Således kender vi også forholdet mellem de periodiske og sociale bidrag i de tilfælde hvor vi ønsker at den periodiske komponent skal være styrende. For både $R_{outlier}$ og $R_{periodisk}$ fjernes bidrag hvor $P_{social} = 0$.

Nu udregnes gennemsnit $\mu_{outlier}$ og $\mu_{periodisk}$ og spredning $\sigma_{outlier}$ og $\sigma_{periodisk}$ for hendholdsvis $R_{outlier}$ og $R_{periodisk}$. I langt de fleste tilfælde vil det gælde at $\mu_{outlier} < \mu_{periodisk}$ i det outliers er defineret som de ualmindeligt små værdier af den periodiske komponent (se afsnit: Hvordan defineres og genkendes en social begrundet bevægelse - Outliers) . Hvis dette skulle vise sig ikke at være tilfældet refittes PM-modellen. I meget sjældne tilfælde hvor dette gentager sig mere end 5 gange evalueres modellen som rent periodisk, da den tilgængelige data for de sociale relationer ikke er tilstrækkelig til at underbygge den sociale komponent.

Fra $\mu_{outlier}$ og $\mu_{periodisk}$ og $\sigma_{outlier}$ og $\sigma_{periodisk}$ kostrueres nomalfordelingerne $N_{outlier}$ og $N_{periodisk}$ og W_{social} defineres som skæringspunktet mellem disse såfremt et sådan eksistere i intervallet mellem $\mu_{outlier}$ og $\mu_{periodisk}$. I modsat fald udregnes gennemsnittet af de to middelværdier.

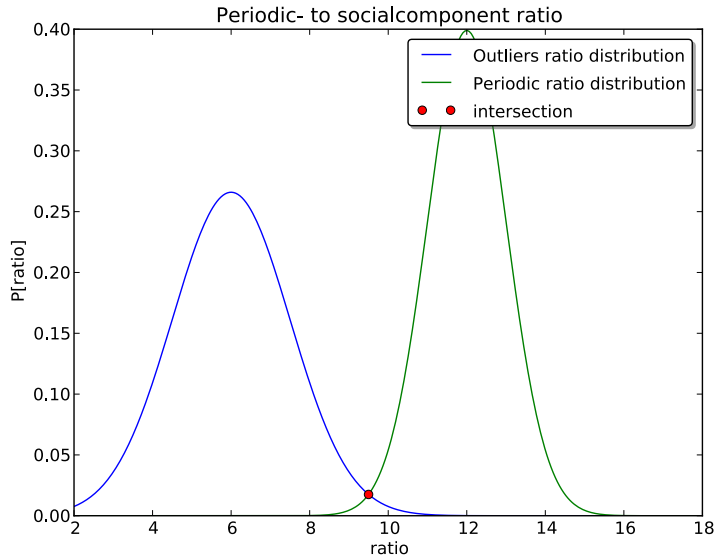


Figure 19: Et eksempel på fordelingen af ratio mellem bidrag fra den periodiske og sociale komponent for henholdsvid outlier (blå) og periodisk beskrevne check-ins (grøn)

ved at bestemme middelværdien som skæringspunktet mellem den periodiske fordeling og outlier fordelingen findes netop det vægtforhold der har størst sandsynlighed for outlier check-ins bliver bestemt med en styrende social komponent mens periodisk velbeskrevne check-ins bliver bestemt med en styrende periodisk komponent.

5.6. Periodisk og social mobilitets model (PSMM)

Det samlede matematiske udtryk for den periodiske og sociale model bliver således enten

Equation 28 parallel

$$P[x_u(t) = x] = P_{social}[x_u(t) = x] \cdot outlier(x,t) + P_{periodisk}[x_u(t) = x] \cdot (1 - outlier(x,t))$$

eller

Equation 29 seriel

$$P[x_u(t) = x] = W_{social} \cdot P_{social}[x_u(t) = x] + P_{periodisk}[x_u(t) = x]$$

Hvor $P_{periodisk}$ repræsenterer bidraget fra PMM og P_{social} den sociale komponents bidrag. W_{social} er vægtforholdet mellem den periodiske og sociale komponent, og *outlier(x,t)* er funktionen beskrevet i Equation 23, der bestemmer om en given position i tid og rum kan klassificeres som en outlier. Givet at en brugers check-in kategoriseres som socialt, det vil sige at positionsgættet fra PMM-modellen er så usikkert at det kategoriseres som en outliers, er sandsynligheden for bruger *u* vil checke ind på position x_i til tiden *t* er styret af to faktorer nemlig hvor lang tid

der er gået siden en ven har checket ind og hvor dette check-in fandt sted. Sandsynligheden er givet ved

Equation 30

$$P_{social}[x_u(t) = x_i \mid z(t) = 1] = \sum_{J_f \in F} \left(FS(f) \cdot \sum_{(t_j, x_j) \in J_f} \left| t_j - t \right|^{-\alpha} \cdot \delta_{x_i x_j} \right)$$

hvor J_f er samlingen af ven *f* check-ins mellem klokken 12 midnat og *t* på dagen hvor $x_u(t)$ forsøges bestemt. *F* er samlinger af J_f for alle en brugers venner. α og β er parametre der bestemmes under træning af modellen (se afsnittet Træning af model – Expectation Maximization*).* $FS(f)$ (Equation 22) er den såkaldte friendscore der fungerer som en vægtning af hver enkelt ven afhængig af antallet af co-occurrences mellem bruger og ven samt signifikansen af disse.

5.7. Træning af model – Expectation Maximization

Den Periodiske og sociale mobilitets model indeholder alt efter antallet af tilladte lokationer mellem 22 og mere end 100 parametre der skal fittes før modellen kan tages i brug. Den temporale komponent af PMM indeholder pr. lokation 7 daglige vægte D_{L_v}, 1 lokations vægt P_{L_v} og 1 temporalt gennemsnit τ_{L_v} og tilhørende spredning σ_{L_v}. Derudover den fælles parameter *v,* til spredningsjustering. Den spatiale komponent har pr. lokation et gennemsnit μ_x (μ_y) og tilhørende spredning σ_x (σ_y) for hver af de to dimensioner, samt covarians mellem dimensionerne $\mathrm{cov}(x,y)$. Den sociale komponent har 6 parametre uafhængigt af antallet af lokationer. Tre parametre der indgår i identifikationen af outliers, $\mu_{outlierThreshold}$, $\sigma_{outlierThreshold}$ og $N_{outlierThreshold}$, samt tre parametre der indgår i den sociale sandsynlighedsfordeling α, β og *FS* for hver af den givne brugers socialerelationer.

Med undtagelse af *v* og *FS* fittes alle parametre ved hjælp af en variation af Expectation Maximization kendt som hard EM. Parametrene fittes af to gange først fittes PM-modellen og derefter den samlede PSM-model. Fitting sker på et træningssæt der udgør 8 ud af de 13 ugers data der er tilgængelige for hver bruger. EM er en iterativ metode med to overordnede step i hver iteration

- Hver lokations position og udtrækning i tid of rum udregnes fra de indeholdte check-ins
- Alle check-ins omfordeles til de lokationer hvor de ifølge modellen har stærkest tilhørsforhold

I step 1 udregnes D_{L_v}, P_{L_v}, τ_{L_v}, σ_{L_v}, μ_x, μ_y, σ_x, σ_y og $\mathrm{cov}(x,y)$ ud fra de check-ins der er indeholdt i hver af de definerede lokationer. Samlet set kan disse parametre ses som definitionen af hver lokations position og udstrækning i tid og rum. I step 2 gennemløbes alle check-ins og tildeles til den lokation hvortil de

har det stærkeste tilhørsforhold i tid og rum. Styrken af et tilhørsforhold af et check-in x, til lokation L_x, er givet ved

Equation 31

$$P[x_u(t) = x] = P[x_u(t) = x \mid c_u(t) = L_x] \cdot P[c_u(t) = L_x]$$

Equation 31 er en simplificeret version Equation 17 kun indeholdende et enkelt element svarende til 1 lokation L_x. De to step gentages indtil metoden konvergerer. Fittingmetodens største svaghed er at den kun konvergere mod et lokalt maximum som er meget afhængigt af start tilstanden. Derfor inddeles check-ins, før fitting algoritmens start, tilfældigt mellem et givent antal lokationer. På den måde kan algoritmen let gentages fra et vilkårligt antal forskellige startpositioner og den fitting der giver det bedste resultat kan herefter vælges. Det bedste fitting resultat er defineret som det resultat der maksimere Equation 32. I den konkrete implementation starter deles alle check-ins i træningsættet tilfældigt i 10 lige store grupper hvorfra EM-algoritment tager udgangspunkt. Algoritmen tillades herfra at konvergere mod det antal lokationer der findes at optimere Equation 32. Dette vil altid være lavere end eller lig antallet af lokationer i udgangspunktet og altid være større end 0.

Equation 32

$$\sum_{i \in CT} P[x_u(t) = x_i]$$

hvor $P[x_u(t) = x]$ er givet ved Equation 17 og CT er sættet alle check-ins i træningsættet.

Efter PM-modellens paramametre er blevet fittet, fittes den sociale komponent. At udelade den sociale komponent fra den iterative fitting af PMM, er et valg truffet på baggrund af den sociale komponents rolle som "skraldespand" for den periodiske i situationer positions bestemmelse ved PMM er for usikkert. Det er derfor også naturligt først at optimere den periodiske model så vidt muligt og derefter identificere de svageste punkter i modellen og fitte den sociale model til at supplere i netop disse situationer. Fitting af den sociale komponent sker således ved at lade den færdigtrænede PM-model gennemløbe træningssættet igen og beregne outliers som beskrevet i afsnittet *Hvordan defineres og genkendes en social begrundet bevægelse – Outliers* og derefter udregne parametrene α, β på baggrund af disse (metode beskrevet i afnittet *Hvordan modelleres og optimeres den sociale sandsynlighedsfordeling*). Den sidste parameter i den sociale komponent, *FS*, er uafhængig af PM-modellen og antages udregnet før modellen parameter optimering igangsættes (se afsnittet: *Implicit bestemmelse af sociale relationer på baggrund af spatietemporalt datasæt*).

5.8. Visualiseret eksempel på EM forløb

For at klargøre EM forløbet for de enkelte komponenter i PSM-modellen følger her et konkret visualiseret eksempel på hvordan EM iterativt optimerer modellen i tid, rum og den sociale dimension. De tre figurer viser udvalgte iterationer af et optimerings forløb for en enkelt bruger set fra forskellige synspunkter.

5.8.1. EM – temporal check-in fordeling

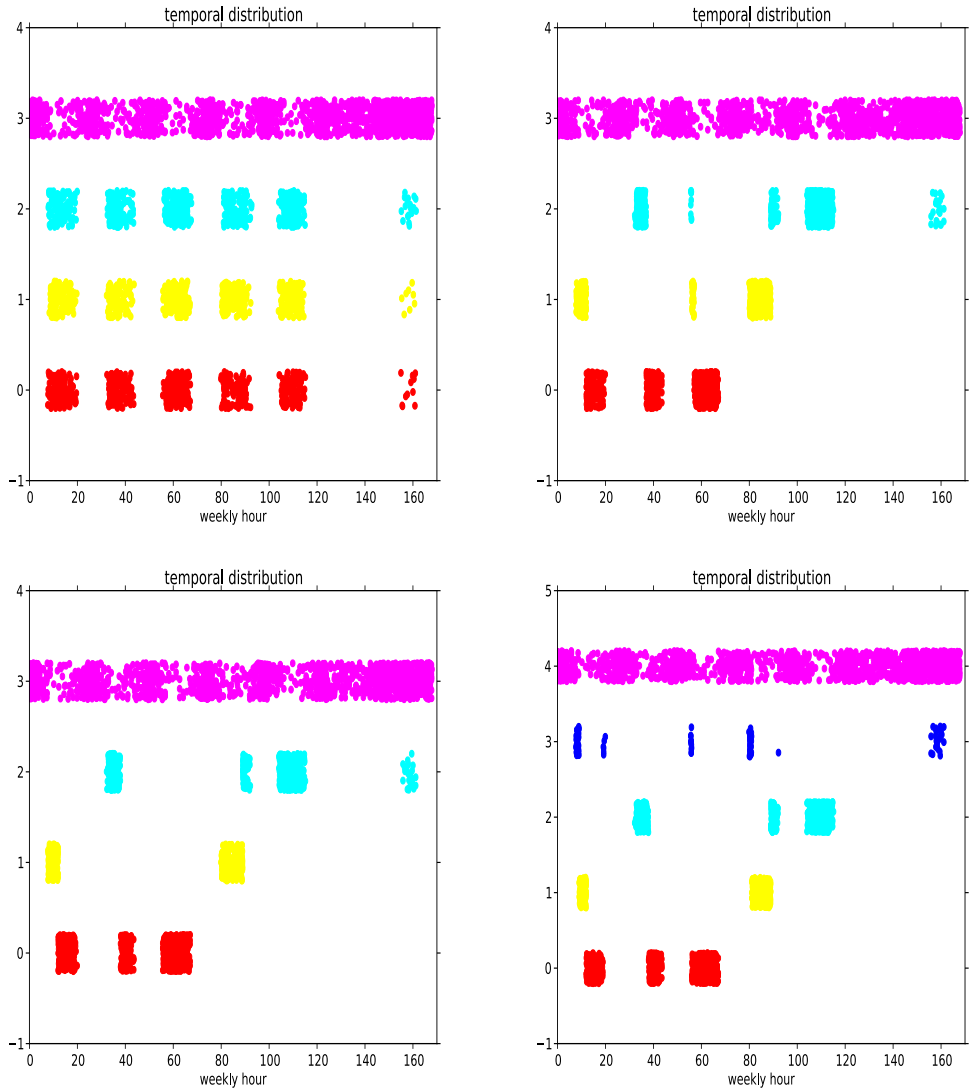


Figure 20: Figuren viser den temporale komponent af en brugers ugentlige check-ins i løbet af de 8 uger der bruges som træning for modellen og hvordan EM-metoden omfordeler disse check-ins mellem 3 lokationer. Optimerings forløbet løber fra øverst venstre mod nederst højre og viser en situation med tre ordinære lokationer rød, gul og tyrkis, samt off campus (lilla). Det ses tydeligt hvordan check-ins fra alle ugens hverdage er ligeligt fordelt mellem de tre lokationer på plottet til venstre, mens der på plottet nederst til venstre er foretaget en klar opdeling således at check-ins tilhørende forskellige lokationer stort set ikke overlapper i tid. På plottet længst til højre er outliers blevet identificeret for hver af lokationerne og samlet i en fælles ekstra lokation (blå). Det ses tydeligt hvordan de ordinære lokationers temporale fordelinger er blevet skarpere I deres afgrænsning efter dette indgreb.

5.8.2. EM – Temporal sandsynlighedsfordeling

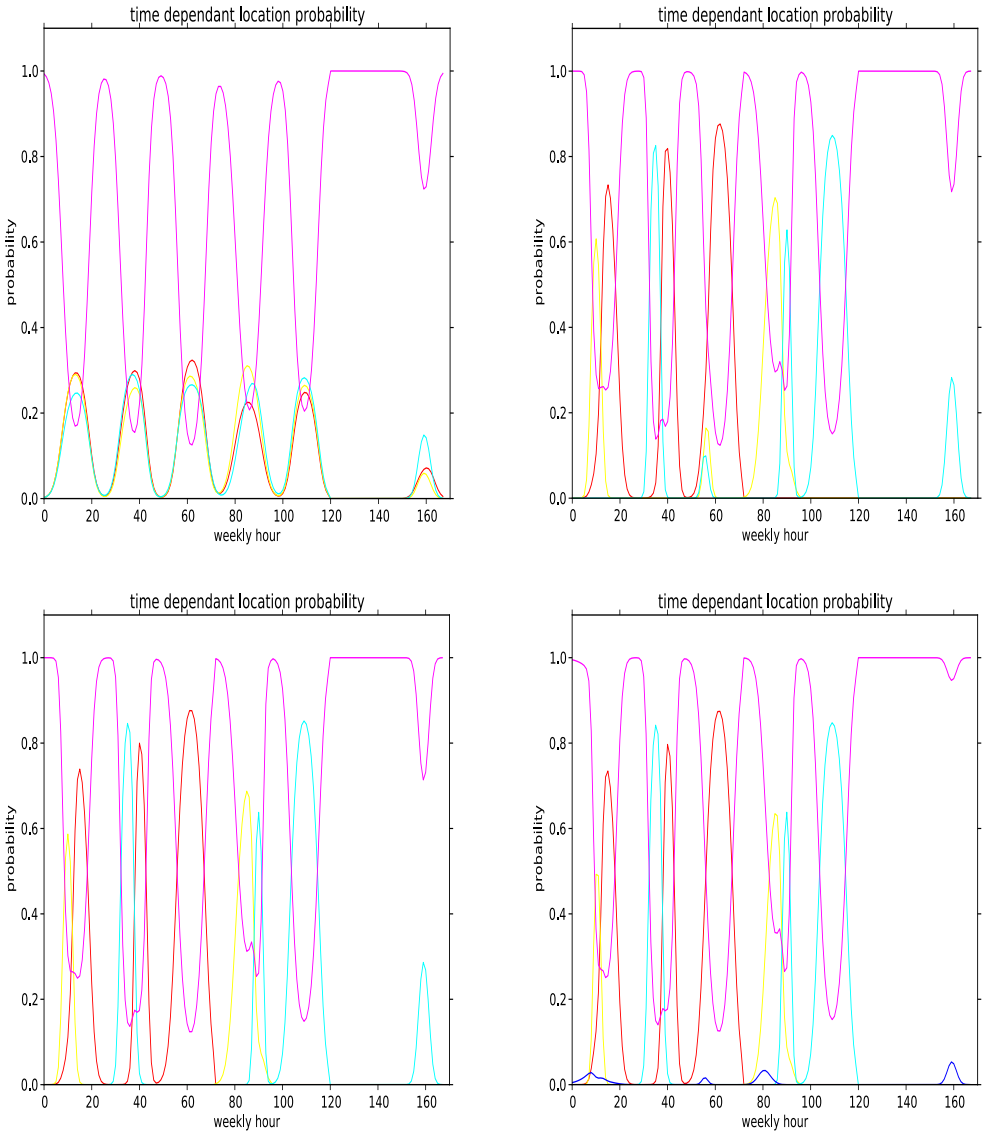


Figure 21: Fra den temporale data på Figure 20 genereres ovenstående sandsynlighedsfordeling ved hjælp af Equation 18 og Equation 19. Hver af de fire fordelinger herover er genereret fra den tilsvarende rå temporale data på Figure 20. Ved at lade off campus (lilla) indgå i fordelingen som brugerens state når denne ikke er kendt, er det muligt at skabe en fuldendt sandsynlighedsfordeling for brugerens opholdslokation der til alle tidspunkter, summere til 1.

5.8.3. EM – Spatial fordeling

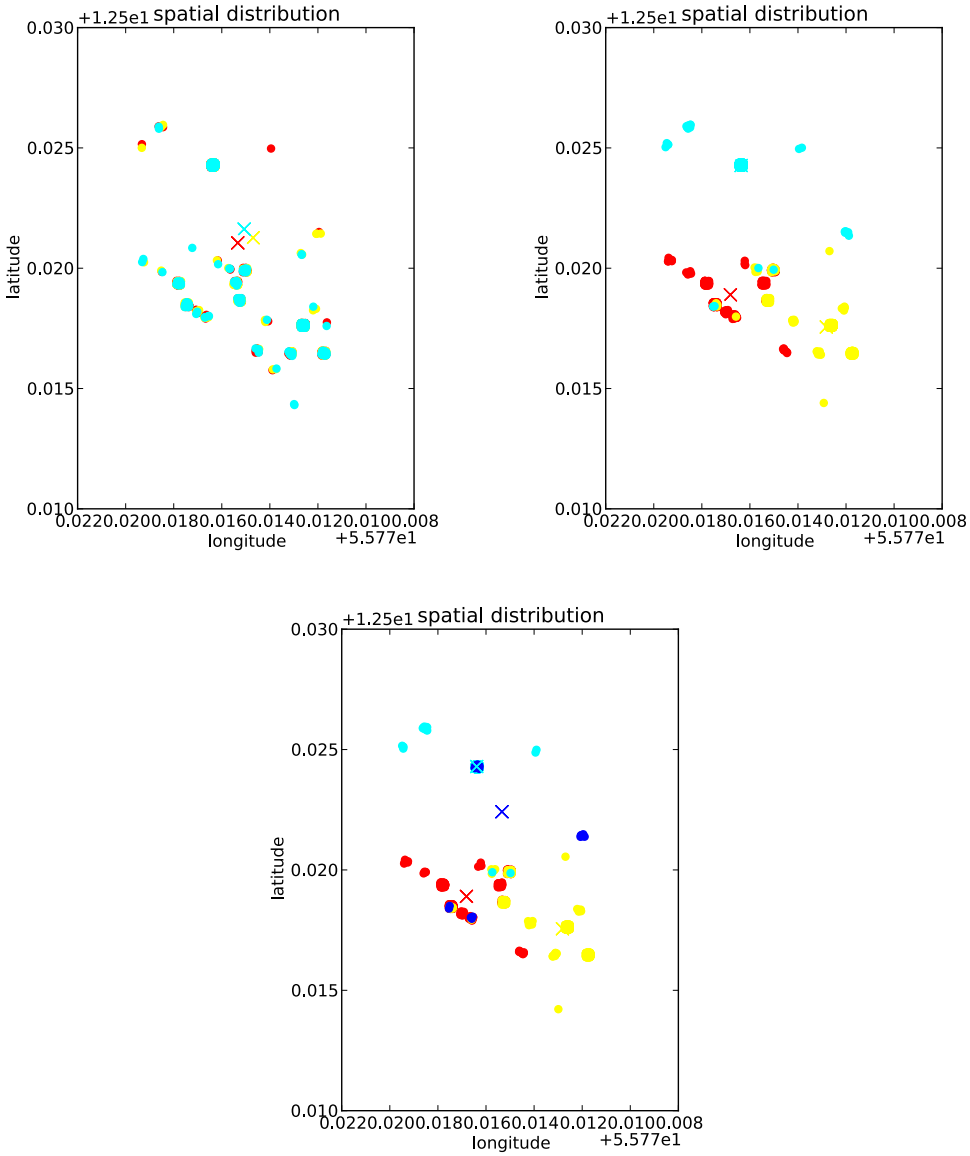


Figure 22: På figuren ses den spatiale fordeling af en brugers check-ins i den 8 uger lange træningsperiode. På plottet øverst til venstre ser vi at den første tilfældige inddeling af check-ins på de 3 lokationer (tyrkis, gul, rød) giver 3 stort set ens lokationer i rummet lige såvel som det var tilfældet i tid (Figure 20). Plottet øverst til højre viser hvordan lokationers spatiale gennemsnit (markeret med kryds) og tilhørende check-ins fordeler sig i mere eller mindre skarpt afgrænsede clusters i rummet. Længst til højre ses hvordan outlier identifikationen ser ud fra et spatialt synspunkt, I dette tilfælde er det klart at flere af de identificerede outliers må være outliers i kraft af deres temporale komponent snarere end den spatiale siden de befinder tæt på lokationernes spatiale midtpunkt.

5.9. Positionsestimering fra PSMM-sandsynlighed

Vi har nu en trænet PSM-model, der udover information om sociale relationers position i og umiddelbart før prædiktions øjeblikket kun tager tiden *t* som input. Outputtet af modellen er en tidsuafhængig sandsynlighedsfordeling for brugerens position i rummet. I praksis er vi imidlertid interesseret i et at modellen giver en konkret position X_{guess}, i stedet for en sandsynlighedsfordeling som output. Derfor bestemmes den mest sandsynlige brugerposition til tiden t_0, ved at finde det globale maksimum af sandsynlighedsfordelingen.

Equation 33

$$P[x_u(t_0) = x_i] = \sum_{i=1}^{N} \frac{1}{2\pi\sigma_{x_i 1}\sigma_{x_i 1}\sqrt{1-\rho^2}} \exp\left[-\frac{z_i}{2(1-\rho)}\right] + \sum_{J_f \in F}\left(FS(f) \sum_{(t_j, x_j)\in J_f} \left|t_j - t\right|^{-\alpha} \cdot \delta_{x_i x_j}\right)$$

Equation 33 er den tidsuafhængige sammensætning af den spaitale periodiske komponent Equation 20 og den sociale komponent Equation 30. Det er ikke muligt at bestemme det globale maksimum af Equation 33 analytisk, hvorfor dette må bestemmes numerisk. Til formålet anvendes en pyton implementation af L-BFGS-B algoritmen (20. R. H. Byrd. et al.)(21. C. Zhu. et al.).

Sandsynlighedsfordelingen, beskrevet i Equation 33, der leder til bestemmelsen af X_{guess} er en kontinuert funktion af positionen x_i, mens de målbare positioner, i form af accesspoints, er diskrete. Dette betyder at X_{guess} stort set aldrig vil være præcis magen til den diskrete position det sammenlignes med, selv hvis X_{guess} kun er få meter fra en diskret position og der ikke findes andre diskrete positioner i nærheden. For denne overgang mellem kontinuert og diskret rum sammenlignes X_{guess} med alle tidligere besøgte accesspoints og positionen af det tætteste accesspoint anvendes som det endelige positionsgæt X_{final}.

Equation 34

$$X_{final} = \arg\min\left(\left|X_{guess}, X_{access}\right|\right)$$

hvor X_{access} er sættet af alle accesspoints besøgt af brugeren og $\left|X_{guess}, X_{access}\right|$ sybolisere afstanden mellem X_{guess} og X_{access}.

6. Evaluering af model

Under evaluering af PSMM ønsker jeg at belyse en række aspekter af modellens praktiske perfomance. For det første vil jeg undersøge hvilken effekt det har at udvide fra den rent periodiske model PMM, til PSMM inkluderende den sociale komponent. Herunder vil jeg også se nærmere på om den parallelle eller serielle version PSMM har bedst performance i praksis. Dernæst vil jeg undersøge PSMM's evne til at beskrive og forudsige en DTU-brugers bevægelsesmønster og endelig vil jeg sammenligne PSMM med en baselinemodel kaldet Most Frequent Location.

Den grundlæggende fremgangsmåde i evaluering PSM-modellens performance er for hver bruger at træne modellen på denne brugers 8 ugers træningsæt og derefter gennemløbe alle *'on campus'*-check-ins i denne brugers evalueringssæt. For hver af disse check-ins sammelingnes den faktiske check-in-positionen $X_{faktisk}$ med modellens gæt X_{final} til check-in tiden og herfra beregnes den spatiale afstand mellem $X_{faktisk}$ og X_{final} herefter kaldet afstandsfejlen. Derudover beregnes antallet af eksakte match mellem $X_{faktisk}$ og X_{final}. Modellens evne til at forudsige om brugeren befinder på eller udenfor campus evalueres ved at beregne antallet af match mellem evalueringssættet og PSMM's *'on/off campus'*-gæt. Her betragtes alle check-ins i evalueringssættet der er forskellige fra *off campus,* som *on campus*. PSMM gættet bestemmes her udelukkende af den temporale komponent, og en bruger betragtes som værende *off campus* når *off campus* udgør 50% eller mere af den temporale sandsynlighedsmasse (se eksempel på **Error! Reference source not found.**). Under almindelig brug vil PSMM før et on campus positiongæt undersøge om der er størst temporal sandsynlighed for at brugeren befinder on eller off campus, og kun i det første tilfælde fortsætte med en egentlig geografisk positionsbestemmelse. I denne evaluering vil *on/off campus* og det egentlige postionsgæt blive evalueret hver for sig. Således vil PSMM blive tvunget til at komme med et on-campus positionsgæt i svarende til hver check-in i evalueringssættet uanset om PSMM vurderer at der er størst sandsynlighed for at brugeren ikke befinder sig på campus.

6.1.1. Praktiske udfordringer ved den implementerede version af PSMM

Den implementerede version af PSMM indeholder to elementer der kræver relativ meget beregningskraft. Det første er EM-optimering, hvor hver af de ofte flere tusinde check-ins i træningssættet skal fordeles til de lokationer de med størst sandsynlighed tilhører. Dette sker gentagne gange indtil metoden konvergere. EM-optimeringen gentages herefter et antal gange fra forskellige udgangspunkter for at opnå den bedst mulige modellering af en bruger. Hver kørsel af optimerings metoden tager ca. 1 minut med den tilgængelige regnekraft. Dernæst skal modellen evalueres mod ofte et par hundrede check-ins i evaluerings sættet. For hver af disse check-ins skal modellens positionsgæt

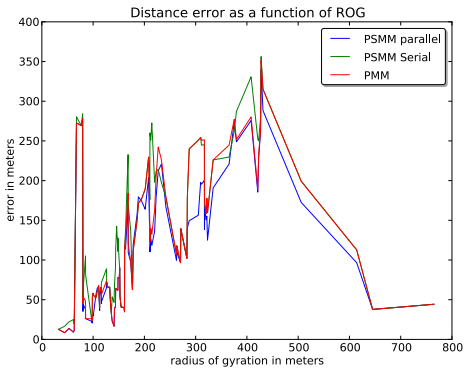
bestemmes ved hjælp af en numerisk optimerings algoritme som omtalt i afsnittet Positionsestimering fra PSMM-sandsynlighed hvilket tager et par sekunder. Samlet set betyder en evaluerings tid pr. bruger på ca. 10 min, hvis EM skal have lov at starte fra bare 5 forskellige udgangspunkter. På grund af modellens evalueringsomkostninger har jeg derfor valgt kun at evaluere modellen på et subsæt bestående af 100 DTU-brugere tilfældigt udvalgt, men med den begrænsning af de skulle have mindst 2000 registrerede check-ins totalt i trænings- og evalueringssættet.

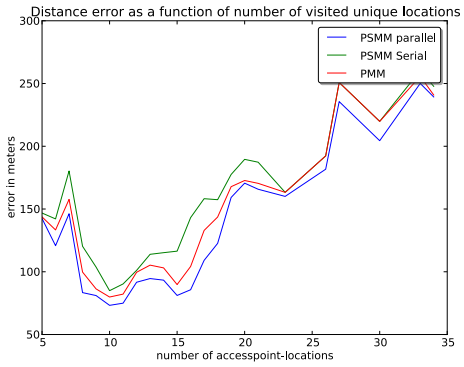
6.1.2. Evaluerings plot

For alle plots i evalueringsafsnitter gælder det at datagrundlaget er de 100 netop udvalgte DTU-brugere. I plots med et lavt antal diskrete x-værdier som eksempelvis Figure 23 (nederst) vil der mange gange være et antal forskellige funktions værdier for hver x-værdi. Derfor anvendes gennemsnittet af alle funktionsværdier tilhørende en given x-værdi i disse tilfælde

6.2. PSMM vs. PMM og parallel vs. Seriel

Af Figure 23 fremgår det af alle tre plots at PMM og den parallelle og serielle version af PSMM alle følges meget tæt, der er altså ikke tale om meget markante forskelle i performance fra det tre model versioner. Når det er sagt er der dog en ganske klar tendens til at PSMM parallel næsten uden undtagelser har en mindre afstandsfejl end PMM, og at PMM igen med ganske få undtagelser performer bedre end PSMM seriel. Heraf kan vi konkludere to ting for det første at den parallelle version af PSMM i praksis er et bedre valg end den serielle version og dernæst at tilføjelsen af den sociale komponent (i parallel med den periodiske) i praksis giver næsten 9% bedre performance målt på afstandsfejl end den rent periodiske. Med dette slået fast vil resten af evaluerings afsnittet omhandle den parallelle version af PSMM, der herefter blot vil blive omtalt som PSMM.





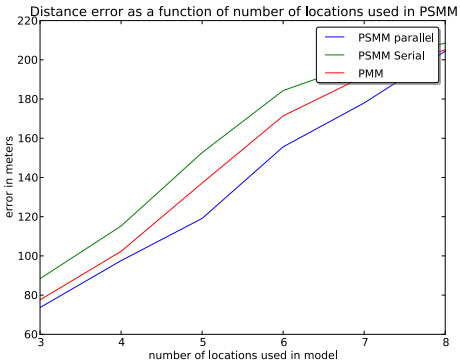


Figure 23: I alle tre plots ses den parallelle PSMM version (blå), den serielle PSMM version (grøn) og den rent periodiske model PMM (rød). Øverst til venstre ses afstandsfejlen som funktion af radius of gyration. Øverst til højre afstandsfejlen som funktion af antallet af unikke lokationer en bruger har besøgt. Nederst ses afstandsfejlen som funktion af antallet modellerede lokationer i PSMM.

6.3. PSMM performance

Lad os først se nærmere på hvor tæt et positionsgæt fra PSMM kommer på en brugers faktiske position ved at kigge på afstandsfejlen. I gennemsnit ligger denne på ca. 111 meter, hvilket alt efter en brugers gennemsnitlige rejseafstand er et mere eller mindre tilfredsstillende resultat. Jeg normalisere derfor afstandsfejlen med den enkelte brugers radius of gyration, der jo netop er den gennemsnitlige rejseafstand, og får næsten 0.61. Vi har altså at PSMM i gennemsnit kan bestemme en brugers position inden for lidt over halvdelen af den almindelige rejseafstand. Dettet tyder på at modellen har svært ved at skelne to på hinanden følgende check-in lokationer fra hinanden, men i og med modellen altid rettes ind til en eksakt accesspoint lokation indikere denne gennemsnitlige afstands fejl nok snarere at ca. halvdelen af disse positionsgæt er eksakte mens den anden halvdel er forkerte med en afstandsfejl svarende til hele brugerens radius of gyration. Denne tolkning bekræftes af at PSMM er i stand til at bestemme en brugers eksakte position i ca. 60% af alle forsøg. På Figure 24 ses hvordan sandsynligheder for en eksakt positions bestemmelse (herefter benævnt P[eksakt]) fordeler sig for de enkelte brugere. Det fremgår at der er stor forskel fra bruger til bruger, hvorfor jeg nu vil se nærmere på årsagen til denne forskel. Af Figure 25 fremgår det at PSMMs evne til at forudsige en brugers eksakte position er omvendt proportional med såvel brugerens radius of gyration som entropien i dennes bevægelsesmønster. Således kan det konstateres at PSMM, som forventet, performer bedst på brugere der bevæger sig inden for et relativt lille geografisk område og hvis bevægelsesmønster er meget periodisk.

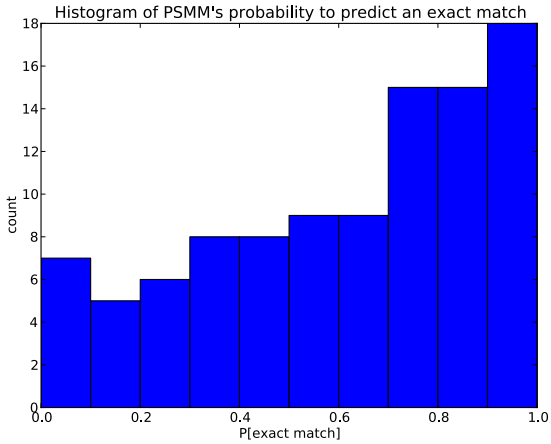


Figure 24: Histogram over fordelingen over PSMMs sandsynlighed for korrekt at forudsige en brugers position

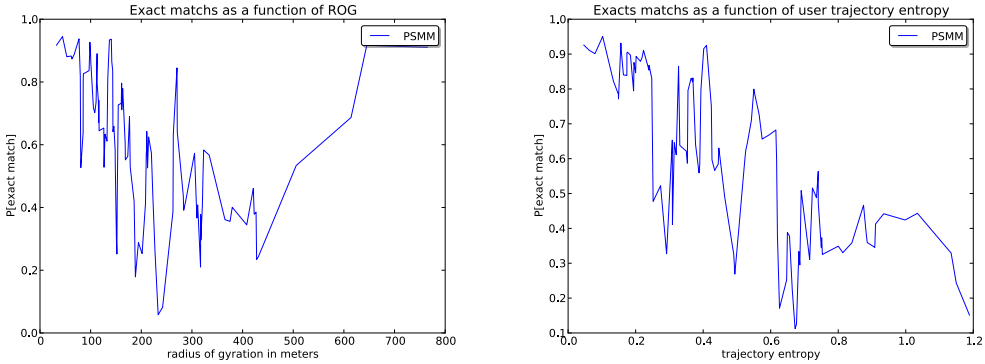


Figure 25: Til venstre ses et plot over sammenhængen mellem PSMMs evne til at forudsige en brugers eksakte position som funktion dennes radius of gyration. Til højre ses igen PSMMs evne til at forudsige en brugers korrekte position men her som funktion af entropien i brugerens bevægelsesmønster (entropi beregnet efter forskriften i Equation 15)

I afsnittet Øvre grænser for forudsigelighed af menneskelig bevægelse beskrev jeg en metode til at udregne den øvre grænse for en brugers forudsigelighed baseret på et givent datagrundlag. Jeg vil her undersøge hvordan PSMM performer i forhold til dette optimum, P_{max}. Gennemsnitligt er PSMM i stand til at udnytte informationen i det tilgængelige datagrundlag til at opnå en eksakt prædiktionsevne på 63% af hvad der er teoretisk muligt. Det skal nævnes at det selvfølgelig er muligt i den relativt korte evaluerings periode at en bruger bevæger sig på en sådan måde at PSMM er i stand til at outperforme det teoretiske maksimum, men givet at brugeren opretholder den samme entropi i sit bevægelsesmønster vil PSMM på sigt konvergere mod en forudsigelsespræcision der ligger under eller lig det teoretiske maksimum. Af Figure 26 fremgår det, ikke overraskende, at *P[eksakt]* er proportional med P_{max}. Hvad der er mere bemærkelsesværdigt er proportionalitets forholdet (lig mærke til x- og y-aksens forskellige skalaer) mellem de to. *P[eksakt]* aftager således flere gange hurtigere end P_{max}, hvilket tyder på at PSMM håndtere høj bruger entropi relativt dårligere end lav bruger entropi i forhold til det teoretiske optimum. Eller med andre ord at PSMM er dårligere til at udnytte den tilgængelige information i trajectories med høj entropi end i trajectories med lavere entropi.

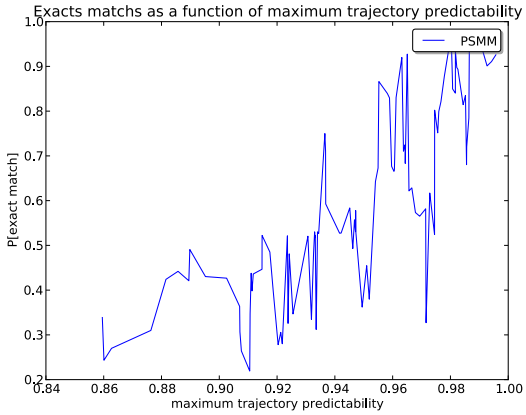


Figure 26: På plottet ses sandsynligheden for at PSMM kan forudsige en brugers position eksakt, P[eksakt], som funktion af den teoretiske maksimumforudsigelighed af denne bruger.

Endelig skal vi undersøge forholdende omkring den specielle lokation *off campus*. Som tidligere nævnt styres PSMMs *on/off campus*-gæt udelukkende af den temporale komponent da *off campus* ikke modelleres som en lokation i rummet, men udelukkende i tid. Og et *off campus* gæt er således defineret som alle de tilfælde hvor *off campus* lokationen udgør mere end 50% af den samlede temporale sandsylighedsmasse. PSMM er i gennemsnit i stand til at gætte om en bruger befinder on eller off campus i 82% af alle forsøg, denne sandsynlighed benævnes P[eksakt$_{on/off}$]. Af Figure 27 fremgår det at P[eksakt$_{on/off}$] tilsyneladende ikke afhænger af entropien i en brugers bevægelsesmønster (Entropien i bruger *u's* bevægelsesmønster benævnes Ent_u). Dette kan forklares ved at entropien i en brugers bevægelsesmønster on campus ikke har nogen indflydelse når alle on campus check-ins betragtes som samme lokation og at alle brugere tilsyneladende har et relativ regelmæssigt *on/off-campus*-bevægelsesmønster. En mere interessant observation er at P[eksakt$_{on/off}$] lader til at være afhængig af antallet af lokationer en brugers bevægelsesmønster modelleres over. Jo flere lokationer des lavere P[eksakt$_{on/off}$]. Da vi lige har konstateret at P[eksakt$_{on/off}$] er uafhængig af Ent_u kan den omvendte proportionalitet ikke tilskrives en tendens til at brugere med højere entropi modelleres ved hjælp at et højere antal lokationer. Ikke desto mindre er den mest sandsynlige forklaring at denne omvendte proportionalitet at brugere modelleret med færre lokationer er bedre temporalt beskrevet af PSMM og at antallet anvendte modellerings lokationer ikke er afhængigt af Ent_u.

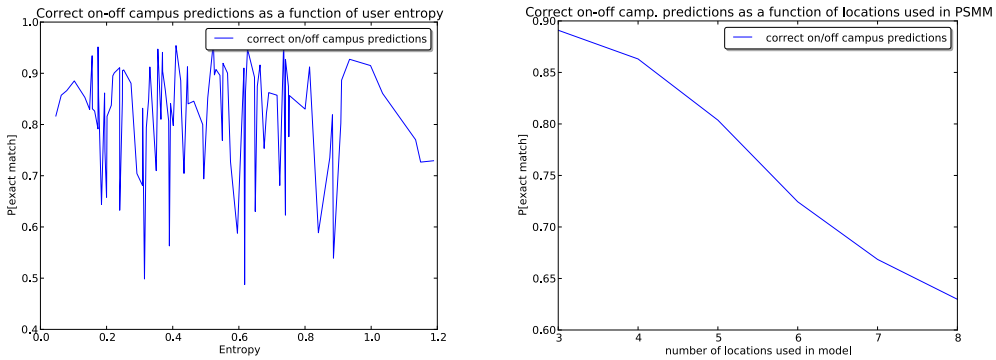


Figure 27: På plottet til venstre ses sandsynligheden for korrekt at gætte om en bruger befinder sig on eller off campus, som funktion af antallet af modellerede lokationer i PSMM. Til højre ses igen

sandsynligheden for at PSMM korrekt kan gætte om en bruger befinder sig on eller off campus, men denne gang som funktion af brugerens trajectory-entropi (Equation 15)

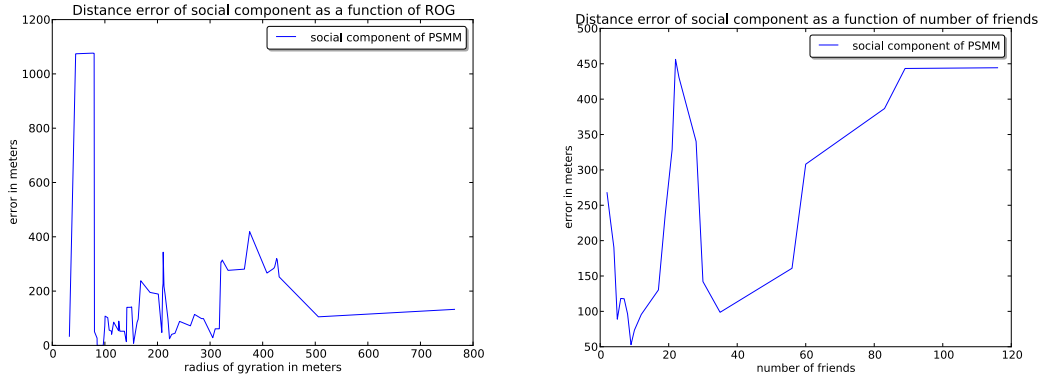


Figure 28: Venstre: afstandsfejlen for den sociale komponent af PSMM som funktion af radius of gyration. Højre: Aftandsfejlen af den sociale komponent af PSMM som funktion af antallet af brugerens sociale relationer.

Figure 28 viser interessant nok, at den sociale komponent af PSMMs afstandsfejl er stort set uafhængig af en brugers radius of gyration, hvilket står i skærende kontrast til afstandsfejlen af den periodiske komponent, der tidligere er blevet konstateret proportional med en brugers radius of gyration. En faktor der derimod påvirker performance af den sociale komponent markant er antallet af sociale relationer. Af Figure 28 (venstre) fremgår det at afstandsfejlen for den sociale komponent er tydeligt mindst for brugere med mellem 3 og 15 sociale relationer. Det andet lokale minimum der observeres ved 36 venner anser jeg et par enkeltstående tilfælde af ekstraordinær præcision. Generelt er der kun 9 ud af de 100 udvalgte brugere der har mere end 22 venner hvorfor datagrundlaget for at drage konklusioner er ganske usikkert for høje antal af venner. Det tyder dog på at for få sociale relationer betyder at datagrundlaget for den sociale komponent bliver for lille, mens for mange sociale relationer giver for meget støj i data til at der kan laves præcise positionsforudsigelser.

6.3.1. PSMM performance konklusion

PSMM kan i gennemsnit korrekt bestemme om en bruger befinder sig on eller off campus i 82% af alle tilfælde og yderligere bestemme en brugers position on campus med en præcision på ca. halvdelen af denne brugers radius of gyration og eksakt i ca. 60% af alle tilfælde. Jeg anser denne performance for at være ganske tilfredsstillende og præcis nok til give en særdeles brugbar forudsigelse af hvor en bruger vil befinde sig på en givent tidspunkt i en kommende uge. Det blev samtidig konstateret at der er en relativ stor spredning i modellens performance fra bruger til bruger og at det for 50% af de udvalgte brugere er muligt at bestemme deres eksakte position med mere end 70% sandsynlighed. I forhold til performance af den sociale komponent blev det konstateret at denne opnår den bedste performance når en bruger har mellem 3 og 15 venner. Endelig blev det observeret at PSMM, i forhold til det teoretiske prædiktions maximum performede, er relativt flere gange dårligere for brugere med høj trajectory entropi end for brugere med lav entropi. Over hele den evaluerede population

svarede PSMMs evne til at forudsige en brugers eksakte position til 63% af teoretiske maksimum.

6.4. PSMM vs. Baseline (MFL)

For at undersøge hvor stor en del PSMMs success i forhold til at forudsige DTU brugeres position der kan tilskrives modellen og hvor meget der kan tilskrives regelmæssighed i datasættet sammenlignes PSMM nu med en baseline model kaldet Most Frequent Location.

6.4.1. Most Frequent Location (MFL)

Most Frequent location er en metode der bestemmer sandsynligheden for at en bruger foretager et check-in på en lokation x_i i løbet af en given time på ugen h_c, udfra brugerens andel af check-ins der historiske er foretaget i denne time h_c, og på denne lokation x_i. Lad nu C_u, være sættet alle check-ins foretaget af bruger u, så er MFL givet ved,

Equation 35

$$P_{MFL}[x_u(t) = x \mid t \in h] = \frac{\left|\left\{c \mid c \in C_u, x_c = x, h_c = h\right\}\right|}{\left|\left\{c \mid c \in C_u, h_c = h\right\}\right|}$$

På trods af modellens simplicitet er modellen er meget stærkt sammengligningsgrundlag. Idet den for hver time på dagen udregner brugerens mest sandsynlige lokation (oftest besøgte lokation i den givne time). Modellen er også særdeles intuitiv idet den implementerer følgende ræsonnement: Forestil dig at du bliver bedt om at gætte en vens position kl. 14.00 onsdag eftermiddag. Hvis du ved hvor din ven arbejder, vil du med høj sikkerhed kunne fastslå at han befinder sig netop der. Ligesådan ville det hvis du kendte din vens private adresse være nemt at gætte hvor denne opholder sig kl. 04.00 onsdag nat. Faktisk forholder det sig sådan at hvis en bruger *u's* bevægelsesmønster er perfekt periodisk, så for $|C_u| \rightarrow \infty$, vil P_{MFL} konvergere mod den sande underliggende bevægelsesmodel. Hvad MFL mangler i forhold til PSMM, når det kommer til brugere der ikke følger perfekte periodiske bevægelsesmønstre er at tage højde for afstande mellem lokationer og deres indbyrdes placering i rummet.

6.4.2. PSMM vs. Baseline

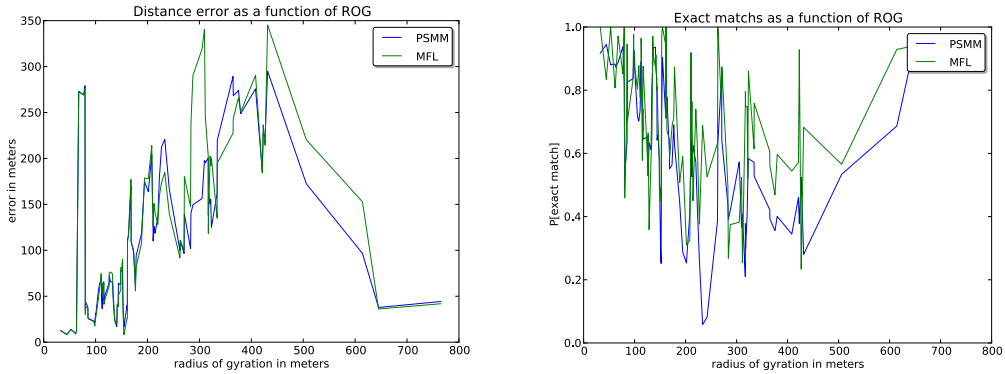


Figure 29: På plottet til venstre ser den gennemsnitlige afstandsfejl mellem positions gæt fra henholdsvis PSMM og MFL til brugerens faktiske position, som funktion af denne brugers radius of gyration. Til højre ses sandsynligheden for et eksakt match mellem positionsgæt foretaget af henholdsvis PSMM og MFL og brugerens faktiske position som funktion af radius of gyration.

Af Figure 29 fremgår det som man kunne forvente at den gennemsnitlige afstandsfejl mellem et positions gæt fra både PSMM og MFL og brugerens faktiske position generelt vokser i takt med at brugerens radius of gyration. Ligeledes kan vi konstatere dog med noget mere støj at sandsynligheden for en eksakt positionsforudsigelse aftager når en brugers radius of gyration vokser. Det kan desuden konstateres at mens PSMM performer marginalt bedre end MFL når performance vurderes på baggrund af afstandsfejl, ses et omvendt og betydelig mere markant forhold mellem de to modellers performance når det kommer til sandsynligheden for foretage eksakte positionsgæt for brugere med høj radius of gyration. Det ser ud som om at MFL begynder at outperforme PSMM for brugere med radius of gyration over 200 meter. Denne forskel kan forklares ved de to modellers forskellige opbygning. MFL kan således kun afgive positionsgæt på et endeligt sæt diskrete positioner svarende til de positioner brugeren har besøgt i træningsperioden. PSMM kan derimod afgive positionsgæt i hele rummet udspændt af brugerens besøgte positioner og sågar positioner brugeren ikke har besøgt hvis en ven har befundet sig her kort før afgivelsen af et positionsgæt. Forskellen i de to modellers performance kan således forklares ved at forestille sig et eksempel hvor brugeren har besøgt tre forskellige lokationer. De tre lokationer er placeret på en ret linje og henholdsvis 40% og 50% af en brugers check-ins er sket på hver af de to lokationer der udgør linjens endepunkter og de sidste 10% check-ins er sket på midt-lokationen. Hvis vi yderligere forestiller os at de 90% check-ins på de to ende-lokationer er sket en ad gangen skiftevis til den ene og den anden lokation så har vi en situation hvor PSMM i et worstcase scenarie i 90% af tilfældene foretage positionsgæt mellem de to ende-lokationer og blive rettet ind til den nærmeste lokation der i dette tilfælde vil være midt-lokationen. PSMM vil i dette tilfælde afgive 0 eksakte lokationsgæt i 90% af tiden og have en gennemsnitlig afstandsfejl på halvdelen af afstanden mellem de to endepunkter. MFL derimod vil gætte på den ene af de tre lokationer der er mest sandsynlig på et givent tidspunkt og vil gætte korekt i 50% af tilfældene men stadig have en afstandsfejl på næsten halvdelen af afstanden mellem de to endepunkter. I eksemplet antages at trænningssæt og evalueringssæt er identiske og at MFL og PSMM ikke betragter en brugers mest

sandsynlige position som værende fordelt 50%, 40% og 10% mellem de tre lokationer. Dette forsimpler de to modeller en smule, men tjener til at forklare hvordan forskellen i de to modellers evne til at foretage eksakte positionsgæt opstå. I forhold til at dette primært observeres ved radius of gyration over 200 meter så kan det forklares ved at det ved gennemsnitlige rejseafstande på over 200 meter bliver langt mere sandsynligt at der findes et af disse 'midt-lokations' et sted undervejs. Denne 200 meter grænse er med al sandsynlighed ikke universal, men et udtryk for de specifikke rummelige egenskaber af DTU datasættet.

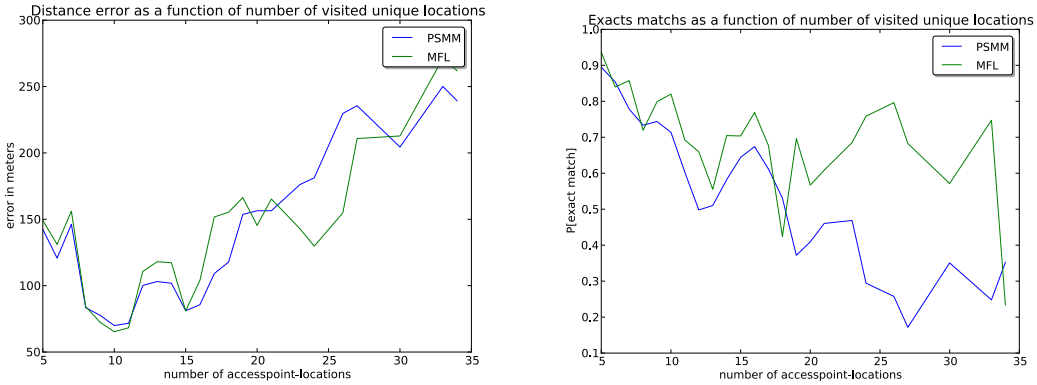


Figure 30: til venstre ses den gennemsnitlige afstandsfejl mellem positions gæt fra henholdsvis PSMM og MFL til brugerens faktiske position, som funktion af antallet af accesspointlokationer besøgt. Til højre ses sandsynligheden for et eksakt match mellem positionsgæt foretaget af henholdsvis PSMM og MFL og brugerens faktiske position ligeledes som funktion af det besøgte antal af accesspointlokationer.

Vi kan konstatere af Figure 30 at PSMM og MFL evne til præcist at forudsige en brugers position falder i takt med antallet af lokationer brugeren har besøgt vokser. PSMM og MFL reagere tilsyneladende relativt ens på et voksende antal at besøgte lokationer når de vurderes på afstandsfejl. MFL stærkere relative performance når det kommer til eksakt at bestemme bruger positioner bliver bekræftet, idet det kan konstateres at der for brugere med mere end ca. 17 besøgte accesspoint er en signifikant forskel mellem PSMM og MFL når det kommer til sandsynligheden for at foretage et eksakt positionsgæt. Ligesådan kan det konstateres at forskellen vokser i takt med antallet af accesspointlokationer brugeren har besøgt. Denne opførsel kan igen forklares ved at sandsynligheden for 'midt-lokationer' vokser i takt med antallet af besøgte lokationer.

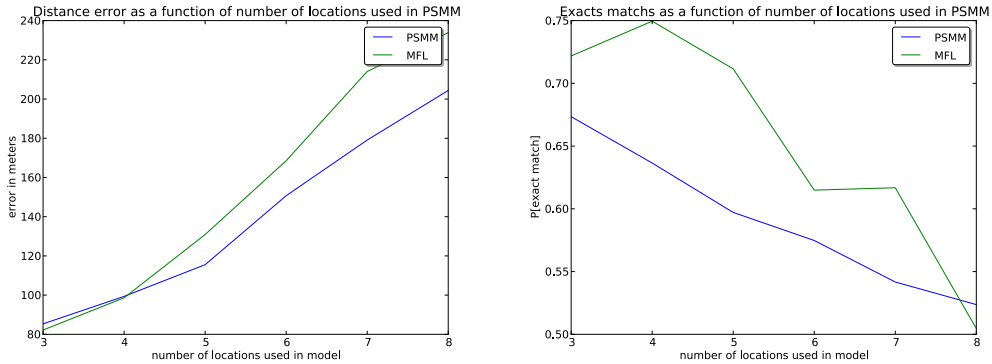


Figure 31: til venstre ses den gennemsnitlige afstandsfejl mellem positions gæt fra henholdsvis PSMM og MFL til brugerens faktiske position, som funktion af antallet af modellerede lokationer i PSMM. Til højre ses sandsynligheden for et eksakt match mellem positionsgæt foretaget af henholdsvis PSMM og MFL og brugerens faktiske position ligeledes som funktion af antallet af modelerede lokationer.

Af Figure 31 (venstre) kan vi se at aftandsfejlen vokser i takt med antallet af modellerede lokationer i PSMM. At dette ikke er et udtryk for at medtagelsen af flere lokationer i PSMM giver en dårligere model, men snarere et udtryk for at der anvendes flere lokationer til at modellere brugere hvis position er sværere at forudsige, kan ses ved at MFL følger samme tendens. Der observeres dog en tendens til at PSMM relativt til MFL, bliver mere præcis både hvad angår afstandsfejl og sandsynlighed for eksakt positionsbestemmelse i takt med at brugerne modelleres over flere lokationer. Dette er en interessant observation der tyder på at PSMM er bedre egnet til at beskrive mere komplicerede bevægelsesmønstre.

6.4.3. PSMM vs. Baseline konklusion

Generelt kan det konstateres at PSMM og MFL har meget ens performance når det kommer til afstandsfejl på tværs af en brugers radius of gyration, antallet af besøgte lokationer og antallet af modellerede lokationer, dog med ca. 6% bedre performance til PSMM over alle de udvalgte brugere. Når det kommer til sandsynligheden for eksakt at bestemme en brugers position følges PSMM og MFL igen ad for brugere med en radius of gyration under 200 meter. For brugere med radius of gyration over 200 meter eller mere end 17 besøgte accesspoints performer MFL dog markant bedre en PSMM. Over hele den evaluere population har MFL ca. 13% højere sandsynlighed for at forudsige brugerens eksakte position end PSMM.

7. Evaluering af resultater

Herunder følger en opsummering og diskussion af de opnåede resultater i forhold til modellering og prædiktion af DTU-brugeres bevægelsesmønstre ved brug af PSMM.

7.1. PSMM opnåede resultater

Under evalueringen af PSMM er det blevet konstateret at den parallelle version af PSMM i praksis er et bedre valg end den serielle version. Derudover blev det slået fast at tilføjelsen af den sociale komponent (i parallel med den periodiske) i praksis giver lidt over 9% bedre performance målt på afstandsfejl end den rent periodiske.

PSMM kan i gennemsnit korrekt bestemme om en bruger befinder sig on eller off campus i 82% af alle tilfælde og yderligere bestemme en brugers position on campus med en præcision på ca. halvdelen af denne brugers radius of gyration og eksakt i ca. 61% af alle tilfælde. Det blev samtidig konstateret at der er en relativ stor spredning i modellens performance fra PSMM vs. Baseline at det for over 50% af de udvalgte brugere er muligt at bestemme deres eksakte position med mere en 70% sandsynlighed. I forhold til performance af den sociale komponent blev det konstateret at denne opnår den bedste performance når en bruger har mellem 3 og 15 venner. Over hele den evaluerede population svarede PSMMs evne til at forudsige en brugers eksakte position til 64% af teoretiske maksimum.

Generelt kan det konstateres at PSMM og MFL har meget ens performance når det kommer til afstandsfejl på tværs af en brugers radius of gyration, antallet af besøgte lokationer og antallet af modellerede lokationer, dog med ca. 6% bedre performance til PSMM over alle de udvalgte brugere. Når det kommer til sandsynligheden for eksakt at bestemme en brugers position følges PSMM og MFL igen ad for brugere med en radius of gyration under 200 meter. For brugere med radius of gyration over 200 meter eller med mere end 17 besøgte accesspoints performer MFL dog markant bedre en PSMM. Over hele den evaluere population har MFL 13% højere sandsynlighed for at forudsige brugerens eksakte position end PSMM.

7.2. Diskussion af resultater

Det er lykkedes at skabe en model der kan forudsige en brugers eksakte position med ca. 61% sandsynlighed hvilket kan sammenlignes med de 40% præcision (7. Eunjoon Cho. et al.) at opnåede i det oprindelige PSMM forsøg. Det er derudover lykkedes at skabe en social model, baseret på et implicit genereret socialt

netværk, der er i stand til at forbedre anstandsfejlen med lidt over 9% fra den rent periodisk model PMM til PSMM. På trods af at 9% er en relativ beskeden størrelse, så er denne præstation signifikant i og med forbedringen er opnået ud fra et uændret datagrundlag, blot ved at analysere den interne struktur i den spatiotemproale data.

Med over 82% sandsynlighed for korrekt at forudsige om en bruger befinder sig på campus og herefter være i stand til at bestemme brugerens position på campus inden for ca. en halv radius of gyration er PSMMs performance tilstrækkelig til at bidrage med brugbar information indenfor en lang række anvendelsesområder. Ikke bare som en metode til at forudsige brugeres position, men også til at analysere og beskrive bevægelsesmønstre i kraft af de relativt præcise modelleringer af individuelle brugere.

7.2.1. PSMM vs. MFL

Når PSMM sammenlignes med MFL bliver det klart at en del af PSMMs gode performance kan tilskrives et 'well-behaved' datasæt, ikke desto mindre er performance af PSMM en smule bedre end MFL målt på afstandsfejl. Målt på sandsynlighed for at forudsige en brugers eksakte position må PSMM imidlertid se sig slået af MFL. Den gode performance af MFL i forhold til PSMM kan forklares ved den spatiale struktur i datasættet. MFL har således, i forhold til PSMM, en fordel i at DTUs brugere i dette datasæt kun kan befinde sig på relativt lavt antal diskrete positioner i rummet (DTUs 430 accesspoints er fordelt på kun 82 geografiske lokationer). Da dette betyder at PSMM kun i ringe grad kan udnytte at modellen tager højde for lokationernes indbyrdes placering i rummet.

Et eksempel hvor dette kan være en fordel er hvis man forestiller sig tre lokationer to med en indbyrdes afstand s og begge disse med afstanden 10 s til den tredje lokation. Hvis vi nu forestiller os en bruger der til tiden t befinder sig på hver af de to nærliggende accesspoints med 30% sandsynlighed og på det sidste accesspoint med 40% sandsynlighed. I denne situation vil MFL altid gætte på at brugeren befinder sig på det accesspoint med højest sandsynlighed (40%), mens PSMM har mulighed for at forme en model hvor de to nærliggende accesspoints betragtes som samme lokation der med 60% sandsynlighed indeholder brugeren. Denne evne vil med stor sandsynlighed være fordelagtig i datasæt hvor brugerens rummelige position har højere granularitet end det er tilfældet i DTU-datasættet. Mens clustering af et lille antal rummelige positioner med stort set samme indbyrdes afstand kan vise sig at være en ulempe, som det er tilfældet MFL er i stand til at forudsige en brugers eksakte position med højere sandsynlighed en PSMM.

7.2.2. Parallel vs. seriel

Jeg valgte at evaluere to forskellige implementeringer af PSMM, hvor den sociale komponent blev inkluderet henholdsvis i parallel og serie med den periodiske. Dette gjorde jeg fordi jeg ikke teoretisk var i stand til at bestemme hvilken af de to der var mest egnet. Det viste sig at den parallelle implementering af PSMM

meget konsistent performede bedre end den serielle. Forklaringen på denne forskel må, eftersom den sociale og periodiske komponent er den samme for de to løsninger, skulle findes i den sociale komponents indflydelse på det samlede resultat der er styret af henholdsvis af vægtning og outlier-vurdering. Mit bud er at denne forskel bunder i at den periodiske og sociale rummelige sandsynlighedsfordeling kan have maksimum forskellige steder og dermed have et sammensat maksimum der ligger et sted mellem de to komponenters maksima. Mens brugeren med stor sandsynlighed tidligere har og igen vil befinde sig på hver af komponenternes separate maksimum , kan det sammensatte maksimum meget vel være en position hvor brugeren aldrig befinder sig. Denne situation opstår kun for den serielle implementering da den parallelle kun benytter en af de to komponenter i en given positionsbestemmelse. Derudover er det muligt at der med de konkrete implementeringer af vægtning og outlier-vurdering er større sandsynlighed for at den sociale komponent går ind og 'forstyrre' den periodiske i tilfælde hvor det periodiske gæt er korrekt.

7.2.3. Udgangspunkt

Træning af PSM-modellen er meget afhængig af det tilfældigt generede lokationer der danner udgangspunkt for EM-metoden, hvilket betyder at resultaterne kan variere fra gang til gang. Dette kan delvis undgås ved at lade EM-metoden køre fra et højt antal forskellige udgangspunkter, men da EM-skridtet er særdeles tidskrævende, er der alvorlige praktiske begrænsninger for hvor højt dette antal kan blive. Med den nuværende implementation af PSMM var det ikke praktisk muligt at lade EM-metoden starte fra mere end 10 forskellige udgangspunkter, hvilket er lavt nok til at jeg forventer en signifikant forbedring af PSMMs performance hvis dette blev sat op til eksempelvis 100 eller flere udgangspunkter.

7.2.4. PSMM i forhold til teoretisk maksimum

PSMM var i gennemsnit i stand til at opnå en prædiktionsevne på 64% af det teoretiske maksimum, hvilket antyder at PSMM kan optimeres yderligere i forhold til DTU-datasættet, og/eller at der kan udvikles alternative prædiktions algoritmer der er bedre egnet til formålet.

8. Konklusion

Igennem projektet er menneskelige bevægelsesmønstre blevet studeret. I de indledende studier blev det konstateret at klassiske partikel forskydningsmodeller som Lèvy flight og random walk, på grund at periodiciteten i menneskelig bevægelse, ikke var egnede til modellering heraf. Studierne blev fokuseret indenfor 3 centrale definerende faktorer for menneskelig bevægelse, tid, sted og sociale relationer. Da kun den spatiale og temporale information er tilgængelig i det anvendte datasæt blev det sociale netværk konstrueret implicit. Herefter blev modellen PSMM, periodisk og social mobilitets model udvalgt som grundlag for modellering af DTU-populationen i kraft af dens inklusion af de tre faktorer tid, sted og sociale relationer. PSMM blev herefter at tilpasset og optimeret til modellering af DTU-populationen.

Det er således lykkedes at skabe en model der kan forudsige en brugers eksakte position med ca. 61% sandsynlighed på baggrund af spatiotemporal data. Denne præstation kan sammenlignes med de 40% præcision (7. Eunjoon Cho. et al.) at opnåede i det oprindelige PSMM forsøg. Det er derudover lykkedes at konstruere en social model, baseret på et implicit genereret socialt netværk, der er i stand til at forbedre anstandsfejlen for modellens positionsestimeringer med lidt over 9% i forhold til den rent periodisk model PMM. På trods af at 9% er en relativ beskeden størrelse, så er denne præstation alligevel signifikant i og med forbedringen er opnået ud fra et uændret datagrundlag, blot ved at analysere den interne struktur i den spatiotemproale data.

Med over 82% sandsynlighed for korrekt at forudsige om en bruger befinder sig på campus og herefter være i stand til at bestemme brugerens position på campus inden for ca. halvdelen af brugerens radius of gyration er PSMMs performance tilstrækkelig til at bidrage med brugbar information indenfor en lang række anvendelsesområder. Ikke kun som en metode til at forudsige brugeres position, men også til at analysere og beskrive bevægelsesmønstre i kraft af de relativt præcise modelleringer af individuelle brugere.

Projektets formål er således opfyldt idet det er lykkedes at udvikle en metode til modellering af DTU-brugeres spatiotemporale bevægelsesmønstre, der kan anvendes til, med signifikant præcision, at forudsige disse brugeres position.

9. Forslag til retninger for videre arbejde

Som jeg ser det er der 5 oplagte områder for videre arbejde med modellering af bevægelsesmønstre og PSMM i forlængelse af dette projekt.

1. Jeg ville først og fremmest arbejde med at strømline implementeringen af PSMM metoden (se kildekode i appendix D), således at træning og evalueringstiden bliver bragt ned på et niveau hvor det er muligt at køre langt flere EM gentagelser og modellen kan evalueres på hele det tilgængelige datasæt.
2. Dernæst ville jeg gentage mit tidligere forsøg på at indsamle data fra brugernes synspunkt (se appendix C) ved hjælp af smartphones og derigennem udforske styrker og svagheder ved data indsamlet fra henholdsvis netværkets og brugernes synspunkt. Jeg forventer at de to datasæt vil kunne supplere hinanden og give en helhed der er større end de to dele.
3. DTU-datasættet indeholder en del metadata om hver enkelt bruger så som oversigt over tilmeldte kurser, studieretning, privat adresse mm. Der ville være interessant at sammenholde disse data med de nuværende resultater.
4. Forsøg med den implementerede version af PSMM på andre datasæt, både fra andre universiteter, men også fra mere inhomogene populationerne og observeret over et større geografisk område.
5. Yderligere arbejde med karakterisering af brugere og lokationer ud fra DTU-brugernes bevægelsesmønstre og eventuelt data indsamlet via smartphones.

10.Kildehenvisning

1. Ascher. et al., U. M., & Petzold, L. R. *Computer methods for ordinary differential equations and differential-algebraic equations. 1998.*
2. Chaoming Song. et al. (support), Z. Q.-L. *Supporting Online Material for Limits of Predictability in Human Mobility.*
3. Chaoming Song. et al., Z. Q.-L. *Limits of Predictability in Human Mobility.*
4. Chiara Boldrini. et al., M. C. *The Sociable Traveller: Human Travelling Patterns in Social-Based Mobility.*
5. Dashun Wang. et al., D. P.-L. *Human Mobility, Social Ties, and Link Prediction.*
6. David J. Crandalla. et al., L. B. *Inferring social ties from geographic coincidences.*
7. Eunjoon Cho. et al., S. A. *Friendship and Mobility: User Movement In Location-Based Social Networks.*
8. Grubbs, F. E. *1969, Procedures for detecting outlying observations in samples. Technometrics 11, 1–21.*
9. Justin Cranshaw. et al., E. T. *Bridging the Gap Between Physical Location and Online Social Networks.*
10. Juyong Park. et al., D.-S. L. *The eigenmode analysis of human motion.*
11. Marta C. Gonza´lez. et al., C.´.-L. *Understanding individual human mobility patterns.*
12. N. Navet. et al., S.-H. C. *Natural Computing in Computational Finance (Springer, Berlin, 2008).*
13. Nathan Eagle. et al., A. (. *Inferring friendship network structure by using mobile phone data.*
14. Fano, M. R. *Transmission of Information (the MIT Press and Wiley, New York and London, 1961).*
15. H. Eugene Stanley. et al., R. N. *Stochastic Process with Ultraslow Convergence to a Gaussian: The Truncated Lévy Flight.*
16. Taylor, J. R. *An Introduction to Error Analysis. 2nd edition. Sausalito, California: University Science Books, 1997. pp 166-8.*
17. Weisstein, E. W. (n.d.). *"Bivariate Normal Distribution." From MathWorld--A Wolfram Web Resource.* From http://mathworld.wolfram.com/BivariateNormalDistribution.html
18. Weisstein, E. W. (n.d.). *"Markov Process." from MathWorld--A Wolfram Web Resource.* From http://mathworld.wolfram.com/MarkovProcess.html
19. Weisstein, E. W. (n.d.). *"Perron-Frobenius Theorem." From MathWorld--A Wolfram Web Resource.* From http://mathworld.wolfram.com/Perron-FrobeniusTheorem.html
20. R. H. Byrd. et al., P. L. *A Limited Memory Algorithm for Bound Constrained Optimization, (1995), SIAM Journal on Scientific and Statistical Computing , 16, 5, pp. 1190-1208.*
21. C. Zhu. et al., R. H. *L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization (1997), ACM Transactions on Mathematical Software, Vol 23, Num. 4, pp. 550 - 560.*

22. Negroni, C. (n.d.). *Tracking Your Wi-Fi Trail*. From http://www.nytimes.com/2011/03/22/business/22airport.html?_r=1
23. (n.d.). *Gowalla.com*. From http://gowalla.com/
24. (n.d.). *BrightKite*. From http://brightkite.com/
25. Milgram, S. (1970). *The experience of living in cities. Science 167:1461–1468.*
26, N. (1949). Guilt by association: Three words in search of a meaning. U Chicago Law Rev 17:148–162.
27, H. K. (2006). *The New Politics of Surveillance and Visibility (University of Toronto Press, Toronto).*
28, F. (n.d.). From http://www.flickr.com/
29. André Panisson. et al., A. B. *On the Dynamics of Human Proximity for Data Diffusion in Ad-Hoc Networks.*
30. D. Brockmann. et al., L. H. (n.d.). The scaling laws of human travel. Nature, 2006.
31. M. C. Gonzalez et al., C. A.-L. (n.d.). Understanding individual human mobility patterns. Nature, 453(7196):779–782, 2008.

11.Appendix A

11.1. Analyse af bevægelsesmønstre i Kastrup lufthavn

I forbindelse med mine studier af brugen af WiFi-teknologi til indsamling af spatiotemporalt data og efterfølgende behandling af denne data, har jeg kontaktet og besøgt PLA-afdelingen (Esben Kolind) i Kastrup Lufthavn. Her arbejder de på nuværende tidspunkt på implementeringen af et system der via WiFi, tracker lufthavnens brugere. Systemet udvikles i samarbejde med SITA et Schweizisk firma specialiseret i udvikling af kommunikations- og IT-løsninger til lufthavne. Tracking systemet er baseret på ca. 500 WiFI-accesspoints fordelt i de offentligt tilgængelige dele af Kastrup Lufthavn. Dette høje antal accesspoint gør det muligt at bestemme en brugers position præcision på 5 til 20 meter. Systemet indsamler spatiotemporalt data fra alle aktive WiFi-enheder inden for rækkevidde med en sampling frekvens på mellem 2 og 5 minutter. Hvis en bruger vælger at registrere sig for at få netværksadgang mod at oprette en profil og betale et trafiktakst, stiger sampling frekvensen til flere gange i minuttet.

For Kastrup Lufthavn er det primære formål med at implementere dette WiFi-trackingsystem at overvåge og analysere passagerflow gennem lufthavnens forskellige områder, ventetid ved check-in skranker og lignende, samt typiske 'paths' gennem lufthavnen. Alt dette med henblik på kommerciel og logistisk optimering af lufthavnens opbygning og dimensionering (22. Negroni). Der er på nuværende tidspunkt ingen planer om at bruge systemet i real time, til f.eks. at løbende tilpasse bemandingen i security-checket efter antallet af passagerer ved check-in skrankerne. Ligesom de i øjeblikket heller gør sig nogen forhåbninger om at forudsige individuelle passagerers bevægelser.

De manglende ambitioner for individuelle bevægelsesprædiktioner skyldes bl.a. at de på nuværende tidspunkt ikke er i stand til at genkende tilbagevendende passagerer, hvilket betyder at en eventuel prædiktionsmodel skulle baseres på en meget lille mængde data, der oven i købet kun vil udgøre en delmængde af et bevægelsesforløb. Forstået på den måde at en passager som regel vil have en konstant fremadrettet bevægelse på sin vej gennem lufthavnen eventuelt med stop og svingærinder men kun sjældent bagudrettet bevægelse og gentagelser.
På sigt er det dog intentionen at tilbagevendende passager skal kunne genkendes om ikke for at kunne lave individuel bevægelsesprædiktion så for mulighederne i at levere personaliserede reklamer baseret på en brugers rutiner. Kastrup Lufthavn har for nyligt lanceret en gratis smartphone applikation der kan bruges som leveringsmedie for både personaliserede reklamer, men også kø-monitorering og andre relevante informationer baseret på trackingsystemets data.

12.Appendix B

12.1. DTU-datasættet

DTU tilbyder gratis trådløst netværk (WiFi) til alle studerende og ansatte via ca. 430 accesspoints fordelt over DTU's campus i Lyngby. Når en bruger ønsker at anvende netværket et det påkrævet at denne logger ind på en personlig konto med brugernavn og password. Derfor kender netværket ikke bare det anvendte device's MAC adresse, men en lang række metadata som brugerens navn og kontotype samt information omkring brugerens tilmeldte kurser mm. Når en bruger er logget på netværket registreres SSID på det involverede accesspoint samt et timestamp for registreringen ca. hvert 10. minut. Et sådan sæt af accesspoint ID og timestamp vil herefter blive omtalt som et check-in. Indsamling af netværks data håndteres af Institut for matematisk modellering (IMM) der løbende arbejder med at analysere den indsamlede data. Gennem IMM har jeg fået lov anvende et beskåret og anonymiseret sæt af disse data indsamlet i perioden Mandag d. 24/01-2011 til mandag 9/5-2011. Dataene er anonymiseret ved at de enkelte brugernavne der normalt består af den studerendes studie ID er blevet erstattet med et hashId og beskæringen betyder at kun brugere med mere end 200 check-ins, i den observerede periode, fordelt på mindst ti forskellige accesspoints er medtaget i sættet. I alt giver det et datasæt med 4367 unikke brugere. Hertil skal det tilføjes at en enkelt studerende kan bruge flere trådløse netværks devices på samme konto og hver af disse vil i så fald have sit eget id og i denne opgave blive betragtet som forskellige brugere.

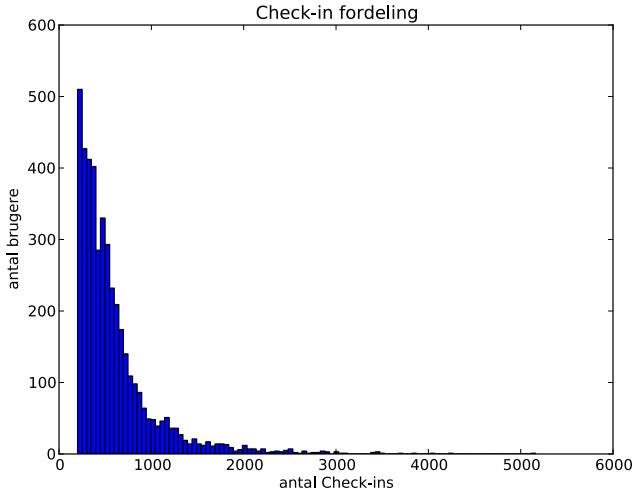


Figure 32: Som det fremgår af Check-in fordelingen har de fleste brugere foretaget mellem 200 og 1000 check-ins i observations perioden fra d. 24/1 til 9/5. Den bratte stigning ved 200 check-ins skyldes at kun brugere med mere end 200 check-ins er blevet medtaget i datasættet.

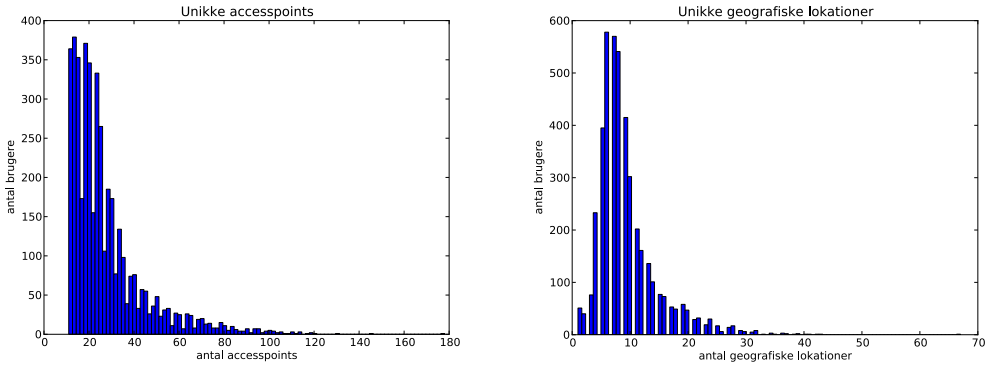


Figure 33: På graffen til venstre ses hvordan kun brugere med check-ins på mere end ti forskellige accesspoints er medtaget i datasættet. Gennemsnitligt har hver bruger checket ind på 28 accesspoints men på fordelingen er assymetrisk omkring denne værdi med største delen af brugerne i lille interval fra 10 til 28, mens de resterende brugere fordeler sig i en 'long tail' der strækker sig til ca. 120 unikke besøgte accesspoint og enkelte særtilfælde med næsten 180 unikke accesspoints. På grafen til højre ses en lignende fordeling men her betragtes alle accesspoints med samme registrerede geografiske postion som et enktelt accesspoint her kaldet lokation. Her ses det at fordelingen af geografiske lokationer med en gennemsnits værdi på lige over 9 minder meget i form om fordelingen over accesspoints blot med 3 gange lavere x-værdier. Vi kan altså konkludere at en bruger i gennemsnit har forbindelse til 3 forskellige accesspoints på hver geografiske lokation. At fordelingen også har brugere registreret på kun 1 eller 2 geografiske lokationer vidner om at nogle lokationer har 10 eller flere unikke accesspoints tilknyttet.

Ud over en CSV-fil for hver bruger indeholdende sammenhørende værdier af accesspoint ID og timestamp, består datasættet af endnu en CSV-fil indeholdende geografiske koordinater for positionen af hver enkelt accesspoint. En kort opsummering af datasættet følger herunder

12.1.1. Datasættet kort

- Sampling frekvens: ca. 1 sampling pr. 10 min
- Antal accesspoints: 430
- Antal brugere: 4367
- Observations periode: Mandag d. 24/01-2011 til mandag 9/5-2011
- Teknologi: WiFi
- Observations format: observations tid i sekunder siden 1. Januar 1970 og position i form af accesspoint SSID
- Data format: Comma Seperated Values (CSV)
- Gennemsnitligt antal check-ins pr. bruger i observationsperioden: 599
- Gennemsnitligt antal besøgte accesspoints pr. bruger i observationsperioden: 28

12.1.2. Rummelig fordeling af accesspoints

På Figure 34 herunder ses den rumelige fordeling af de 430 accesspoints på DTU's campus i Lyngby.

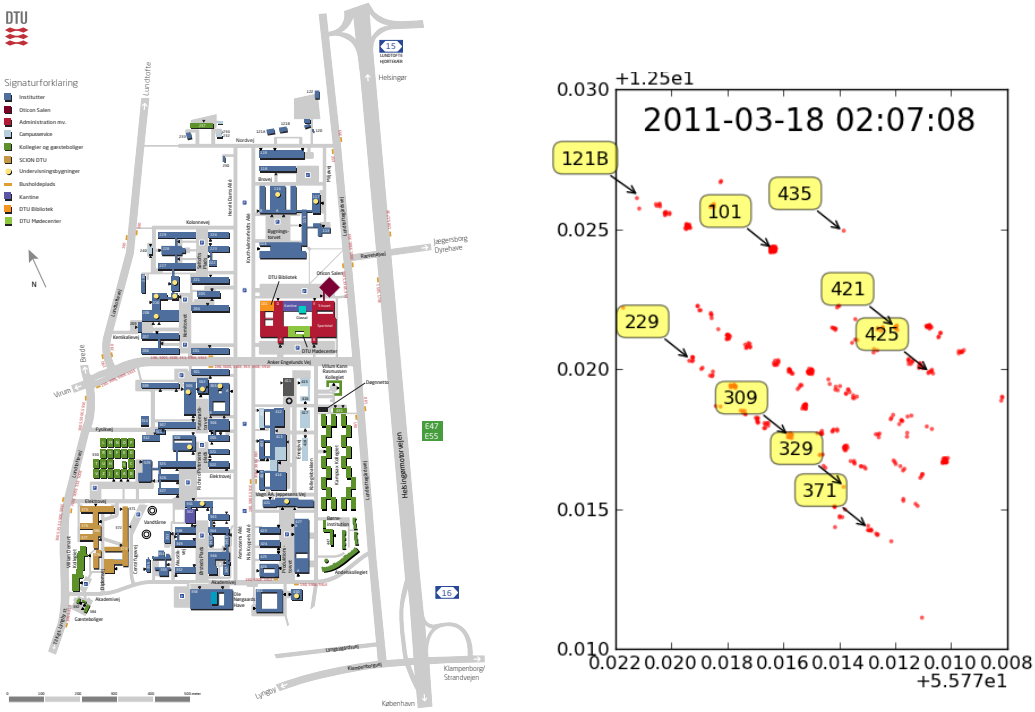


Figure 34: Til højre et kort over DTU og til venstre et plot at den rummelige fordeling af DTU's accesspoints. Bemærk at kortet ikke er orienteret med nord som den lodrette akse, men drejet en smule mod nordvest. På plottet til venstre er tilføjet en svag tilfældig støj for at adskille forskellige accesspoints på samme geografiske koordinater visuelt fra hinanden.

12.1.3. Tidsafhængig bruger aktivitet og udvælgelse og temporal pruning af datasæt

Den implementerede prædiktionsmodel vil i høj grad være baseret på spatioteporale regelmæssigheder i brugernes bevægelsesmønstre. Derfor vælger jeg i første omgang at se bort fra de mest uregelmæssige perioder i datasættet f.eks. omkring eksamen og officielle ferie perioder. Til træning og evaluering af den konstruerede model udvælges således 10 uger ud af de 15 registrerede. De udvalgte uger ligger i intervallet fra mandag d. 7/2 kl. 00.01 til søndag d. 17/4 kl. 23.59, herved undgås de værste uregelmæssigheder forsaget af ferier og semesterstart og -slutning (Figure 35). Af de 10 uger anvendes de 8 første til træning og de to sidste til evaluering af prædiktionsmodellen.

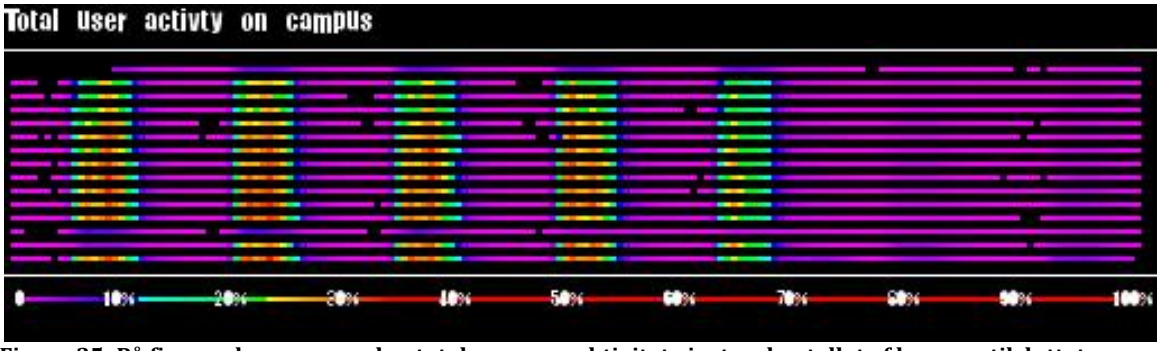


Figure 35: På figuren herover ses den totale campus aktivitet givet ved antallet af brugere tilsluttet til netværket til et givent tidspunkt i løbet af den 15 uger lange dataindsamlings periode. Hver

vandret linie repræsenterer en uge med periodens første uge øverst og sidste nederst. Bruger aktiviteten til et givent tidspunkt er givet ved liniens farve, med sort svarende til 0 tilsluttede brugere og med resten af farveskalaen brugt som anvist nederst i plottet. Procentsatserne svarer til den angivne procentdel af datasættets samlede antal brugere. Af plottet er det tydeligt at se at den første registrerede uge (uge 4) er ugen for DTU's vinterferie og den tredje sidste uge (uge 16) er påskeferien med den karakteristiske påskemandag i den efterfølgende uge. Derud over er det interessant at ligge mærke til hvordan der er signifikant lavere aktivitet på fredage end på ugens fire andre hverdage og hvordan der er højere og længerevarende aktivitet efterhånden som semestret skrider frem.

13.Appendix C

13.1. Indsamling af spatiotemporal- og kontekstdata gennem mobil applikation

Ved projektets start havde jeg en intention om at sammenligne data indsamlet fra brugerens synspunkt med data indsamlet fra netværkets synspunkt. Data indsamlet fra brugerens synspunkt er interessant af flere grunde. For det første er muligt at indsamle data med langt højere spatial- og temporal granularitet. For det andet er det muligt at følge brugeren kontinuerligt også uden for campus. For det tredje (call and sms log), aktivitets niveau (accelerometer) og andre brugere i nærheden (bluetooth scanning). Selv data fra et relativt lille antal brugere ville være interessant at sammenholde med data opsamlet fra netværkets synspunkt.

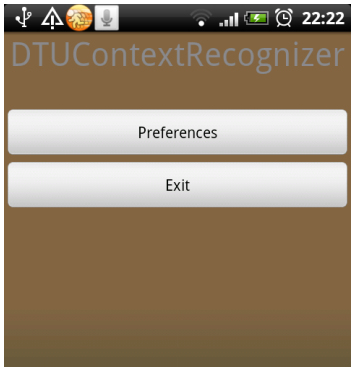
Spatiotemporal data fra netværkets synspunkt var allerede tilgængelig i form af det datasæt jeg har benyttet mig af gennem hele projektet men der fandtes ingen data indsamlet fra brugerens synspunkt. Derfor valgte jeg at udvikle en android applikation til dette formål i håb om at kunne få et rimeligt antal brugere til at installere applikationen og på den måde levere data til mit projekt.

Det lykkedes imidlertid aldrig at få mere end 3 personer til at installere og bruge applikationen i en længere periode. På trods af at nedenstående invitation til at bruge applikationen blev sendt pr. mail til mere end 100 DTU-studerende og applikationen blev gjort tilgængelig på android market. På grund af det lave antal erhvervede besluttede jeg forlade denne datakilde og fokusere på det datasættet indsamlet via DTUs WiFi netværk.

DTU context recognition

Dear fellow DTU student I am currently working on my master thesis in digital media engineering

The goal of my thesis is to be able to recognize social and professional contexts on and around the DTU Campus using position data collected through the eduroam Wi-Fi-network and movement (accelerometer), position and audio data gathered and analyzed on android smartphones. The smartphone datacollection is conducted by the *"DTUContextRecognizer"*-application. As mentioned above the application collects data from phone accelerometers, microphones and Wi-Fi or GPS. The data collection is performed in two background services; one for audio data and one for position and movement data. The two icons below will be displayed in the notification bar (the bar in the top of the screen).



 *Icon indicating that audio data is being collected*

 *Icon indicating that movement and position data is being collected*

Installation
In order to install the application on your phone please go to android market and search for *"DTUContextRecognizer"*.

Application notes
For this project I am only interested in data collected at DTU, so I recommend that you exit the application whenever you are not at DTU (please remember to start it again when you return to DTU:), in order to protect your privacy and battery life. Please also keep the phones Wi-Fi turned on at all time.

All data is stored locally on your phone's sdcard and the application will not transfer any recorded data.

If you have questions of any kind, please contact me by email:
christian.raaby@gmail.com or phone 31244592

Thanks in advance and kind regards
Christian Raaby

14.Appendix D

14.1. Kildekode til det primære PSMM Python script

```
from __future__ import division
import os
import csv
import numpy as np
import math
from pylab import *
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
from hcluster import pdist, linkage, dendrogram, centroid,fclusterdata
from scipy.cluster.vq import *
from scipy.stats import norm
from scipy import *
import scipy.optimize
from scipy.optimize import fsolve
import random
import copy
import time
import exceptions


def getWeekHour(CV):
    t1 = (CV-startTraining)/(7*24*3600)
    t2 = (CV-startTraining)//(7*24*3600)
    t = int(math.floor((t1-t2)*168))
    return t

def getWeekHourFromStep(CV):
    t2 = (CV)//(7*24)
    t = int(math.floor(CV-t2*168))
    return t

def getWeekDay(CV):
    t1 = (CV-startTraining)/(7*24*3600)
    t2 = (CV-startTraining)//(7*24*3600)
    t = int(math.floor((t1-t2)*7))
    return t

def getWeekDayFromStep(CV):
    t2 = (CV)//(24)
    t = int(t2)
    return t

def getDayHour(CV):
    t1 = (CV-startTraining)/(7*24*3600)
    t2 = (CV-startTraining)//(7*24*3600)
    t = int(math.floor((t1-t2)*168)-math.floor((t1-t2)*7)*24)
    return t
```

```
def getDayHourFloat(CV):
    t1 = (CV-startTraining)/(7*24*3600)
    t2 = (CV-startTraining)//(7*24*3600)
    t = (t1-t2)*168-math.floor((t1-t2)*7)*24
    return t

def getDayHourFromStep(CV):
    t2 = (CV)//(24)
    t = int(math.floor((CV-t2*24)))
    return t

def getHour(CV):
    t1 = (CV)//(24)
    t = math.ceil(CV-t1*24)
    return t

def getDay(CV):
    t1 = CV//24
    t2 = 1//7
    t = t1-t2*7
    return t

def getTime(iterNumber):
    day = getDay(iterNumber)+1
    hour = getHour(iterNumber)
    now = "Day "+str(day)+" hour "+str(hour)
    return str(now)

def imageNameGenerater(i):
    if i < 10:
        name = "00"+str(i)
    elif i < 100 and i >= 10:
        name = "0"+str(i)
    else:
        name = str(i)
    return name

def variance(a):
    if len(a) > 1:
        Ea = mean(a)
        n = len(a)
        var = sum([(x-Ea)**2 for x in a])/(n)
        if var < 0.0000001:
            var = 0.0000001
    else:
        var = 0.00004
    return var
```

```
def covariance2(a,b):
    Ea = mean(a)
    Eb = mean(b)
    c = [i*j for i, j in zip(a,b)]
    Eab = mean(c)
    cov = Eab-Ea*Eb
    return cov

def covariance(a,b):
    Ea = mean(a)
    Eb = mean(b)
    n = len(a)
    cov = sum([(i-Ea)*(j-Eb) for i, j in zip(a,b)])/(n)
    return cov

def mean(a):
    m = sum(a)/len(a)
    return m

# returns the unique elements of a list
def f5(seq, idfun=None):
    # order preserving
    if idfun is None:
        def idfun(x): return x
    seen = {}
    result = []
    for item in seq:
        marker = idfun(item)
        if marker in seen: continue
        seen[marker] = 1
        result.append(item)
    return result

def getDistance(p1,p2):
    distance = 0
    if ((p1[0]-p2[0])**2+(p1[1]-p2[1])**2)**(0.5) != 0:
        """
        Calculate the great circle distance between two points
        on the earth (specified in decimal degrees) and returns
        the distance in kilometers
        """
        lon1 = p1[0]
        lat1 = p1[1]
        lon2 = p2[0]
        lat2 = p2[1]
        # convert decimal degrees to radians
        lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
        # haversine formula
```

```
        dlon = lon2 - lon1
        dlat = lat2 - lat1
        a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
        c = 2 * math.atan2(sqrt(a), sqrt(1-a))
        distance = 6367 * c
    return distance

def getCoordinates(a):
    filePath20 = '/Users/Raaby/Desktop/Master/DTUWiFi/DTU_Buildings_coordinates_v2.csv'
    fileReader20 = csv.reader(open(filePath20, 'rb'), delimiter=';', quotechar='|')
    accesspointPositions = []

    for row20 in fileReader20:
        accesspointPositions.append(row20)

    b = []
    for i in a:
        if i[1] == "off_campus":
            b.append([i[0],55,12])
        else:
            for row in accesspointPositions:
                string = str(row[0])[1:-1]
                if string == i[1]:
                    b.append([i[0],float(row[1])/1000000,float(row[2])/1000000])
    return b

def getAccesspoints(locationCheckIns):
    filePath2 = '/Users/Raaby/Desktop/Master/DTUWiFi/DTU_Buildings_coordinates_v2.csv'

    coordinatesXY = []
    if (55.792,12.51) in locationCheckIns[0]:
        for accesspoint in locationCheckIns:
            coordinatesXY.append((accesspoint[0][0],accesspoint[0][1],accesspoint[1]))
    else:
        for accesspoint in locationCheckIns:
            fileReader2 = csv.reader(open(filePath2, 'rb'), delimiter=';', quotechar='|')
            if accesspoint[0] == "off_campus":
                coordinatesXY.append((55.792,12.51,accesspoint[1]))
            else:
                for row2 in fileReader2:
                    string = str(row2[0])[1:-1]
                    if string == accesspoint[0]:

coordinatesXY.append((float(row2[1])/1000000,float(row2[2])/1000000,accesspoint[1]))

    sortedCoordinatesXY = sorted(coordinatesXY, key=lambda item: item[2], reverse=True)

    return coordinatesXY,sortedCoordinatesXY
```

```
def getFriends(currentDevice):
    tempFilename = "social_relations_"+str(currentDevice)
    pathsocial =
'/Users/Raaby/Desktop/Master/DTUWiFi/Python/supportingDataAnalysis/Data/SocialRelations'
    filePathSocial = os.path.join(pathsocial, tempFilename)
    fileReader = csv.reader(open(filePathSocial, 'rb'), delimiter=',', quotechar='|')

    marker = 'name'
    friends = []
    coocs = []
    friendScore = []
    locations = []
    counter = -1
    first = True
    for row in fileReader:
        if first == False:
            if marker == 'name':
                friends.append(str(row))
                marker = 'cooc'
            elif marker == 'cooc':
                coocs.append(str(row)[2:-2])
                marker = 'entropy'
            elif marker == 'entropy':
                friendScore.append(row)
                marker = 'hours'
            elif marker == 'hours':
                marker = 'numlocations'
            elif marker == 'numlocations':
                marker = 'locations'
            elif marker == 'locations':
                counter += 1
                locations.append([])
                f = True
                for location in row:
                    if f:
                        locations[counter].append(str(location)[1:-1])
                    else:
                        locations[counter].append(str(location)[2:-1])
                        f = False
                marker = 'name'
        else:
            if marker == 'name':
                marker = 'cooc'
            elif marker == 'cooc':
                maxCooc = str(row)[2:-2]
                marker = 'entropy'
            elif marker == 'entropy':
```

```
                marker = 'hours'
            elif marker == 'hours':
                marker = 'numlocations'
            elif marker == 'numlocations':
                marker = 'locations'
            elif marker == 'locations':
                marker = 'name'
                first = False

    placeFriendAll = []
    timeAndPlaceTrainingFriendAll = []
    timeAndPlaceTrainingFriend_offcampusAll = []
    timeAndPlaceEvalFriendAll = []
    timeAndPlaceEvalFriend_offcampusAll = []
    friendCounter = -1
    for friend in friends:
        friendCounter += 1
        tempFilename = str(friend).strip("'[]")
        pathTrajectory = '/Users/Raaby/Desktop/Master/DTUWiFi/trajectories_full_semester'
        fileTrajectory = os.path.join(pathTrajectory, tempFilename[0:-27]+".txt")
        fileReaderTrajectory = csv.reader(open(fileTrajectory, 'rb'), delimiter=' ',
quotechar='|')

        placeFriend = []
        timeAndPlaceTrainingFriend = []
        timeAndPlaceTrainingFriend_offcampus = []
        timeAndPlaceEvalFriend = []
        timeAndPlaceEvalFriend_offcampus = []

        first = True
        for row in fileReaderTrajectory:
            if first:
                currentCheckIn = float(row[0])
                first = False

            nextCheckIn = float(row[0])
            if nextCheckIn-currentCheckIn > 3*3600:
                while nextCheckIn-currentCheckIn > 90*60:
                    if getDayHour(currentCheckIn) > 3 and getDayHour(currentCheckIn) < 23 and
getWeekDay(currentCheckIn) < 6: # lower concetration of off_campus check-ins in the regualar
workhours
                        currentCheckIn = currentCheckIn + 90*60
                    else:
                        currentCheckIn = currentCheckIn + 30*60
                    row = (currentCheckIn,"off_campus")
                    if float(row[0]) > startTraining and float(row[0]) < endTraining:
                        placeFriend.append(row[1])
                        timeAndPlaceTrainingFriend_offcampus.append([row[0],row[1]])
```

```
                    if float(row[0]) > endTraining and float(row[0]) < evalEnd:
                        timeAndPlaceEvalFriend_offcampus.append([row[0],row[1]])
                currentCheckIn = nextCheckIn

            else:
                if float(row[0]) > startTraining and float(row[0]) < endTraining:
                    placeFriend.append(row[1])
                    timeAndPlaceTrainingFriend.append([row[0],row[1]])
                if float(row[0]) > endTraining and float(row[0]) < evalEnd:
                    timeAndPlaceEvalFriend.append([row[0],row[1]])
                    timeAndPlaceEvalFriend_offcampus.append([row[0],row[1]])
                currentCheckIn = nextCheckIn

        placeFriendAll.append(getCoordinates(placeFriend))
        timeAndPlaceTrainingFriendAll.append(getCoordinates(timeAndPlaceTrainingFriend))

timeAndPlaceTrainingFriend_offcampusAll.append(getCoordinates(timeAndPlaceTrainingFriend_offca
pus))
        timeAndPlaceEvalFriendAll.append(getCoordinates(timeAndPlaceEvalFriend))

timeAndPlaceEvalFriend_offcampusAll.append(getCoordinates(timeAndPlaceEvalFriend_offcampus))

    return
friends,friendScore,timeAndPlaceEvalFriend_offcampusAll,timeAndPlaceTrainingFriendAll

def angularAverage(c):
    if c == []:
        result = 'nan'
    else:
        vx = []
        vy = []
        for element in c:
            radianElement = element*(math.pi/12)
            vx.append(math.cos(radianElement))
            vy.append(math.sin(radianElement))
        avx = sum(vx)/len(vx)
        avy = sum(vy)/len(vy)
        result = math.atan2(avy,avx)*(12/math.pi)
        if result < 0:
            result = 24 + result
    return result

def angularVariance(a):
    if len(a) > 1:
        Ea = angularAverage(a)
        n = len(a)
        var = sum([(min((24) - abs(x - Ea), abs(x - Ea)))**2 for x in a])/(n)
        if var < 1:
```

```
            var = 1
    else:
        var = 1
    return var

def temporalGuassian(t,p,tauSigWeek,dailyWeights):
    weekGauss = 0
    for day in range(len(tauSigWeek)):
        if str(tauSigWeek[day][0]) != "nan" and tauSigWeek[day][1] != 0:
            #weekGauss +=
(dailyWeights[day]/math.sqrt(2*math.pi*tauSigWeek[day][1]**2))*math.exp(-((math.pi/12)**2)*((t
tauSigWeek[day][0])**2/(2*tauSigWeek[day][1]**2)))

            weekGauss +=
dailyWeights[day]*(p/math.sqrt(2*math.pi*tauSigWeek[day][1]**2))*math.exp(-((math.pi/12)**2)*(
t-tauSigWeek[day][0])**2/(2*tauSigWeek[day][1]**2)))
    #tempGauss =
(p/math.sqrt(2*math.pi*rho**2))*math.exp(-((math.pi/12)**2)*((t-tau)**2/(2*rho**2)))

    return weekGauss

def
temporalDistribution(t,dailyAveragesLocal,dailyWeights,locations,currentAccesspoint,offCampusW
ekProbs):
    p = []
    tau = []
    sig = []
    tauSigWeek = []
    locationCounter = -1
    for location in locations:
        locationCounter += 1
        dp = float(location[1]/sum(locations))
        #if dp < (1/len(locations)):
        #   dp = 1/len(locations)
        #dp = 1
        p.append(dp)
        tau.append(dailyAveragesLocal[locationCounter][getWeekDayFromStep(t)][0])
        sig.append(dailyAveragesLocal[locationCounter][getWeekDayFromStep(t)][1])
        tauSigWeek.append(dailyAveragesLocal[locationCounter])
    temporalGuassianArray = []
    for i in range(len(p)):
        if str(tau[i]) != "nan" and sig[i] != 0:

temporalGuassianArray.append(temporalGuassian(t,p[i],tauSigWeek[i],dailyWeights[i]))
        else:
            temporalGuassianArray.append(0)
    if sum(temporalGuassianArray) == 0:
        tempDist = 0
```

```
    else:
        tempDist = temporalGuassianArray[currentAccesspoint]/sum(temporalGuassianArray)
#remember to change locations when changing back
        #tempDist =
((offCampusWeekProbs[t-2]+offCampusWeekProbs[t-1]+offCampusWeekProbs[t]+offCampusWeekProbs[t+1
+offCampusWeekProbs[t+2])/5)*(temporalGuassianArray[currentAccesspoint]/sum(temporalGuassianArr
y))
    return tempDist

def getDailyTemporalAverages(L,L_off):
    variance_mod = 3
    dailyAverages = []
    dailies = []
    dailyWeights = []
    accesspointCounter = -1
    L_all = []
    for l in L:
        L_all.append(l)
    L_all.append(L_off[0])

    for location in L_all:
        dailyAverages.append([]) #list of locations
        dailies.append([]) #list of accesspoints
        dailyWeights.append([]) #list of accesspoints
        accesspointCounter += 1
        checkIns = [[],[],[],[],[],[],[]]
        for checkIn in location[0]:
            weekDay = getWeekDay(float(checkIn))
            checkIns[weekDay].append(getDayHourFloat(float(checkIn)))

        dayCounter = -1
        for day in checkIns:
            dayCounter += 1
            dailyAverages[accesspointCounter].append([]) # list of days in accesspoint
            dailies[accesspointCounter].append([])
            for item in day:
                dailies[accesspointCounter][dayCounter].append(item)
            dailyAverages[accesspointCounter][dayCounter] =
(float(angularAverage(dailies[accesspointCounter][dayCounter]))+dayCounter*24,((angularVarianc
(dailies[accesspointCounter][dayCounter]))**(0.5))/variance_mod)

dailyWeights[accesspointCounter].append(len(dailies[accesspointCounter][dayCounter]))
    return dailyAverages,dailies,dailyWeights

# Weisstein, Eric W. "Bivariate Normal Distribution." From MathWorld--A Wolfram Web Resource.
http://mathworld.wolfram.com/BivariateNormalDistribution.html
def bivariateGaussianProabilityDistribution(x,y,sigmax,sigmay,mux,muy,cov):
    rho=cov/(sigmax*sigmay)
```

```
    if abs(rho) < 1 and rho != 0:
        z = (x-mux)**2/(sigmax**2) - (2*rho*(x-mux)*(y-muy))/(sigmax*sigmay) +
(y-muy)**2/(sigmay**2)
        p = 1/(2*math.pi*sigmax*sigmay*(1-(rho)**2)**(0.5)) * math.exp((-z)/(2*(1-rho**2)))
    else:
        p = 0

    pnorm = p/(1/(2*math.pi*sigmax*sigmay*(1-(rho)**2)**(0.5)))
    return pnorm

# hey man kan da ogsaa bare goere det paa den nemme maade
def faerdigRet(x,y,sigmax,sigmay,mux,muy,cov):
    p = mlab.bivariate_normal(x,y,sigmax,sigmay,mux,muy,cov)
    pnorm = p/(1/(2*math.pi*sigmax*sigmay*(1-(cov/(sigmax*sigmay))**2)**(0.5)))
    return pnorm

def
getOutliers(origArray,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov,O
tlierThreshold):
    allProbs = []
    for item in origArray:
        t = getWeekHour(float(item[0]))
        p = []
        for location in range(len(locations)-1):
            x = item[1]
            y = item[2]

p.append(temporalDistribution(t,dailyTemporalAverages,dailyWeights,locations,location,offCampu
WeekProbs)*faerdigRet(x,y,sigmax[location],sigmay[location],mux[location],muy[location],cov[loc
tion]))
        allProbs.append(sum(p))

    mu = mean(allProbs)
    sigma = (variance(allProbs))**(0.5)
    n = len(allProbs)
    outliers = []
    withOutOutliers = []
    # Chauvenet's criterion (almost)
    for prob,item in zip(allProbs,origArray):
        if mu-prob > 1.5*sigma:

        #c = mlab.normpdf(prob,mu,sigma)
        #print "c*n",c
        #if c < OutlierThreshold:
            outliers.append(item)
        else:
            withOutOutliers.append(item)
```

```
    return outliers,withOutOutliers

def
getOutlierThreshold(withOutOutliers,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay
mux,muy,cov):
    allProbs = []
    for item in withOutOutliers:
        t = getWeekHour(float(item[0]))
        p = []
        for location in range(len(locations)-2):
            x = item[1]
            y = item[2]

p.append(temporalDistribution(t,dailyTemporalAverages,dailyWeights,locations,location,offCampu
WeekProbs)*faerdigRet(x,y,sigmax[location],sigmay[location],mux[location],muy[location],cov[loc
tion]))
        allProbs.append(sum(p))

    mu = mean(allProbs)
    sigma = (variance(allProbs))**(0.5)
    n = len(allProbs)

    return mu,sigma,n

def calcTimeDecay(origArray):
    # run through all training check-ins
        # find last time each friend checked in to that same location
    friends,friendScore,friendActualLocation,friendTraining = getFriends(device)

    temporalDistance = []
    my_counter = 0
    for userCheckIn in origArray:
        my_counter += 1
        #print "calculating temporal decay parameter",my_counter,"of",len(origArray)
        for friend in friendTraining:
            time = 0
            for checkIn in friend:
                if float(userCheckIn[0])-float(checkIn[0]) < 0:
                    break
                if [userCheckIn[1],userCheckIn[2]]==[checkIn[1],checkIn[2]] and
float(userCheckIn[0])-float(checkIn[0]) > 0 and getWeekDay(float(userCheckIn[0])) ==
getWeekDay(float(checkIn[0])):
                    time = float(userCheckIn[0])-float(checkIn[0])
            if time != 0 and time < 24*3600:
                temporalDistance.append(time/3600)

    if temporalDistance != []:
        plt.figure(1)
```

```
        plt.clf()

        sortedTemporalDistance = sorted(temporalDistance)
        n, bins, patches = plt.hist(sortedTemporalDistance,bins=100,log=False)

        n2 = len(n)
        xmin = 1#min(temporalDistance)
        alpha = 1+n2/sum([log((x+0.01)/xmin) for x in n])
        if alpha < 0:
            alpha = -alpha
        #plt.plot(bins,bins**(-alpha),color='g')
        #expalpha = 1/(sum([x for x in n])/n2)
        #plt.plot((bins+0.01),[expalpha*math.exp(-expalpha*(x+0.01)) for x in bins],color='r')
        #plt.axis([0,max(temporalDistance)+1,0,max(n)+10])
        #tempFilename = str(device)+' - Time dacay histogram outliers.pdf'
        #filename = os.path.join(pathOut, tempFilename)
        #plt.savefig(filename)
    else:
        alpha = 0
    return alpha

def calcDistanceDecay(origArray):
    # for each cooccurence with a friend get distance between user and friend at their last
respective check-ins if within threshold
    friends,friendScore,friendActualLocation,friendTraining = getFriends(device)

    spatialDistanceCurPrev = []
    spatialDistancePrevPrev = []
    my_counter = 0
    previousUserCheckIn = 0
    for userCheckIn in origArray:
        my_counter += 1
        #print "calculating spatial decay parameter",my_counter,"of",len(origArray)
        for friend in friendTraining:
            distCurPrev = -1
            distPrevPrev = -1
            previousFriendCheckIn = 0
            for friendCheckIn in friend:
                if float(userCheckIn[0])-float(friendCheckIn[0]) < 0:
                    break
                if [userCheckIn[1],userCheckIn[2]]==[friendCheckIn[1],friendCheckIn[2]] and
float(userCheckIn[0])-float(friendCheckIn[0]) < 3600 and
float(userCheckIn[0])-float(friendCheckIn[0]) > 0:
                    if previousFriendCheckIn != 0 and previousFriendCheckIn[1] != 'off_campus':
                        distCurPrev =
getDistance([float(userCheckIn[1]),float(userCheckIn[2])],[float(previousFriendCheckIn[1]),flo
t(previousFriendCheckIn[2])])
                        if previousUserCheckIn != 0:
```

```
                            distPrevPrev =
getDistance([float(previousUserCheckIn[1]),float(previousUserCheckIn[2])],[float(previousFrien
CheckIn[1]),float(previousFriendCheckIn[2])])
                    previousFriendCheckIn = friendCheckIn
                if distCurPrev != -1:
                    spatialDistanceCurPrev.append(distCurPrev*1000)
                if distPrevPrev != -1:
                    spatialDistancePrevPrev.append(distPrevPrev*1000)
            previousUserCheckIn = userCheckIn

        #if spatialDistanceCurPrev != []:
        #   plt.figure(2)
        #   plt.clf()
        #   sortedSpatialDistance = sorted(spatialDistanceCurPrev)
        #   n,bins,patches = plt.hist(sortedSpatialDistance,bins=100,log=False)
        #   n2 = len(n)
        #   xmin = 1#min(spatialDistance)
        #   beta = 1+n2/sum([log((x+0.01)/xmin) for x in n])
        #   if beta < 0:
        #       beta = -beta
            #expbeta = 1/(sum([x for x in n])/n2)
            #plt.plot((bins+0.01),(bins+0.01)**(-beta),color='g')
            #plt.plot((bins+0.01),[expbeta*math.exp(-expbeta*(x+0.01)) for x in bins],color='r')
            #plt.axis([0,max(spatialDistanceCurPrev)+1,0,max(n)+10])
            #tempFilename = str(device)+' - Distance decay histogram outliers - CurPrev.pdf'
            #filename = os.path.join(pathOut, tempFilename)
            #plt.savefig(filename)
        #else:
        #   beta = 0

        if spatialDistancePrevPrev != []:
            plt.figure(2)
            plt.clf()
            sortedSpatialDistance = sorted(spatialDistancePrevPrev)
            n,bins,patches = plt.hist(sortedSpatialDistance,bins=100,log=False)
            n2 = len(n)
            xmin = 1#min(spatialDistance)
            beta2 = 1+n2/sum([log((x+0.01)/xmin) for x in n])
            if beta2 < 0:
                beta2 = -beta2
            #expbeta = 1/(sum([x for x in n])/n2)
            #plt.plot((bins+0.01),(bins+0.01)**(-beta2),color='g')
            #plt.plot((bins+0.01),[expbeta*math.exp(-expbeta*(x+0.01)) for x in bins],color='r')
            #plt.axis([0,max(spatialDistancePrevPrev)+1,0,max(n)+10])
            #tempFilename = str(device)+' - Distance decay histogram outliers - PrevPrev.pdf'
            #filename = os.path.join(pathOut, tempFilename)
            #plt.savefig(filename)
        else:
```

```
        beta2 = 0

    return beta2

def socialDistribution(X,t,alpha,beta,approvedCheckIns,friendScore):
    # outliers bruges bare til at lave den temporale fordeling og en gang for alle (laes. hver
bruger) bestemme alpha og beta
    # naar der skal bestemmes gat efter fitting har social delen en temporal del ligesom
off-campus
    # og naar denne har den hoejeste sadnsynligehd anvendes nedenstaaende fordeling til at
komme med et gaet
    # maaske skal alpha og beta ogsaa kun fittes efter 24 timer check-in
    # the sum of all friends check-ins with-in the last 12 or 24 hours (since 12 midnight)
times the friends score of each friend
    # for friend in frieds
        #friendScore * sum(timeDist**(-alpha) * spatDist**(-beta))
    score = 0
    scoreDelta = 0
    friendCounter = -1
    scoreNormalizer = 0
    scoreNormalizerDelta = 0
    for friend in approvedCheckIns:
        friendCounter += 1
        for checkIn in friend:
            spatDist = getDistance([X[0],X[1]],[float(checkIn[1]),float(checkIn[2])])
            tempDist = abs(t-float(checkIn[0]))
            if spatDist < 1:
                scoreNormalizer += float(friendScore[friendCounter][0])
                if spatDist == 0 and tempDist == 0:
                    score += float(friendScore[friendCounter][0])
                elif spatDist == 0:
                    score += float(friendScore[friendCounter][0]) * tempDist**(-alpha)
                elif tempDist == 0:
                    score += float(friendScore[friendCounter][0]) * spatDist**(-beta)
                else:
                    score += float(friendScore[friendCounter][0]) * tempDist**(-alpha) *
spatDist**(-beta)
            #
            if spatDist < 0.01:
                scoreNormalizerDelta += float(friendScore[friendCounter][0])
                if tempDist == 0:
                    scoreDelta += float(friendScore[friendCounter][0])
                else:
                    scoreDelta += float(friendScore[friendCounter][0]) * tempDist**(-alpha)

    if scoreDelta != 0:
        scoreDelta = scoreDelta/scoreNormalizerDelta
    if score != 0:
```

```
        score = score/scoreNormalizer
    return score,scoreDelta

def getRandomStartingPoint(origArray,numLocations):
    L = []
    for i in range(numLocations):
        L.append([[],[],[]])
    # put all check-ins in "random" piles
    marker = 0
    size = len(origArray)
    whileCounter = 0
    while size:
        size -= 1
        index = random.randint(0, size)
        item = origArray[index]
        origArray[index] = origArray[size]
        whileCounter += 1

        jazzy = whileCounter-numLocations*(whileCounter//numLocations)
        L[jazzy][0].append(item[0])
        L[jazzy][1].append(item[1])
        L[jazzy][2].append(item[2])
    return L

def getFixedStartingPoint(origArray,uniqueCheckInCoords):
    L = []
    for i in range(len(uniqueCheckInCoords)):
        L.append([[],[],[]])

    for item in origArray:
        coords = [item[1],item[2]]
        index = uniqueCheckInCoords.index(coords)
        L[index][0].append(item[0])
        L[index][1].append(item[1])
        L[index][2].append(item[2])

    needSpecialVar = True
    return L,needSpecialVar

def
expectationMaximization(origArray,offcampus,num_locations,evalArray,offCampusWeekProbs,uniqueC
eckInCoords):
    varianceModifier = 0.01
    spatialImportance = 1
    OutlierThreshold = 0.5
    exceptionCounter = 0
    delta = True
    draw = False
```

```
    training = True
    pBest = 0;
    for numLocationsStart in range(10,11):#range(0,num_locations-1):
        numLocations = numLocationsStart#num_locations-numLocationsStart
        print numLocations

        for trythis in range(10):
            numLocations = numLocationsStart#num_locations-numLocationsStart
            print "Attempt:",exceptionCounter+1,"of 5"

            L_off = [[[],[],[]]]
            for item in offcampus:
                L_off[0][0].append(item[0])
                L_off[0][1].append(item[1])
                L_off[0][2].append(item[2])

            needSpecialVar = False
            L = getRandomStartingPoint(origArray,numLocations,)
            #L,needSpecialVar = getFixedStartingPoint(origArray,uniqueCheckInCoords)


            locationSizes_old = [0,0,0]
            locationSizes_new = [1,1,1]
            centroids_old = [[0,0]]
            centroids_new = [[1,1]]
            iterationCounter = 0
            pTotalOld = 0
            pTotalNew = 1
            try:
                while pTotalNew > pTotalOld: #locationSizes_old != locationSizes_new and
centroids_old != centroids_new:
                    iterationCounter += 1
                    print "EM iteration",iterationCounter
                    locationSizes_old = locationSizes_new
                    centroids_old = centroids_new
                    pTotalOld = pTotalNew

                    for i in L:
                        if i == [[],[],[]]:
                            L.remove(i)
                            numLocations = len(L)
                            print 'something has been removed'

                    # calculate mean, variance, and covariance for the "piles"
                    locationSizes_new = []
                    centroids_new = []
                    sigmax = []
                    sigmay = []
```

```
                    mux = []
                    muy = []
                    cov = []
                    for i in range(len(L)):
                        locationSizes_new.append(len(L[i][1]))
                        if needSpecialVar:
                            sigmax.append((0.000000000001)**(0.5)/varianceModifier)
                            sigmay.append((0.000000000001)**(0.5)/varianceModifier)
                            needSpecialVar = False
                        else:
                            sigmax.append((variance(L[i][1]))**(0.5)/varianceModifier)
                            sigmay.append((variance(L[i][2]))**(0.5)/varianceModifier)
                        mux.append(mean(L[i][1]))
                        muy.append(mean(L[i][2]))
                        cov.append(covariance(L[i][1],L[i][2]))
                    centroids_new = [mux,muy]
                    print cov
                    print sigmax
                    print sigmay
                    print "locationSizes",locationSizes_new
                    #print "centroids",centroids_new

                    # t should be given i weekhour 0 - 167
                    # locations is the 3 piles and should just hold the total number of
check-ins in that pile
                    # currentAccesspoint is just a number idicating which pile's
temporaldistribution that is wanted
                    # dailyAverages i a mother fucker!!!
                    dailyTemporalAverages,dailies,dailyWeights =
getDailyTemporalAverages(L,L_off)

                    locations = []
                    for location in L:
                        locations.append([0,len(location[0])])
                    for location in L_off:
                        locations.append([0,len(location[0])])

                    if draw:

drawTemporalPlots(locations,dailyTemporalAverages,dailies,dailyWeights,L,mux,muy,str(iteration
ounter))

                    # redistribute check-ins
                    L = []
                    for i in range(numLocations):
                        L.append([[],[],[]])
                    pTotalNew = 0
                    for item in origArray:
```

```
                        t = getWeekHour(float(item[0]))
                        p = []
                        # remember that the last part of most of the list is offcampus that are
not to count in the spatial distribution
                        for location in range(len(L)):
                            x = item[1]
                            y = item[2]

p.append(temporalDistribution(t,dailyTemporalAverages,dailyWeights,locations,location,offCampu
WeekProbs)*(faerdigRet(x,y,sigmax[location],sigmay[location],mux[location],muy[location],cov[lo
ation]))**spatialImportance)
                        pTotalNew += max(p)
                        L_index = p.index(max(p))
                        L[L_index][0].append(item[0])
                        L[L_index][1].append(item[1])
                        L[L_index][2].append(item[2])
                    #
                    for i in L:
                        if i == [[],[],[]]:
                            L.remove(i)
                            numLocations = len(L)
                            print 'something has been removed'

                    locations = []
                    for location in L:
                        locations.append([0,len(location[0])])
                    for location in L_off:
                        locations.append([0,len(location[0])])
                    print "loc",len(locations)
                    print "daily",len(dailyTemporalAverages)
                    print "sigmax",len(sigmax)

                    print pTotalNew
                    print pBest
                    if pTotalNew > pBest:
                        pBest = pTotalNew; locationsBest = locations; dailyTemporalAveragesBest
= dailyTemporalAverages; dailyWeightsBest = dailyWeights; sigmaxBest = sigmax; sigmayBest =
sigmay; covBest = cov;numberoflocations = numLocations

            except ValueError:
                print "value exception"
                exceptionCounter += 1

    #if exceptionCounter == 10:
    #
    dailyTemporalAverages = dailyTemporalAveragesBest; locations = locationsBest; dailyWeights
= dailyWeightsBest; sigmax = sigmaxBest; sigmay = sigmayBest; cov = covBest;
    print "pBest",pBest
```

```
    print "finding outliers"

    print "get outliers here in you know"
    print "locations number",len(locations)
    print "len daily averages",len(dailyTemporalAverages)
    print "sigmax",len(sigmax)

    outliers,withOutOutliers =
getOutliers(origArray,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov,O
tlierThreshold)
    print "calculating outlier distribution"
    alpha = 0
    beta = 0
    socialWeight = 0
    if len(outliers) != 0:
        alpha = calcTimeDecay(outliers)
        beta = calcDistanceDecay(outliers)

        # for outliers calculate periodic guess and social guess and find ratio between
guesses.
        # calculate mean of all ratios and use as weight
        print "calculating social weight"
        socialWeight,muIn,muOut =
calculateSocialWeight(outliers,withOutOutliers,dailyTemporalAverages,dailyWeights,locations,si
max,sigmay,mux,muy,cov,alpha,beta,training,friends,friendScore,friendActualLocation,friendTrain
ng)
        print socialWeight
        if muOut > muIn:
            print "socialWeight exception!"
            exceptionCounter = 0

        print "redistributing check-ins"
        # redistribute check-ins
        L = []
        for i in range(len(locations)):
            L.append([[],[],[]])

        print "get outliers here in you know"
        print "locations number",len(locations)
        print "len daily averages",len(dailyTemporalAverages)
        print "sigmax",len(sigmax)
        for item in withOutOutliers:
            t = getWeekHour(float(item[0]))
            p = []
            # remember that the last part of most of the list is offcampus that are not to
count in the spatial distribution
            for location in range(len(L)-1):
                x = item[1]
```

```
                y = item[2]

p.append(temporalDistribution(t,dailyTemporalAverages,dailyWeights,locations,location,offCampu
WeekProbs)*(faerdigRet(x,y,sigmax[location],sigmay[location],mux[location],muy[location],cov[lo
ation]))**spatialImportance)
            L_index = p.index(max(p))
            L[L_index][0].append(item[0])
            L[L_index][1].append(item[1])
            L[L_index][2].append(item[2])

        for item in outliers:
            L[-1][0].append(item[0])
            L[-1][1].append(item[1])
            L[-1][2].append(item[2])

        for i in L:
            if i == [[],[],[]]:
                L.remove(i)
                numLocations = len(L)
                print 'something has been removed'

    # calculate mean, variance, and covariance for the "piles"
    locationSizes_new = []
    centroids_new = []
    sigmax = []
    sigmay = []
    mux = []
    muy = []
    cov = []
    for i in range(len(L)):
        locationSizes_new.append(len(L[i][1]))
        sigmax.append((variance(L[i][1]))**(0.5)/varianceModifier)
        sigmay.append((variance(L[i][2]))**(0.5)/varianceModifier)
        mux.append(mean(L[i][1]))
        muy.append(mean(L[i][2]))
        cov.append(covariance(L[i][1],L[i][2]))
    centroids_new = [mux,muy]

    locations = []
    for location in L:
        locations.append([0,len(location[0])])
    for location in L_off:
        locations.append([0,len(location[0])])

    locationSizes_new = []
    for i in range(len(L)):
        locationSizes_new.append(len(L[i][1]))
    print "locationSizes",locationSizes_new
```

```
    dailyTemporalAverages,dailies,dailyWeights = getDailyTemporalAverages(L,L_off)
    muOutlierThreshold,sigmaOutlierThreshold,nOutlierThreshold =
getOutlierThreshold(withOutOutliers,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay
mux,muy,cov)

    print "drawing temporal plots"
    if draw:

drawTemporalPlots(locations,dailyTemporalAverages,dailies,dailyWeights,L,mux,muy,"IsolatedOutl
ers")

    print "evaluating model"
    meanDistCombined,meanDistPeriodic,meanDistSocial,meanDistSocialDelta,meanCombinedWeigthedD
st,meanAccesspointParallel,meanAccesspointGuessWeighted,meanAccesspointGuess,exactFitCombinedRa
io,exactFitPeriodicRatio,exactFitSocialRatio,offCampusSuccessRatio,onCampusCheckIns,actualOffCa
pusCounter,socialCheckInCounter =
evaluateModel(L,evalArray,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,c
v,alpha,beta,muOutlierThreshold,sigmaOutlierThreshold,nOutlierThreshold,OutlierThreshold,delta,
raw,socialWeight)
    print "check-ins on/off campus:",onCampusCheckIns,"/",actualOffCampusCounter
    print "percentage of offcampus check-ins successfully guessed:",offCampusSuccessRatio
    print "average distance between guess and actual position in meters
(meanDistCombined):",1000*meanDistCombined,1000*meanAccesspointParallel
    print "average distance between guess and actual position in meters
(meanDistPeriodic):",1000*meanDistPeriodic,1000*meanAccesspointGuess
    print "average distance between guess and actual position in meters
(meanDistSocial):",1000*meanDistSocial
    print "average distance between guess and actual position in meters
(meanDistSocialDelta):",1000*meanDistSocialDelta
    print "average distance between guess and actual position in meters
(meanCombinedWeigthedDist):",1000*meanCombinedWeigthedDist,1000*meanAccesspointGuessWeighted
    print "percentage of exact (less than 10 meters on campus) guesses
(exactFitCombinedRatio):",exactFitCombinedRatio
    print "percentage of exact (less than 10 meters on campus) guesses
(exactFitPeriodicRatio):",exactFitPeriodicRatio
    print "percentage of exact (less than 10 meters on campus) guesses
(exactFitSocialRatio):",exactFitSocialRatio

    return
meanDistCombined,meanDistPeriodic,meanDistSocial,meanAccesspointParallel,meanAccesspointGuessW
ighted,meanAccesspointGuess,exactFitCombinedRatio,exactFitPeriodicRatio,exactFitSocialRatio,onC
mpusCheckIns,actualOffCampusCounter,offCampusSuccessRatio,meanDistSocialDelta,meanCombinedWeigt
edDist,numberoflocations,socialCheckInCounter,pBest



def
```

```
evaluateModel(L,evalArray,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,c
v,alpha,beta,muOutlierThreshold,sigmaOutlierThreshold,nOutlierThreshold,OutlierThreshold,delta,
raw,socialWeight):
    distSocialError = []
    distSocialDeltaError = []
    combinedWeigthedDistError = []
    distCombinedError = []
    distPeriodicError = []
    exactFitsCombined = 0
    exactFitsPeriodic = 0
    exactFitsSocial = 0
    actualOffCampusCounter = 0
    guessOffCampus = 0
    offCampusSuccess = 0
    checkInCounter = 0
    accesspointGuessWeightedError = []
    accesspointGuessError = []
    accesspointParallelError = []
    friends,friendScore,friendActualLocation,friendTraining = getFriends(device)
    training = False
    socialCheckInCounter = 0
    onCampusCheckIns = 0
    for checkIn in evalArray:
        onCampus = True
        checkInCounter += 1
        print "eval check in",checkInCounter,"of",len(evalArray)
        t = getWeekHour(float(checkIn[0]))
        actualPosition = [float(checkIn[1]),float(checkIn[2])]

        if actualPosition == [55.0,12.0]:
            actualOffCampusCounter += 1
            actualPosition = "off_campus"

        if
temporalDistribution(t,dailyTemporalAverages,dailyWeights,locations,len(locations)-1,offCampus
eekProbs) > 0.6: #len(locations)-1 is the last element since 0 also counts
            onCampus = False
            guessOffCampus += 1
            guess = "off_campus"
            if actualPosition == guess:
                offCampusSuccess += 1
        else:
            if actualPosition != "off_campus":
                offCampusSuccess += 1

        if actualPosition != "off_campus":
            onCampusCheckIns += 1
            socialCheckIn = False
```

```
            # if guess from the periodic model is not confident enough according to the
Chauvenet's criterion the check-in will be treated as a social checkin
            p = []
            for location in range(len(locations)-2):
                x = checkIn[1]
                y = checkIn[2]

p.append(temporalDistribution(t,dailyTemporalAverages,dailyWeights,locations,location,offCampu
WeekProbs)*faerdigRet(x,y,sigmax[location],sigmay[location],mux[location],muy[location],cov[loc
tion]))
            outlierProb = sum(p)
            # Chauvenet's criterion (almost)
            #c = mlab.normpdf(outlierProb,muOutlierThreshold,sigmaOutlierThreshold)
            if muOutlierThreshold-outlierProb > 1.5*sigmaOutlierThreshold:
            #if c < OutlierThreshold:
                socialguessDelta,maxScore =
getSocialFuncMax(float(checkIn[0]),alpha,beta,training,friends,friendScore,friendActualLocatio
,friendTraining,True)
                if maxScore > outlierProb:
                    socialCheckIn = True
                    socialCheckInCounter += 1

            socialguessDelta,maxScore =
getSocialFuncMax(float(checkIn[0]),alpha,beta,training,friends,friendScore,friendActualLocatio
,friendTraining,True)
            if socialCheckIn:
                socialguessDelta,maxScore =
getSocialFuncMax(float(checkIn[0]),alpha,beta,training,friends,friendScore,friendActualLocatio
,friendTraining,True)
                socialguess,maxScore =
getSocialFuncMax(float(checkIn[0]),alpha,beta,training,friends,friendScore,friendActualLocatio
,friendTraining,False)
                #calculate distance error
                distSocial =
getDistance([float(socialguess[0]),float(socialguess[1])],actualPosition)
                distSocialDelta =
getDistance([float(socialguessDelta[0]),float(socialguessDelta[1])],actualPosition)

            guess,maxScore =
getfuncMax(t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov)
            #calculate distance error
            distPeriodic = getDistance([float(guess[0]),float(guess[1])],actualPosition)

            combinedWeigthedGuess =
[(guess[0]+socialWeight*socialguessDelta[0])/(socialWeight+1),(guess[1]+socialWeight*socialgue
sDelta[1])/(socialWeight+1)]
            combinedWeigthedDist =
```

```
getDistance([float(combinedWeigthedGuess[0]),float(combinedWeigthedGuess[1])],actualPosition)
            combinedWeigthedDistError.append(combinedWeigthedDist)
            if draw:
                text = 'periodic'

contourPlot(L,t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov,guess,t
xt,checkInCounter)

            accesspointGuessWeighted =
getNearestKnownAccesspoint([float(combinedWeigthedGuess[0]),float(combinedWeigthedGuess[1])],a
tualPosition)
            dist1 =
getDistance([float(accesspointGuessWeighted[0]),float(accesspointGuessWeighted[1])],actualPosi
ion)
            accesspointGuessWeightedError.append(dist1)
            accesspointGuess =
getNearestKnownAccesspoint([float(guess[0]),float(guess[1])],actualPosition)
            dist2 =
getDistance([float(accesspointGuess[0]),float(accesspointGuess[1])],actualPosition)
            accesspointGuessError.append(dist2)
            if socialCheckIn:
                accesspointParallel =
getNearestKnownAccesspoint(socialguessDelta,actualPosition)
            else:
                accesspointParallel = accesspointGuess
            dist3 =
getDistance([float(accesspointParallel[0]),float(accesspointParallel[1])],actualPosition)
            accesspointParallelError.append(dist3)

            if socialCheckIn:
                distCombined = distSocialDelta
                distSocialError.append(distSocialDelta)
                distSocialDeltaError.append(distSocialDelta)
                distCombinedError.append(distCombined)
                if distSocial < 0.010:
                    exactFitsSocial += 1

            else:
                distCombined = distPeriodic

distPeriodicError.append(getDistance([float(guess[0]),float(guess[1])],actualPosition))
                distCombinedError.append(distCombined)

            if dist3 < 0.010:
                exactFitsCombined += 1

            if dist2 < 0.010:
                exactFitsPeriodic += 1
```

```
    if offCampusSuccess != 0:
        offCampusSuccessRatio = offCampusSuccess/len(evalArray)
    else:
        offCampusSuccessRatio = 0
    meanDistCombined = mean(distCombinedError)
    meanAccesspointParallel = mean(accesspointParallelError)
    meanDistPeriodic = mean(distPeriodicError)
    meanAccesspointGuess = mean(accesspointGuessError)
    meanDistSocial = mean(distSocialError)
    meanDistSocialDelta = mean(distSocialDeltaError)
    meanCombinedWeigthedDist = mean(combinedWeigthedDistError)
    meanAccesspointGuessWeighted = mean(accesspointGuessWeightedError)
    if onCampusCheckIns != 0:
        exactFitCombinedRatio = exactFitsCombined/onCampusCheckIns
        exactFitPeriodicRatio = exactFitsPeriodic/onCampusCheckIns
    else:
        exactFitCombinedRatio = 0
        exactFitPeriodicRatio = 0
    if socialCheckInCounter != 0:
        exactFitSocialRatio = exactFitsSocial/socialCheckInCounter
    else:
        exactFitSocialRatio = 0

    return
meanDistCombined,meanDistPeriodic,meanDistSocial,meanDistSocialDelta,meanCombinedWeigthedDist,
eanAccesspointParallel,meanAccesspointGuessWeighted,meanAccesspointGuess,exactFitCombinedRatio,
xactFitPeriodicRatio,exactFitSocialRatio,offCampusSuccessRatio,onCampusCheckIns,actualOffCampus
ounter,socialCheckInCounter

def getNearestKnownAccesspoint(bestguess,actualPosition):
    shortestDist = 10000000
    for coords in uniqueCheckInCoords:
        distance = getDistance(coords,bestguess)
        if float(distance) < shortestDist:
            shortestDist = float(distance)
            bestCoords = coords
    return bestCoords

def probDist(X,t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov):
    p = 0
    for location in range(len(locations)-2): #excluding offcampus and outliers
        p +=
temporalDistribution(t,dailyTemporalAverages,dailyWeights,locations,location,offCampusWeekProb
)*faerdigRet(X[0],X[1],sigmax[location],sigmay[location],mux[location],muy[location],cov[locati
n])
```

```
    return -p

def getfuncMax(t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov):
    params = [t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov]
    maxcoords,maxScore,otherStuff =
scipy.optimize.fmin_l_bfgs_b(probDist,[55.7865,12.52],args=params,approx_grad=True)
    return maxcoords, maxScore

#needs to be there for the optimization function to work...
def fcons(X,t,alpha,beta,approvedCheckIns,friendScore):
    return X[0]

def socialMaxhelp(X,t,alpha,beta,approvedCheckIns,friendScore):
    score,scoreDelta = socialDistribution(X,t,alpha,beta,approvedCheckIns,friendScore)
    return -score

def
getSocialFuncMax(t,alpha,beta,training,friends,friendScore,friendActualLocation,friendTraining
delta):
    if training:
        friends = friendTraining
    else:
        friends = friendActualLocation

    approvedCheckIns = []
    friendCounter = -1
    for friend in friends:
        friendCounter += 1
        approvedCheckIns.append([])
        for checkIn in friend:
            if t-float(checkIn[0]) < 0:
                break
            if t-float(checkIn[0]) > 0 and t-float(checkIn[0]) < 24*3600 and getWeekDay(t) ==
getWeekDay(float(checkIn[0])):
                approvedCheckIns[friendCounter].append(checkIn)
    params = [t,alpha,beta,approvedCheckIns,friendScore]
    if delta:
        coordScores = []
        for coords in uniqueCheckInCoords:
            score,scoreDelta =
socialDistribution(coords,t,alpha,beta,approvedCheckIns,friendScore)
            coordScores.append((coords,scoreDelta))
        maxcoords = max(coordScores, key=lambda x: x[1])[0]
        maxScore = max(coordScores, key=lambda x: x[1])[1]
    else:
        maxcoords = scipy.optimize.fmin_cobyla(socialMaxhelp,[55.7865,12.52],fcons,args=params)
        maxScore = -socialMaxhelp(maxcoords,t,alpha,beta,approvedCheckIns,friendScore)
    #maxcoords,maxVal,otherStuff =
```

```
scipy.optimize.fmin_l_bfgs_b(socialDistribution,[55.7865,12.52],args=params,approx_grad=True)
    return maxcoords,maxScore

def
contourPlot(L,t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov,guess,t
xt,checkInCounter):
    colors = [[1,0,0],[1,1,0],[0,1,1],[0,0,1],[1,0,1]]

    #colors =
[[1,0,0],[1,0.33,0],[1,0.66,0],[1,1,0],[0,1,0.66],[0,1,1],[0,0,1],[0.33,0,1],[0.66,0,1],[1,0,1
]
    loc = -1
    plt.figure(10)
    plt.clf()
    for l,MUX,MUY in zip(L,mux,muy):
        loc += 1
        plt.scatter([x+0.0001*random.uniform(-1,1) for x in
l[1]],[y+0.0001*random.uniform(-1,1) for y in
l[2]],s=20,color=colors[loc],marker='o',label='check-ins')
        plt.scatter(MUX,MUY,s=120,color=colors[loc],marker='x',label='Centroid')
    plt.scatter(guess[0],guess[1],s=120,color='c',marker='s',label='guess')
    #plt.scatter([x[1]+0.0001*random.uniform(-1,1) for x in
approvedCheckIns],[y[2]+0.0001*random.uniform(-1,1) for y in
approvedCheckIns],s=20,color='c',marker='o',label='friends')

    Z = []
    for location in range(len(locations)-2): #excluding offcampus

Z.append(temporalDistribution(t,dailyTemporalAverages,dailyWeights,locations,location,offCampu
WeekProbs)*mlab.bivariate_normal(X,Y,sigmax[location],sigmay[location],mux[location],muy[locati
n],cov[location]))

    Zet = Z[0]
    first = True
    for i in Z:
        if first:
            first = False
        else:
            Zet += i
    plt.contour(X, Y, Zet)

    legend(loc="upper right",shadow=True, fancybox=True)
    leg = plt.gca().get_legend()
    ltext  = leg.get_texts()  # all the text.Text instance in the legend
    plt.setp(ltext, fontsize=6)    # the legend text fontsize
    plt.axis('scaled')
    plt.axis([55.792,55.778,12.51,12.53])
    plt.title('Time independent spatial distribution')
```

```
    plt.xlabel('longitude')
    plt.ylabel('latitude')
    tempFilename = str(device)+'_Eval_Spatial_'+str(text)+"_"+str(checkInCounter)+'.pdf'
    filename = os.path.join(pathOut, tempFilename)
    plt.savefig(filename)
    #plt.show()

def
calculateSocialWeight(outliers,withOutOutliers,dailyTemporalAverages,dailyWeights,locations,si
max,sigmay,mux,muy,cov,alpha,beta,training,friends,friendScore,friendActualLocation,friendTrain
ng):
    # ratio when social score is highest on the users position and ratio when the social score
i lowest
    # friend attraction should not allways add up to one but depend on the average co-occurence
ratio.
    # I guess one could say that this is somehow worked into the weighting though i would seem
that it is kind of opposite

    ratioOut = []
    socialArray = []
    periodicArray = []
    for outlier in outliers:
        t = getWeekHour(float(outlier[0]))
        approvedCheckIns = []
        friendCounter = -1
        for friend in friendTraining:
            friendCounter += 1
            approvedCheckIns.append([])
            for checkIn in friend:
                if float(outlier[0])-float(checkIn[0]) < 0:
                    break
                if float(outlier[0])-float(checkIn[0]) > 0 and
float(outlier[0])-float(checkIn[0]) < 24*3600 and getWeekDay(t) ==
getWeekDay(float(checkIn[0])):
                    approvedCheckIns[friendCounter].append(checkIn)
        #guess,score =
getfuncMax(t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov)
        #socialguessDelta,scoreSocial =
getSocialFuncMax(float(outlier[0]),alpha,beta,training,friends,friendScore,friendActualLocatio
,friendTraining,True)
        jimmy,scoreSocial =
socialDistribution([outlier[1],outlier[2]],float(outlier[0]),alpha,beta,approvedCheckIns,frien
Score)
        score =
probDist([outlier[1],outlier[2]],t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,
ux,muy,cov)
        if scoreSocial > 0.01:
            socialArray.append(scoreSocial)
```

```
        if score != 0:
            periodicArray.append(-score)
        if scoreSocial > 0.01 and score != 0:
            ratioOut.append(-score/scoreSocial)
    if socialArray == [] or ratioOut == []:
        xOpt = 0
        muOut = 0
        muIn = 1
    else:
        muOut2 = mean(ratioOut)
        muOut = mean(periodicArray)/mean(socialArray)
        print "meanOut",muOut
        sigmaOut = variance(ratioOut)**(0.5)*(muOut/muOut2)

        ratioIn = []
        socialArray = []
        periodicArray = []
        for inlier in withOutOutliers:
            t = getWeekHour(float(inlier[0]))
            approvedCheckIns = []
            friendCounter = -1
            for friend in friendTraining:
                friendCounter += 1
                approvedCheckIns.append([])
                for checkIn in friend:
                    if float(inlier[0])-float(checkIn[0]) < 0:
                        break
                    if float(inlier[0])-float(checkIn[0]) > 0 and
float(inlier[0])-float(checkIn[0]) < 24*3600 and getWeekDay(t) ==
getWeekDay(float(checkIn[0])):
                        approvedCheckIns[friendCounter].append(checkIn)
            #guess,score =
getfuncMax(t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mux,muy,cov)
            #socialguessDelta,scoreSocial =
getSocialFuncMax(float(inlier[0]),alpha,beta,training,friends,friendScore,friendActualLocation
friendTraining,True)
            jimmy,scoreSocial =
socialDistribution([inlier[1],inlier[2]],float(inlier[0]),alpha,beta,approvedCheckIns,friendSc
re)
            score =
probDist([inlier[1],inlier[2]],t,dailyTemporalAverages,dailyWeights,locations,sigmax,sigmay,mu
,muy,cov)
            if scoreSocial > 0.01:
                socialArray.append(scoreSocial)
            if score != 0:
                periodicArray.append(-score)
            if scoreSocial > 0.01 and -score > 0.01:
                ratioIn.append(-score/scoreSocial)
```

```
        if socialArray == [] or ratioIn == []:
            xOpt = 0
            muOut = 0
            muIn = 1
        else:
            muIn2 = mean(ratioIn)
            muIn = mean(periodicArray)/mean(socialArray)
            print "meanIn",muIn
            sigmaIn = variance(ratioIn)**(0.5)*(muIn/muIn2)

            if muIn > muOut:
                myMin = muOut
                myMax = muIn
            else:
                myMin = muIn
                myMax = muOut

            xOpt = fsolve(lambda x : mlab.normpdf(x,muOut,sigmaOut) -
mlab.normpdf(x,muIn,sigmaIn),(muIn+muOut)/2)
            if xOpt < myMin or xOpt > myMax:
                xOpt = (muIn+muOut)/2

    print xOpt,muIn,muOut
    return xOpt,muIn,muOut

def
drawTemporalPlots(sortedAccesspoints,dailyTemporalAverages,dailies,dailyWeights,L,mux,muy,post
ix):
        plt.figure(1)
        plt.clf()
        plt.figure(2)
        plt.clf()
        plt.figure(3)
        plt.clf()

        print "der tegnes temporale plots... Vaersgooo!!!"
        # spatial distribution of training set
        colors = [[1,0,0],[1,1,0],[0,1,1],[0,0,1],[1,0,1]]

        #colors =
[[1,0,0],[1,0.33,0],[1,0.66,0],[1,1,0],[0,1,0.66],[0,1,1],[0,0,1],[0.33,0,1],[0.66,0,1],[1,0,1
]
        loc = -1
        plt.figure(10)
        plt.clf()
        for l,MUX,MUY in zip(L,mux,muy):
            loc += 1
            plt.scatter([x+0.0001*random.uniform(-1,1) for x in
```

```
l[1]],[y+0.0001*random.uniform(-1,1) for y in
l[2]],s=20,color=colors[loc],marker='o',label='check-ins')
        loc = -1
        for l,MUX,MUY in zip(L,mux,muy):
            loc += 1
            plt.scatter(MUX,MUY,s=120,color=colors[loc],marker='x',label='Centroid')
        #legend(loc="upper right",shadow=True, fancybox=True)
        #leg = plt.gca().get_legend()
        #ltext  = leg.get_texts()  # all the text.Text instance in the legend
        #plt.setp(ltext, fontsize=6)    # the legend text fontsize
        plt.axis('scaled')
        plt.axis([55.792,55.778,12.51,12.53])
        plt.title('spatial distribution')
        plt.xlabel('longitude')
        plt.ylabel('latitude')
        tempFilename = str(device)+'_EM_FIG_'+str(postfix)+'.pdf'
        filename = os.path.join(pathOut, tempFilename)
        plt.savefig(filename)

        # temporal function distribution
        xet = range(168)
        ca = 0
        for ca in range(len(dailies)):
            plt.figure(2)
            if ca == (len(dailies)-1):
                ca = -1

plt.plot(xet,[temporalDistribution(x,dailyTemporalAverages,dailyWeights,sortedAccesspoints,ca,
ffCampusWeekProbs) for x in xet],color=colors[ca])
        tempFilename = str(device)+"temporal_distribution_"+str(postfix)+".pdf"
        filename = os.path.join(pathOut, tempFilename)
        plt.axis([0,170,0,1.1])
        plt.title('time dependant location probability')
        plt.xlabel('weekly hour')
        plt.ylabel('probability')
        plt.savefig(filename)


        # temporal distribution of training set
        for ca in range(len(dailies)):
            yet = []
            xerne = []
            icounter = -1
            for i in dailies[ca]:
                icounter += 1
                for j in i:
                    xerne.append(j+icounter*24)
                    yet.append(1*ca)
```

```
            plt.figure(3)
            if ca == (len(dailies)-1):
                ca = -1
            plt.scatter(xerne,[b+0.2*random.uniform(-1,1) for b in yet],color=colors[ca])
        plt.title('temporal distribution')
        plt.xlabel('weekly hour')
        plt.axis([0,170,-1,max(yet)+1])
        tempFilename = str(device)+"check_in_distribution_"+str(postfix)+".pdf"
        filename = os.path.join(pathOut, tempFilename)
        plt.savefig(filename)


def buildMFLModel(timeAndPlaceTraining):
    checkInCoords = getCoordinates(timeAndPlaceTraining)
    uniqueCheckInCoords = []
    for checkIn in checkInCoords:
        if [checkIn[1],checkIn[2]] not in uniqueCheckInCoords:
            uniqueCheckInCoords.append([checkIn[1],checkIn[2]])
    locations = []
    locationCounter = -1
    for coords in uniqueCheckInCoords:
        locationCounter += 1
        locations.append([])
        for checkIn in checkInCoords:
            if [checkIn[1],checkIn[2]] == coords:
                locations[locationCounter].append(checkIn)
    weeklyCheckIns = []
    locationCounter = -1
    for location in locations:
        locationCounter += 1
        weeklyCheckIns.append([])
        for i in range(168):
            weeklyCheckIns[locationCounter].append(0)
        for checkIn in location:
            weekHour = getWeekHour(float(checkIn[0]))
            weeklyCheckIns[locationCounter][weekHour] += 1

    return weeklyCheckIns,uniqueCheckInCoords

def MostFrequentedLocation(timeAndPlaceTraining,timeAndPlaceEval):
    #very like temporal model
    weeklyCheckIns,uniqueCheckInCoords = buildMFLModel(timeAndPlaceTraining)
    distCombinedError = []
    evalCheckIns = getCoordinates(timeAndPlaceEval)
    exactFits = 0
    totalCheckInsEval = 0
    for checkIn in evalCheckIns:
        totalCheckInsEval += 1
```

```
        t = getWeekHour(float(checkIn[0]))
        a = []
        for i in range(len(uniqueCheckInCoords)):
            a.append(weeklyCheckIns[i][t])
        guess = uniqueCheckInCoords[a.index(max(a))]
        dist = getDistance([float(guess[0]),float(guess[1])],[checkIn[1],checkIn[2]])
        distCombinedError.append(dist)
        if dist < 0.010:
            exactFits += 1
    totalCheckInsEval = len(evalCheckIns)
    meanDist = mean(distCombinedError)*1000
    exactFitRatio = exactFits/totalCheckInsEval
    print "mean MFL",meanDist
    print "exactfits MFL",exactFitRatio
    return meanDist,exactFitRatio

def offCampus(allCheckIns):
    startTraining
    weekCounter = -1
    weeks = []
    for i in allCheckIns:
        if float(i[0]) > startTraining+7*24*60*60*(weekCounter+1):
            weeks.append([])
            weekCounter +=1
        elif float(i[0]) > startTraining:
            weeks[weekCounter].append(i)
    weekProbs = []
    for i in range(168):
        weekProbs.append(0)
    for week in weeks:
        #print "new week"
        weeklyCheckIns = []
        for i in range(168):
            weeklyCheckIns.append(0)
        for checkIn in week:
            weekHour = getWeekHour(float(checkIn[0]))
            weeklyCheckIns[weekHour] += 1
        for i in range(168):
            if weeklyCheckIns[i] > 0:
                #print weekProbs[i]
                weekProbs[i] += 1

    for i in range(168):
        weekProbs[i] = weekProbs[i]/weekCounter
    return weekProbs

def calculateRadiusOfGyration(coords):
    Xoncamp = []
```

```
    Yoncamp = []
    for checkIn in coords:
        Xoncamp.append(float(checkIn[1]))
        Yoncamp.append(float(checkIn[2]))

    n = len(Xoncamp)
    averageX = sum(Xoncamp)/n
    averageY = sum(Yoncamp)/n

    rgSum = 0
    for x,y in zip(Xoncamp,Yoncamp):
        dist = getDistance([x,y],[averageX,averageY])
        rgSum += dist**2
    rg = (rgSum/n)**(0.5)

    return rg


def calculatePhiMax(S, N):
    print S
    print N
    try:
        def funcPi(x):
            if x <= 0 or x >= 1:
                func = 5
            else:
                func = -(x*log2(x)+(1-x)*log2(1-x))+(1-x)*log2(N-1)
            return func

        xOpt = fsolve(lambda x : S-funcPi(x),0.8)
    except TypeError:
        print "something fucked up in the pimax calculation"
        xOpt = 0
    print xOpt
    return xOpt[0]


def calculatePredictability(timeAndPlaceTraining,uniqueCheckInCoords):
    timeseries = []
    for j in timeAndPlaceTraining:
        timeseries.append(j[1])

    N = len(uniqueCheckInCoords)
    n = len(timeAndPlaceTraining)

    knownSubseries = []
    sumOfSubseries = 0
    #print "time 6.1",str(time.clock()-base)
```

```
    for i in range(len(timeseries)):
        subseries = []
        subseries.append(timeseries[i])
        subseriesLength = 0
        #print "time 6.2",str(time.clock()-base)

        while subseries in knownSubseries and i+subseriesLength < n:
            subseriesLength += 1
            if i+subseriesLength < n:
                subseries.append(timeseries[i+subseriesLength])
        #print "time 6.3",str(time.clock()-base)

        knownSubseries.append(subseries)
        sumOfSubseries += len(subseries)
    #print "time 6.4",str(time.clock()-base)

    estimatedIndividualEntropy = 1/(1/n*sumOfSubseries)*math.log(n) #estiamted according to
equation xxx

    phiMax = calculatePhiMax(estimatedIndividualEntropy,N)
    print "entropy",estimatedIndividualEntropy
    print "phiMax",phiMax
    return estimatedIndividualEntropy,phiMax

#
def doIT():
    try:

meanDistCombined,meanDistPeriodic,meanDistSocial,meanAccesspointParallel,meanAccesspointGuessW
ighted,meanAccesspointGuess,exactFitCombinedRatio,exactFitPeriodicRatio,exactFitSocialRatio,onC
mpusCheckIns,actualOffCampusCounter,offCampusSuccessRatio,meanDistSocialDelta,meanCombinedWeigt
edDist,numberoflocations,socialCheckInCounter,pBest =
expectationMaximization(getCoordinates(timeAndPlaceTraining),getCoordinates(timeAndPlaceTraini
g_offcampus),i,getCoordinates(timeAndPlaceEval_offcampus),offCampusWeekProbs,uniqueCheckInCoord
)
    except IndexError:
        doIT()
    return
meanDistCombined,meanDistPeriodic,meanDistSocial,meanAccesspointParallel,meanAccesspointGuessW
ighted,meanAccesspointGuess,exactFitCombinedRatio,exactFitPeriodicRatio,exactFitSocialRatio,onC
mpusCheckIns,actualOffCampusCounter,offCampusSuccessRatio,meanDistSocialDelta,meanCombinedWeigt
edDist,numberoflocations,socialCheckInCounter,pBest

# choose individual that i going to be tracked
#device = "5ae2c6776f5b3d638e0770cf85ed6cdaaac727ac"

numberOfAccesspoints = 10
```

```
minimumVisitsToAccesspoint = 0.01
stackAccesspoints = True
delta = 0.00025
xx = np.arange(55.792, 55.778, -delta)
yy = np.arange(12.51, 12.53, delta)
X, Y = np.meshgrid(xx, yy)
startTraining = 1297033200
endTraining = 1301868000    #the training set runs from February 7, 2011 12:00:00 AM to April
3, 2011 11:59:59 AM
evalEnd = 1303077600 #1303077600    #the evaluation set runs from to April 4, 2011 12:00:00 AM
to April 17, 2011 11:59:59 AM


pathIn = '/Users/Raaby/Desktop/Master/DTUWiFi/trajectories_full_semester/'
pathIn2 = '/Users/Raaby/Desktop/Master/DTUWiFi/highActivity100/'
pathsIN =
[pathIn2]#['/Users/Raaby/Desktop/Master/DTUWiFi/prunedUsers10/','/Users/Raaby/Desktop/Master/D
UWiFi/highEntropy/','/Users/Raaby/Desktop/Master/DTUWiFi/randomHighActivity/']
pathADDONs = ["highActivity100"]#["prunedUsers10","highEntropy","randomHighActivity"]
for pathIn2,pathaddon in zip(pathsIN,pathADDONs):
    pathBase = os.path.join(os.getcwd(), "Data")
    pathOut = os.path.join(pathBase, pathaddon)
    pathOutMachine = os.path.join(pathBase, "machine")


    if not os.path.exists(pathBase):
        os.mkdir(pathBase)
    if not os.path.exists(pathOut):
        os.mkdir(pathOut)
    if not os.path.exists(pathOutMachine):
        os.mkdir(pathOutMachine)

    listing = os.listdir(pathIn2)
    listing2 = os.listdir(pathOutMachine)

    fileCounter = 0
    for device in listing:
        simulationActive = False
        tempFilename = "model_success_machine"+str(device)
        if device.startswith('.'):
            print "joker file filtered"
        elif tempFilename in listing2:
            fileCounter += 1
            print str(tempFilename)+" already created"
        else:#if fileCounter > 5:
            fileCounter += 1
            print "current file is: " + device +" "+str(fileCounter)+" of:"+str(len(listing))
            friends,friendScore,friendActualLocation,friendTraining = getFriends(device)
```

```
            if len(friends) > 0:
                filePathTrajectories = os.path.join(pathIn, str(device)[0:-27]+".txt")
                fileReaderTrajectories = csv.reader(open(filePathTrajectories, 'rb'),
delimiter=' ', quotechar='|')
                fileReaderTrajectories2 = csv.reader(open(filePathTrajectories, 'rb'),
delimiter=' ', quotechar='|')

                allCheckIns = []
                for row in fileReaderTrajectories2:
                    if float(row[0]) > startTraining and float(row[0]) < endTraining:
                        allCheckIns.append(row)
                offCampusWeekProbs = offCampus(allCheckIns)
        # temporal model
                print "calculating temporal distribution"
                place = []
                timeAndPlaceTraining = []
                timeAndPlaceTraining_offcampus = []
                timeAndPlaceEval = []
                timeAndPlaceEval_offcampus = []

                first = True
                for row in fileReaderTrajectories:
                    if first:
                        currentCheckIn = float(row[0])
                        first = False

                    nextCheckIn = float(row[0])
                    if nextCheckIn-currentCheckIn > 3*3600:
                        while nextCheckIn-currentCheckIn > 90*60:
                            if getDayHour(currentCheckIn) > 8 and getDayHour(currentCheckIn) <
20 and getWeekDay(currentCheckIn) < 6: # lower concetration of off_campus check-ins in the
regualar workhours
                                currentCheckIn = currentCheckIn + 90*60
                            else:
                                currentCheckIn = currentCheckIn + 30*60
                            row = (currentCheckIn,"off_campus")
                            if float(row[0]) > startTraining and float(row[0]) < endTraining:
                                place.append(row[1])
                                timeAndPlaceTraining_offcampus.append([row[0],row[1]])
                            if float(row[0]) > endTraining and float(row[0]) < evalEnd:
                                timeAndPlaceEval_offcampus.append([row[0],row[1]])
                        currentCheckIn = nextCheckIn

                    else:
                        if float(row[0]) > startTraining and float(row[0]) < endTraining:
                            place.append(row[1])
                            timeAndPlaceTraining.append([row[0],row[1]])
```

```
                        if float(row[0]) > endTraining and float(row[0]) < evalEnd:
                            timeAndPlaceEval.append([row[0],row[1]])
                            timeAndPlaceEval_offcampus.append([row[0],row[1]])
                        currentCheckIn = nextCheckIn

                if len(timeAndPlaceTraining) != 0 and len(timeAndPlaceEval) != 0:

                    checkInCoords = getCoordinates(timeAndPlaceTraining)
                    uniqueCheckInCoords = []
                    for checkIn in checkInCoords:
                        if [checkIn[1],checkIn[2]] not in uniqueCheckInCoords:
                            uniqueCheckInCoords.append([checkIn[1],checkIn[2]])

                    if len(uniqueCheckInCoords) > 3:
                        totalCheckIns = len(place)
                        uniqueAccesspointsTrajectories = f5(place)
                        sortedAccesspoints = []
                        for accesspoint in uniqueAccesspointsTrajectories:
                            sortedAccesspoints.append((accesspoint,place.count(accesspoint)))
                        sortedAccesspoints = sorted(sortedAccesspoints, key=lambda item:
item[1], reverse=True)


                        entropy,phiMax =
calculatePredictability(getCoordinates(timeAndPlaceTraining),uniqueCheckInCoords)

                        score = []
                        data = []
                        icounter = -1
                        for i in range(5,6):#Fiterlen(sortedAccesspoints)):
                            icounter += 1

meanDistCombined,meanDistPeriodic,meanDistSocial,meanAccesspointParallel,meanAccesspointGuessW
ighted,meanAccesspointGuess,exactFitCombinedRatio,exactFitPeriodicRatio,exactFitSocialRatio,onC
mpusCheckIns,actualOffCampusCounter,offCampusSuccessRatio,meanDistSocialDelta,meanCombinedWeigt
edDist,numberoflocations,socialCheckInCounter,pBest = doIT()

data.append([meanDistCombined,meanDistPeriodic,meanDistSocial,exactFitCombinedRatio,exactFitPe
iodicRatio,exactFitSocialRatio,onCampusCheckIns,actualOffCampusCounter,offCampusSuccessRatio,me
nDistSocialDelta,meanCombinedWeigthedDist,meanAccesspointParallel,meanAccesspointGuessWeighted,
eanAccesspointGuess])
                            if data[icounter][0] < data[icounter][1]:
                                meanDist = data[icounter][0]
                            else:
                                meanDist = data[icounter][1]
                            score.append(meanDist)

                        rg = calculateRadiusOfGyration(getCoordinates(timeAndPlaceTraining))
```

```
                        MFLmeanDist,MFLexactFitRatio =
MostFrequentedLocation(timeAndPlaceTraining,timeAndPlaceEval)


                        tempFilename = "model_success_machine"+str(device)
                        filename = os.path.join(pathOutMachine, tempFilename)
                        FILE = open(filename,"w")
                        FILE.writelines("Device name," + str(device))
                        FILE.writelines("\n")
                        FILE.writelines("total number of check-ins (training and evaluation
set),"+str(totalCheckIns))
                        FILE.writelines("\n")
                        FILE.writelines("check-ins (training
set),"+str(float(totalCheckIns)-float(data[0][6])-float(data[0][7])))
                        FILE.writelines("\n")
                        FILE.writelines("check-ins on/off campus (evaluation
set),"+str(data[0][6])+","+str(data[0][7]))
                        FILE.writelines("\n")
                        FILE.writelines("number of friends,"+str(len(friends)))
                        FILE.writelines("\n")
                        FILE.writelines("number of unique visited
locations,"+str(len(uniqueCheckInCoords)))
                        FILE.writelines("\n")
                        FILE.writelines("radius of gyration,"+str(rg))
                        FILE.writelines("\n")
                        FILE.writelines("calculated maximum predictability,"+str(phiMax))
                        FILE.writelines("\n")
                        FILE.writelines("distribution likellihood,"+str(pBest))
                        FILE.writelines("\n")
                        FILE.writelines("entropy,"+str(entropy))
                        FILE.writelines("\n")
                        FILE.writelines("MFL mean dist
error,"+str(MFLmeanDist)+","+str(MFLmeanDist/(rg*1000)))
                        FILE.writelines("\n")
                        FILE.writelines("MFL exact on campus fit ratio,"+str(MFLexactFitRatio))
                        FILE.writelines("\n")
                        icounter = 0
                        for i in data:
                            icounter += 1
                            FILE.writelines("number of used locations,"+str(numberoflocations))
                            FILE.writelines("\n")
                            FILE.writelines("percentage of offcampus check-ins successfully
guessed,"+str(i[8]))
                            FILE.writelines("\n")
                            FILE.writelines("average distance between guess and actual position
in meters
(meanDistCombined),"+str(float(i[0])*1000)+","+str(float(i[0])/rg)+","+str(float(i[11])*1000))
                            FILE.writelines("\n")
```

```
                            FILE.writelines("average distance between guess and actual position
in meters
(meanDistPeriodic),"+str(float(i[1])*1000)+","+str(float(i[1])/rg)+","+str(float(i[13])*1000))
                            FILE.writelines("\n")
                            FILE.writelines("average distance between guess and actual position
in meters (meanDistSocial),"+str(float(i[2])*1000)+","+str(float(i[2])/rg))
                            FILE.writelines("\n")
                            FILE.writelines("average distance between guess and actual position
in meters (meanDistSocialDelta),"+str(float(i[9])*1000)+","+str(float(i[9])/rg))
                            FILE.writelines("\n")
                            FILE.writelines("average distance between guess and actual position
in meters
(meanCombinedWeigthedDist),"+str(float(i[10])*1000)+","+str(float(i[10])/rg)+","+str(float(i[1
])*1000))
                            FILE.writelines("\n")
                            FILE.writelines("percentage of exact (less than 10 meters on
campus) guesses (exactFitCombinedRatio),"+str(i[3]))
                            FILE.writelines("\n")
                            FILE.writelines("percentage of exact (less than 10 meters on
campus) guesses (exactFitPeriodicRatio),"+str(i[4]))
                            FILE.writelines("\n")
                            FILE.writelines("percentage of exact (less than 10 meters on
campus) guesses (exactFitSocialRatio),"+str(i[5]))
                            FILE.writelines("\n")
                            FILE.writelines("number of social
check-ins,"+str(socialCheckInCounter))
                            FILE.writelines("\n")
                        FILE.close()


                        tempFilename = "model_success_"+str(device)
                        filename = os.path.join(pathOut, tempFilename)
                        FILE = open(filename,"w")
                        FILE.writelines("Device name: " + str(device))
                        FILE.writelines("\n")
                        FILE.writelines("total number of check-ins (training and evaluation
set): "+str(totalCheckIns))
                        FILE.writelines("\n")
                        FILE.writelines("check-ins (training set):
"+str(float(totalCheckIns)-float(data[0][6])-float(data[0][7])))
                        FILE.writelines("\n")
                        FILE.writelines("check-ins on/off campus (evaluation set):
"+str(data[0][6])+"/"+str(data[0][7]))
                        FILE.writelines("\n")
                        FILE.writelines("number of friends: "+str(len(friends)))
                        FILE.writelines("\n")
                        FILE.writelines("number of unique visited locations:
"+str(len(uniqueCheckInCoords)))
```

```
                        FILE.writelines("\n")
                        FILE.writelines("radius of gyration: "+str(rg))
                        FILE.writelines("\n")
                        FILE.writelines("MFL mean dist error:
"+str(MFLmeanDist)+"/"+str(MFLmeanDist/(rg*1000)))
                        FILE.writelines("\n")
                        FILE.writelines("MFL exact on campus fit ratio:
"+str(MFLexactFitRatio))
                        FILE.writelines("\n")
                        icounter = 0
                        for i in data:
                            icounter += 1
                            FILE.writelines("number of used locations:
"+str(numberoflocations))
                            FILE.writelines("\n")
                            FILE.writelines("percentage of offcampus check-ins successfully
guessed: "+str(i[8]))
                            FILE.writelines("\n")
                            FILE.writelines("average distance between guess and actual position
in meters (meanDistCombined):
"+str(float(i[0])*1000)+"/"+str(float(i[0])/rg)+"/"+str(float(i[11])*1000))
                            FILE.writelines("\n")
                            FILE.writelines("average distance between guess and actual position
in meters (meanDistPeriodic):
"+str(float(i[1])*1000)+"/"+str(float(i[1])/rg)+"/"+str(float(i[13])*1000))
                            FILE.writelines("\n")
                            FILE.writelines("average distance between guess and actual position
in meters (meanDistSocial): "+str(float(i[2])*1000)+"/"+str(float(i[2])/rg))
                            FILE.writelines("\n")
                            FILE.writelines("average distance between guess and actual position
in meters (meanDistSocialDelta): "+str(float(i[9])*1000)+"/"+str(float(i[9])/rg))
                            FILE.writelines("\n")
                            FILE.writelines("average distance between guess and actual position
in meters (meanCombinedWeigthedDist):
"+str(float(i[10])*1000)+"/"+str(float(i[10])/rg)+"/"+str(float(i[12])*1000))
                            FILE.writelines("\n")
                            FILE.writelines("percentage of exact (less than 10 meters on
campus) guesses (exactFitCombinedRatio): "+str(i[3]))
                            FILE.writelines("\n")
                            FILE.writelines("percentage of exact (less than 10 meters on
campus) guesses (exactFitPeriodicRatio): "+str(i[4]))
                            FILE.writelines("\n")
                            FILE.writelines("percentage of exact (less than 10 meters on
campus) guesses (exactFitSocialRatio): "+str(i[5]))
                            FILE.writelines("\n")
                        FILE.close()
                    else:
                        print device, "has less than 4 unique check-in positions"
```

```
                else:
                    print "no on campus check-ins in evaluation period"
            else:
                print device,"has no friends"
```