Aalborg University
Department of Computer Science
Selma Lagerlöfs Vej 300, 2.2.16
DK-9220 Aalborg Ø
http://www.cs.aau.dk/

Titel:
Afprøvning af näive bayesiansk netværk til estimering af filmbedømmelser i movieLens datasettet

Fag:
Beslutningsstøtte og maskinindlæring

Projekt enhed:
EVU, forår 2009

Studerende:
Niels Klitgaard Lassen

Vejleder:
Thomas D. Nielsen

Antal printede eksemplarer:
3

Antal sider:
22 + 1CD

Afleveringsdato:
10. juni 2009

Niels Klitgaard Lassen

Synopsis:

Denne rapport beskriver collaborativ filtrering vha en hukommelsesbaseret metode contra en modelbaseret metode. Formålet har været via et praktisk forsøg at finde ud af, om den modelbaserede metode kunne overgå den hukommelsesbaserede metode. Teorierne for begge metoder er kort præsenteret og der er lavet en gennemgang af det overordnede programdesign af den egenudviklede kode. Resultaterne af programafviklingerne peger på den hukommelsesbaserede metode som den mest præcise, mens den modelbaserede metode som forventet langt overgår den hukommelsesbaserede metode, når det kommer til performance.

Forord

Denne rapport er udarbejdet af Niels Lassen som en del af kurset ”Beslutningsstøtte og maskinindlæring”, der blev udbudt af Institut for Datalogi, Aalborg universitet, i efteråret 2006 under uddannelsen ”Master i Softwarekonstruktion”. Rapporten handler om collaborativ filtrering, hvor der laves undersøgelser af to hukommelsesbaserede metoders samt en modelbaseret metodes (modelleret som et bayesiansk netværk) egnethed til at estimere brugeres bedømmelser af film. Rapporten dokumenterer dels de anvendte teorier samt den udviklede java-kode, hvori der bl.a. integreres til Hugin Expert 7.0. Integrationen fra egenudviklet kode til Hugin Expert 7.0 foretages via et API til Hugin Expert 7.0, og dette API vil i det følgende blive benævnt ”Hugin API”. Hugin Expert 7.0 har også en grafisk brugergrænseflade, som i det følgende benævnes ”Hugin GUI”.

Tak til Thomas D. Nielsen for hans villighed til at vejlede i projektet på trods af det forskudte kursus- og projektforløb samt for hans hjælp til at forstå Hugins virkemåde.

Indholdsfortegnelse

Indledning

Mængden af information om alt muligt vokser eksplosivt i disse år, og det kan være meget svært at holde sig opdateret inden for alle områder. Af den grund vil enhver form for hjælp til at tage beslutninger være kærkomment for mange mennesker – også selv om beslutningerne kun vedrører mindre svære dagligdags beslutninger. Det er min opfattelse at almindelige mennesker vil kunne gøre bedre brug af hinandens erfaringer ved valg af mange forskellige produkter, hvis der var en god beregningsmæssig understøttelse til at træffe de rigtige valg. Derfor er recommender systemer og collaborativ filtrering interessante emner at behandle i et projektforløb.

Der er i litteraturen beskrevet flere forskellige modeller til at estimere nye brugeres bedømmelser af produkter. Modellerne kan splittes op i to hovedgrupper: Implicitte eller eksplicitte modeller. De eksplicitte modeller forudsætter, brugerne aktivt har afgivet en bedømmelse af de enkelte produkter, mens behandlingen i de implicitte modeller udelukkende går på, om brugerene har valgt (eller implicit fravalgt) et produkt. Implicitte modeller anvendes fx på internetsites, hvor man ud fra tidligere sidevisninger kan forudsige noget om brugernes præferencer for andre sider.
I movieLensdatasettet som anvendes i denne opgave har brugerne netop afgivet bedømmelser af filmene på en skala fra 1-5 og derfor er det naturligt at anvende de eksplicitte modeller til at forsøge at estimere deres bedømmelser af andre film.

MovieLens dataene

Dataene fra MovieLens ligger i filer, og kan downloades fra GroupLens hjemmeside (http://www.grouplens.org/). Filerne, der er anvendt i dette projekt, kan kort beskrives ved følgende:

- u.data: Fil med brugernes bedømmelser af filmene. Hver bruger (i alt 943 brugere) har bedømt mindst 20 film, og det samlede antal bedømmelser er 100000.
- u.item: Fil der beskriver filmene (i alt 1682 fil) med id, titel, datoer, genre... I dette projekt anvendes udelukkende id, men alle data er indlæst og der instantieres objekter med samtlige attributter.
- u.user: Fil med demografiske informationer om brugerne. I dette projekt anvendes udelukkende brugernes id, men alle data er indlæst og der bliver instantieret komplette objekter for alle bruger.
- u1.base. u2.base, u3.base, u4.base, u5.base: Fil med 80% af dataene fra u.data. Anvendes til læring.
- u1.test, u2.test, u3.test , u4.test , u5.test: Fil med de resterende 20% af data fra u.data. Anvendes til test af model indlært ved brug af den tilsvarende base-fil.

u1-u5 base-fil og test-fil er beregnet til at lave 5 uafhængige lærings- og testkørsler, hvorfra der kan laves krydssammenligninger af de genererede data og dannes data til en afsluttende sammenligning af den modelbaserede kontra hukommelsesbaserede metodes egnethed til at estimere brugerbedømmelser af filmene.
Det skal her bemærkes, at jeg på et sent tidspunkt i projektforløbet har fundet ud af, at Hugin API´et ikke fungerer korrekt, når der arbejdes på det samlede antal film i movieLens datasettet, og jeg har derfor valgt kun at medtage de første 999 film i mine kørsler. Når antallet af film begrænses til de første 999 film, viser det sig, at der er film som kun en bruger har bedømt, og der opstår givetvis andre specielle sideeffekter. Der har desværre ikke været tid til at lave en nærmere analyse af de data der ligger til grund for resultaterne i rapporten.

Problemformulering.

Denne rapport omhandler ”Collaborativ filtering”, der er et begreb, der dækker over en række metoder, hvormed man forsøger at estimere brugeres bedømmelse af nogle produkter ud fra kendskab til, hvordan de har bedømt andre produkter og kendskab til hvordan andre brugere har bedømt produkterne. I rapporten bliver der ud fra et dataset kaldet movielens 100k[1] arbejdet med en hukommelsesbaseret model, som tager udgangspunkt i dels vektoralgebra dels korrelationsanalyse, og med en modelbaseret model, som tager udgangspunkt i et naïve bayesiansk netværk. De to modellers evne til at estimere brugernes bedømmelse af filmene bliver sammenlignet, og rapporten skal ende ud i en konklusion over, om en modelbaseret model kan give ligeså præcise bedømmelsesestimater som den modelbaserede metode.

1 Der anvendes et dataset fra http://www.grouplens.org/node/73 med 100000 bedømmelses fra 943 bruger på 1682 forskellige film.

Collaborativ filtrering.

Den hukommelsesbaserede metode.

Den hukommelsesbaserede model, der anvendes i dette projekt, baseres dels på en metode med udgangspunkt i vektorregning, hvor hver enkelt brugers bedømmelse af filmene lægges ind i en vektor, hvis komponenter antager bedømmelsesværdierne 1-5 (0 svarer til ikke bedømt) dels på en metode, der har sammenfald med correlationsanalyse. Begge metoder er præsenteret i materialet fra movieLens [Collab] og gør brug af de samme principper for beregning af middelværdi af filmbedømmelser og beregning af estimerede filmbedømmelser. Derfor bliver disse to principper først præsenteret, hvorefter de øvrige elementer i metoderne præsenteres.

Middelværdien af brugers *i´*s bedømmelser kan udtrykkes som:

$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$$

$v_{i,j}$: bruger *i´*s bedømmelse af film *j*.
I_i: Mængden af film som bruger *i* har bedømt.

En aktuel brugers estimerede bedømmelse, $p_{a,j}$, af film *j* kan udtrykkes ved en vægtet sum af tidligere brugeres bedømmelser:

$$p_{a,j} = \bar{v}_a + k \sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i)$$

n: Antallet af andre brugere, der har angivet bedømmelser på film j.
k: Normaliseringsfaktor som sikrer, at de vægtede summe summerer til 1.
w(i,a): Vægtfaktor udregnet på baggrund af bedømmelser af film som begge brugere har set.

Vægtfaktoren, *w(a,i)*, kan opstille på flere forskellige måder og her præsenteres en metode tager udgangspunkt i vektorregningen og en metode der tager udgangspunkt i korrelationsanalyse.

Vægtfaktorberegning vha elementer fra vektorregning

Vægtfaktoren, *w(a,i)*, reducerer med denne metode den aktuelle brugers vektors længde med en størrelse, der svarer til cosinus af vinklen mellem brugerens bedømmelsesvektor og den vektor, han aktuelt bliver sammenlignet med.

$$w(a,i) = \sum_j \left(\frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \quad \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}} \right)$$

Det tilsvarende udtryk vil med en normal vektornotation blive udtrykt på følgende vis i vektoralgebraen[Calc]

$$\cos\Theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

- hvor Θ er vinklen mellem de to vektorer.

Vægtfaktorberegning vha elementer fra korrelationsanalyse

I metoden, der tager udgangspunkt i korrelationsanalyse, betragter man vægten som korrelationskoefficienten mellem den aktuelle brugers bedømmelser og øvrige

brugeres bedømmelser[2]. Dette resulterer i, at vægtfaktoren, *w(a,i)*, kan udtrykkes ved

$$w(a,i)=\frac{\sum_j (v_{a,j}-\bar{v}_a)(v_{i,j}-\bar{v}_i)}{\sqrt{\sum_j (v_{a,j}-\bar{v}_a)^2 \sum_j (v_{i,k}-\bar{v}_i)^2}}$$

Den modelbaserede metode

Den modelbaserede metode baseres på, at tidligere brugeres bedømmelser af filmene kan anvendes til at estimere, hvordan den aktuelle bruger, *a,* vil bedømme en film, *j*, givet de bedømmelser, han allerede har oplyst om andre film.

Dette gøres i denne rapport ved at implementere et näive bayesiansk netværk, hvor en clusternode med et antal tilstande, s, knyttes til underliggende filmnoder, som alle har 5 tilstande svarende til de bedømmelser brugerne kan give til filmene[3] – se Illustration 1.

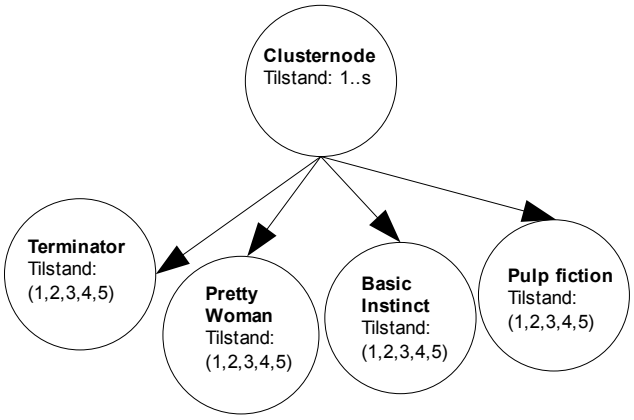

Illustration 1: Princippet i det bayesianske netværk der opbygges under programafviklingen.

Fordelen ved det näive bayesianske netværk er, at man med det antager uafhængighed mellem filmnoderne givet et kendskab til clusternodens tilstand. Dette er en meget vigtig egenskab ved det näive bayesianske netværk, idet man under betingelse af, at filmnoderne er uafhængige af hinanden, begrænser omfanget af beregninger, der skal udføres.

Clusternodens sandsynlighedsfordeling givet filmnoderne, f_1, ... f_n vil således kunne beregnes med udtrykket [BNDG]:

2 Korrelationskoefficienten anvendes i andre sammenhænge i forbindelse med at finde sammenhænge mellem data. Korrelationskoefficienten kan anvendes til at finde den bedst mulige estimerede rette linie, der beskriver sammenhængen mellem et sæt variabler.

3 Under programafviklingen anvendes der en simplere navngivning af filmnoderne, idet de hver især bliver navngivet "ID"+"itemnr", hvor itemnr er det id, der er knyttet til filmene i dataene fra movieLens.

$$P(clusterNode|f_1,...f_n) = \frac{P(clusterNode)P(f_1,...f_n|clusterNode)}{P(f_1,...f_n)}$$

$$= \frac{P(H)\prod_{i=1}^{n} P(f_i|H)}{P(f_1,...f_n)}$$

, hvor nævneren er en normaliseringskonstant.

Clusternoden og dens stateværdier er her ukendte størrelser og behandles i EM-algoritmen som MCAR **M**issing **C**ompletely **A**t **R**andom. MCAR bruges som begreb om manglende statedata, hvorom det gælder, at der ikke er en speciel grund til data mangler. Dette er præcist tilfældet for statedata på clusternoden. MAR - "**M**issing **a**t **R**andom" er tilsvarende et begreb, der dækker over, at data mangler, fordi der er en afhængighed til andre variabler i det bayesianske netværk. Dette kunne være tilfældet imellem filmnoderne, idet en bruger måske efter at have set en film med en bestemt skuespiller som hovedrolleindehaver har fravalgt alle øvrige film, der har denne skuespiller som hovedrolleindehaver. Film, der hverken er MCAR eller MAR, skal helt sorteres ud af datasættet, fordi deres fravær skyldes brugernes måde at svare på. Da brugerne har skullet opgive demografiske data, kan det ikke udelukkes, at nogle brugere har fravalgt at give bedømmelser af film, de synes, det er pinligt, at de har set disse film og de er bange for at blive genkendt i undersøgelsen. De demografiske data anvendes i øvrigt ikke i opsætningen af det näive bayesianske netværk.

Sandsynlighedsfordelingerne i den modelbaserede metode estimeres vha. EM-algoritmen[4]. EM-algoritmen fungerer på den måde, at man ud fra initial-sandsynlighederne og læringsdata (som ikke er komplette) beregner sig frem til forventede værdier for de manglende læringsdata. Ud fra de nye estimerede data laver man en optælling af de forventede værdier og casedata og estimerer nu herudfra en ny sandsynlighedsfordeling. Og så fremdeles indtil algoritmen møder sit stopkriterie/konvergens.

Delkonklusion

Afsnittet præsenterede kort teorierne bag den anvendte hukommelsesbaserede metode og teorierne bag den modelbaserede metode.

4 EM: Expectation og Maximization.

Eksperimentafsnit

Der er i forbindelse med projektarbejdet kodet en del mhp at lade en programafvikling understøtte de præsenterede teorier. Koden er udført i java, og det samlede omfang er på 11-1200 linier fordelt på ca. 15 klasser, samt 5-600 linier kode til unit-tests af i alt 8 klasser. Nedenfor vil datamodellen for problemområdet blive præsenteret i et klassediagram, og selve programafviklingen i forbindelse med bygning af det bayesianske netværk, generering af case-fil, indlæsning af evidens samt udtræk af estimerede bedømmelser (beliefs) vil blive præsenteret med sekvensdiagrammer. Alt er holdt på et overordnet niveau, og koden er vedlagt på CD. Både den modelbaserede og den hukommelsesbaserede metode forudsætter indlæsning af bedømmelsesdata fra base-filen, og programmet vil ud fra disse data have et grundlag for at beregne estimerede bedømmelser på node-states angivet i testfilen. Bedømmelsesdata fra test-filen bliver ikke opsamlet og anvendt som læringsdata under kørslen.

Klassediagram for datamodel i problemområdet

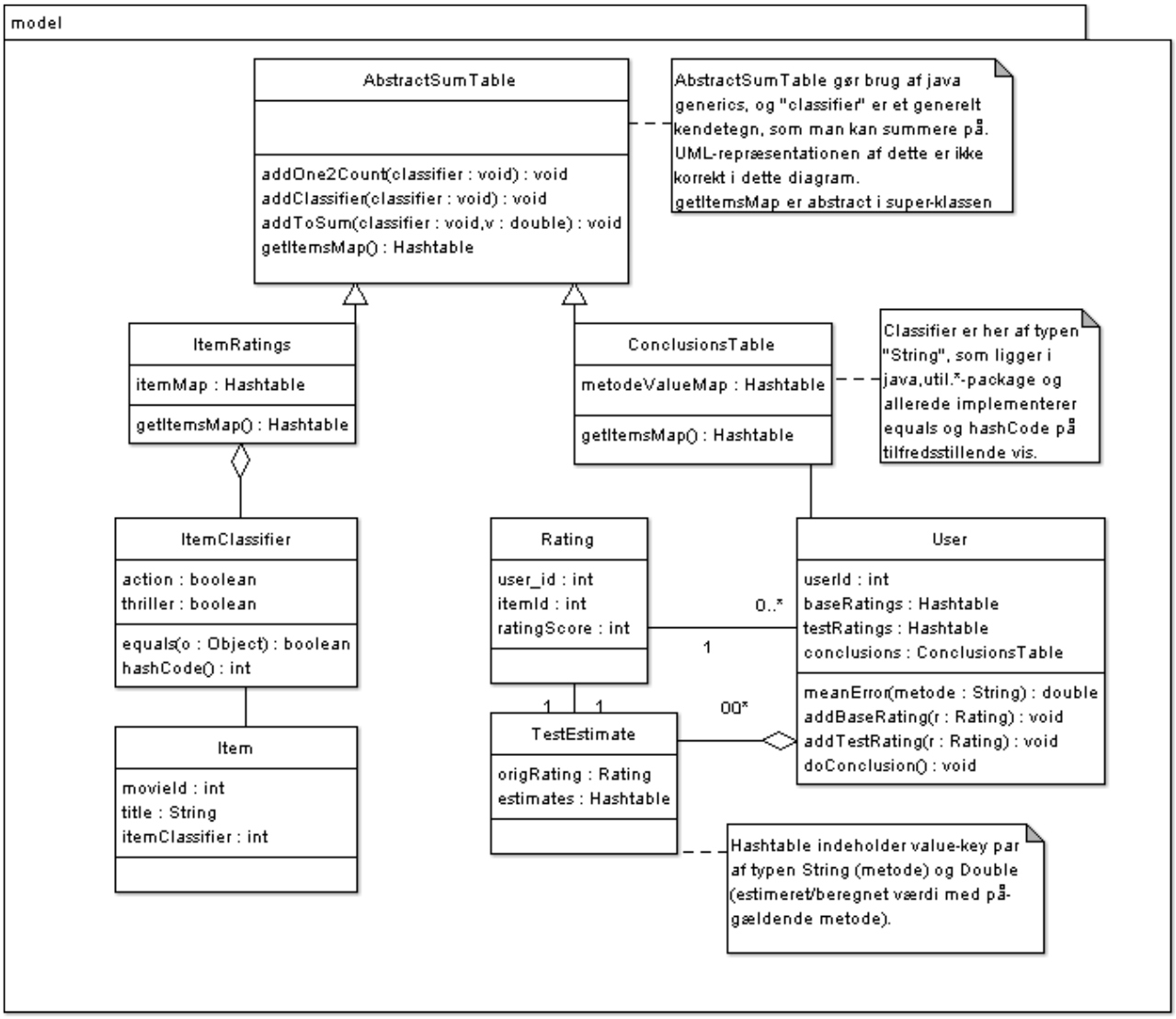

Illustration 2: Klassediagram, der viser modellering af de væsentligste klasser i problemområdet. Item-klassen er ikke direkte associeret til Rating-klassen, idet der ikke er behov for kendskab til Item-klassens attributter for at lave estimater på filmbedømmelser (I et "renere" oo-design ville Rating-klassen ikke have attributterne user_id og itemid men derimod associationer til klasserne i stedet). Item-klassen indlæses alligevel og anvendes bl.a. i forbindelse med generering af bayesiansk netværk og generering af case-data til EM-læringsalgoritmen.

Under programafviklingen opbygges der et näive bayesiansk netværk ved at oprette en clusternode og ellers "mappe" hver enkelt item-objekt over i en filmnode med 5 states svarende til de 5 bedømmelser, en bruger kan knytte til en film. Herefter genereres der casedata til EM-algoritmen ved for hver enkelt bruger at oprette en record i en fil, hvoraf det fremgår, om han har bedømt den enkelte film. Casedata loades og EM algoritmen afvikles. Det samlede forløb ses af Illustration 3 og i javakoden i metoderne `model.NBN.buildNetwork()`, `model.NBN.generateDataFile()`, `model.NBN.loadCases()`.

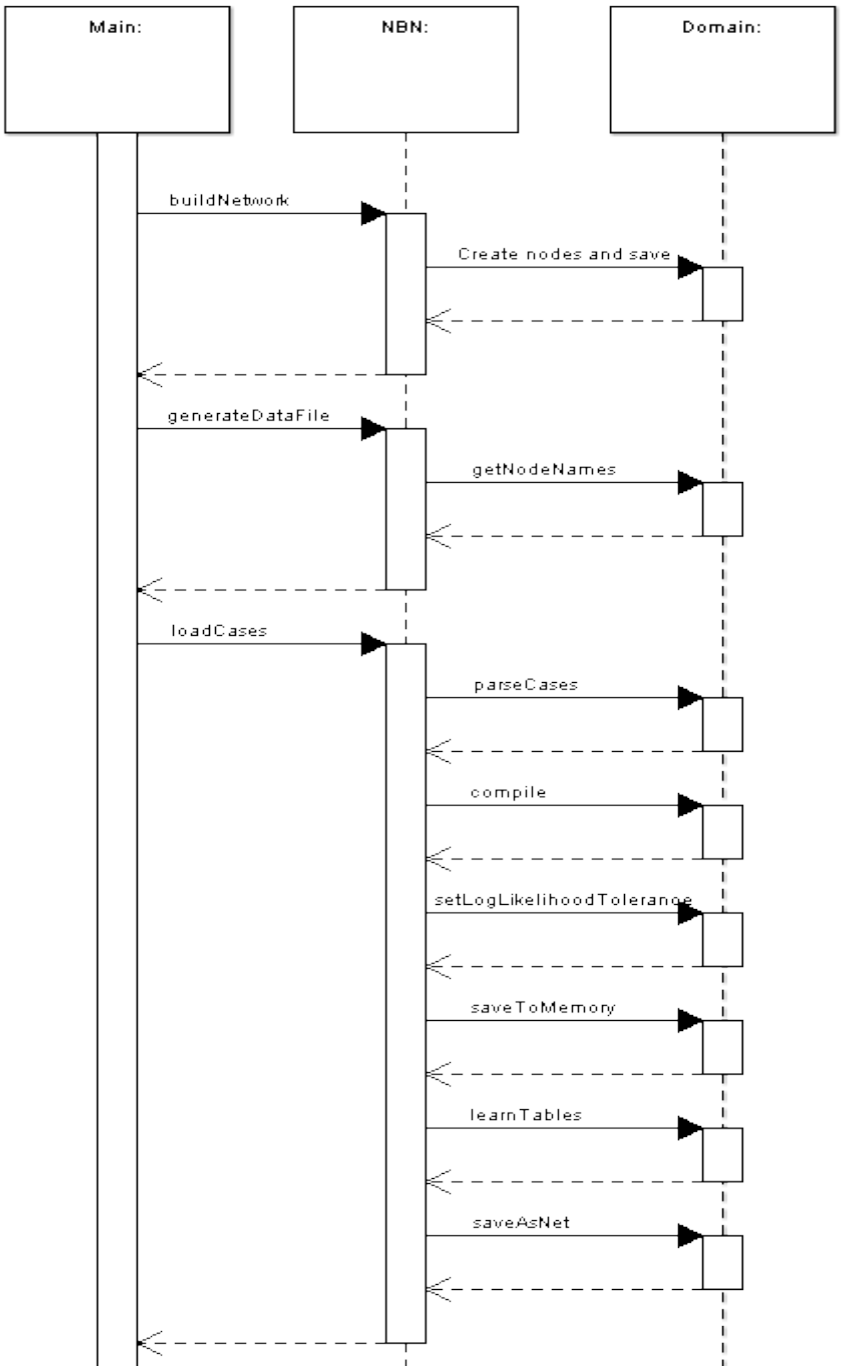

Illustration 3: viser i grove træk forløbet med at bygge domaine, generere casedata, loade casedata, udføre EM-læring samt gemme resultat af læring i fil. Processen starter i main, der først kalder buildNetwork på klassen NBN...

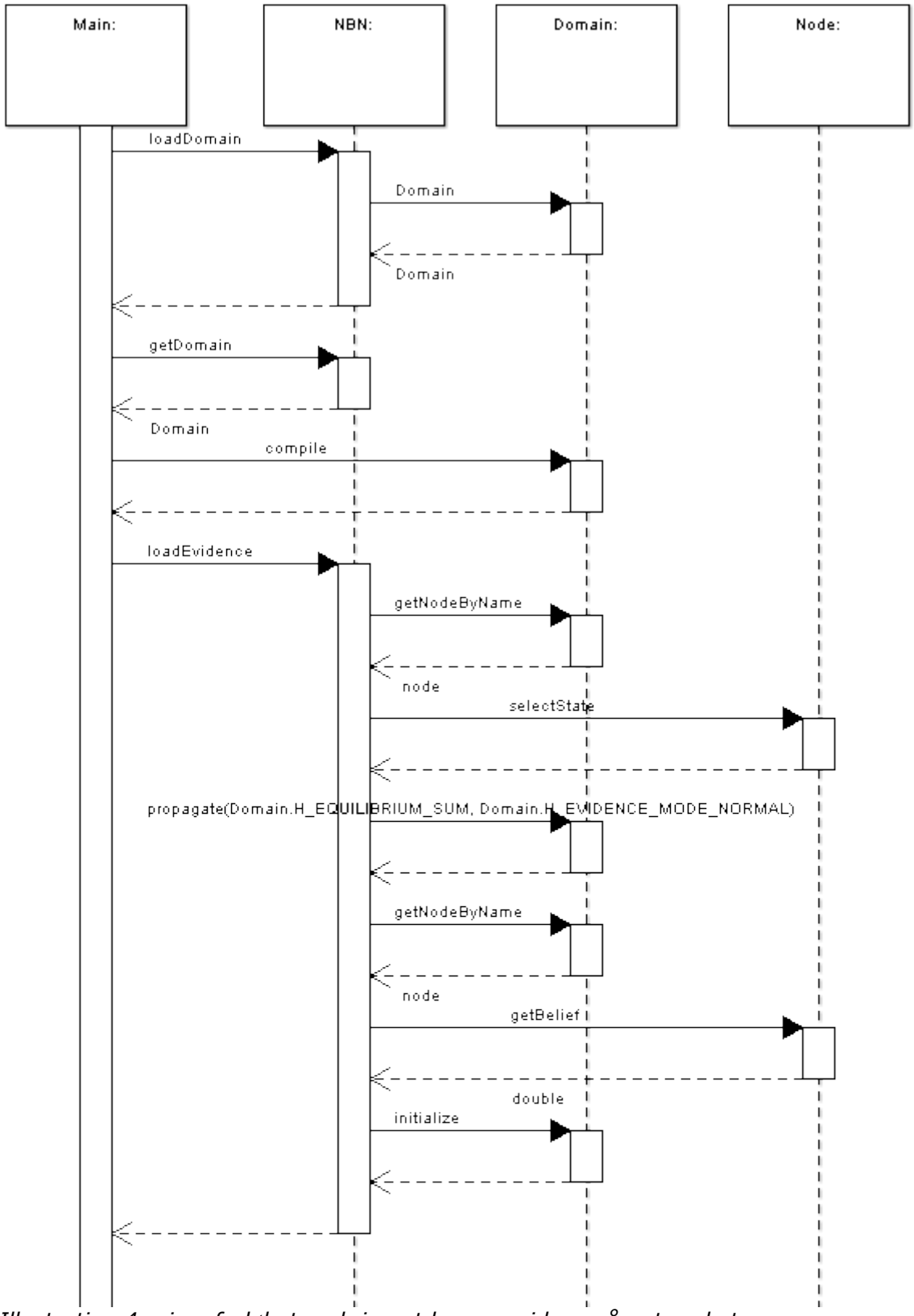

Illustration 4: viser forløbet omkring at lægge evidens på netværket, propagere og udtrække beregnede sandsynligheder.

Herefter er modellen klar til at loade evidens og estimere filmbedømmelser – se Illustration 4. Der er to løkker i metoden loadEvidence på klassen NBN. I første løkke itereres der over brugerens angivne bedømmelser fundet i base-filen – se side 4 - og efter at have propageret evidens, loopes der igennem de data, som der skal estimeres

for brugeren – disse findes i test-filen se side 4 - og de estimerede stateværdier trækkes fra netværket. Koden findes i javakoden under `model.NBN.loadEvidence()`.

Sammenligning af metodernes estimerede filmbedømmelser

Den gennemsnitlige fejl af forventede bedømmelser, m_a, for bruger, *a,* kan udtrykkes ved

$$\bar{S}_a = \frac{1}{m_a} \sum_{j \in P_a} |p_{a,j} - v_{a,j}|$$

hvor:
m_a: antallet af estimerede bedømmelser for bruger, *a.*
$v_{a,j}$: bruger *a´*s virkelige bedømmelse af film *j* fundet ud fra bedømmelsesdata i testfilen (se beskrivelse af testfil på side 4).
$p_{a,j}$: bruger *a´*s forventede bedømmelse af film *j* beregnet ud fra bedømmelsesdata i basefilen (se beskrivelse af basefil på side 4).

De to hukommelsesbaserede metoder og den modelbaserede metode kan bedømmes i forhold til hinanden ved at sammenligne middelværdien af "gennemsnitlige fejl af forventede bedømmelser" for hver enkelt bruger der har afgivet bedømmelser, som bliver testet.

$$S_{metode} = \frac{\sum_{a \in Brugere} \bar{S}_a}{antal\ brugere}$$

Gennemsnitlige fejl af forventede bedømmelser for den hukommelsesbaserede metode.

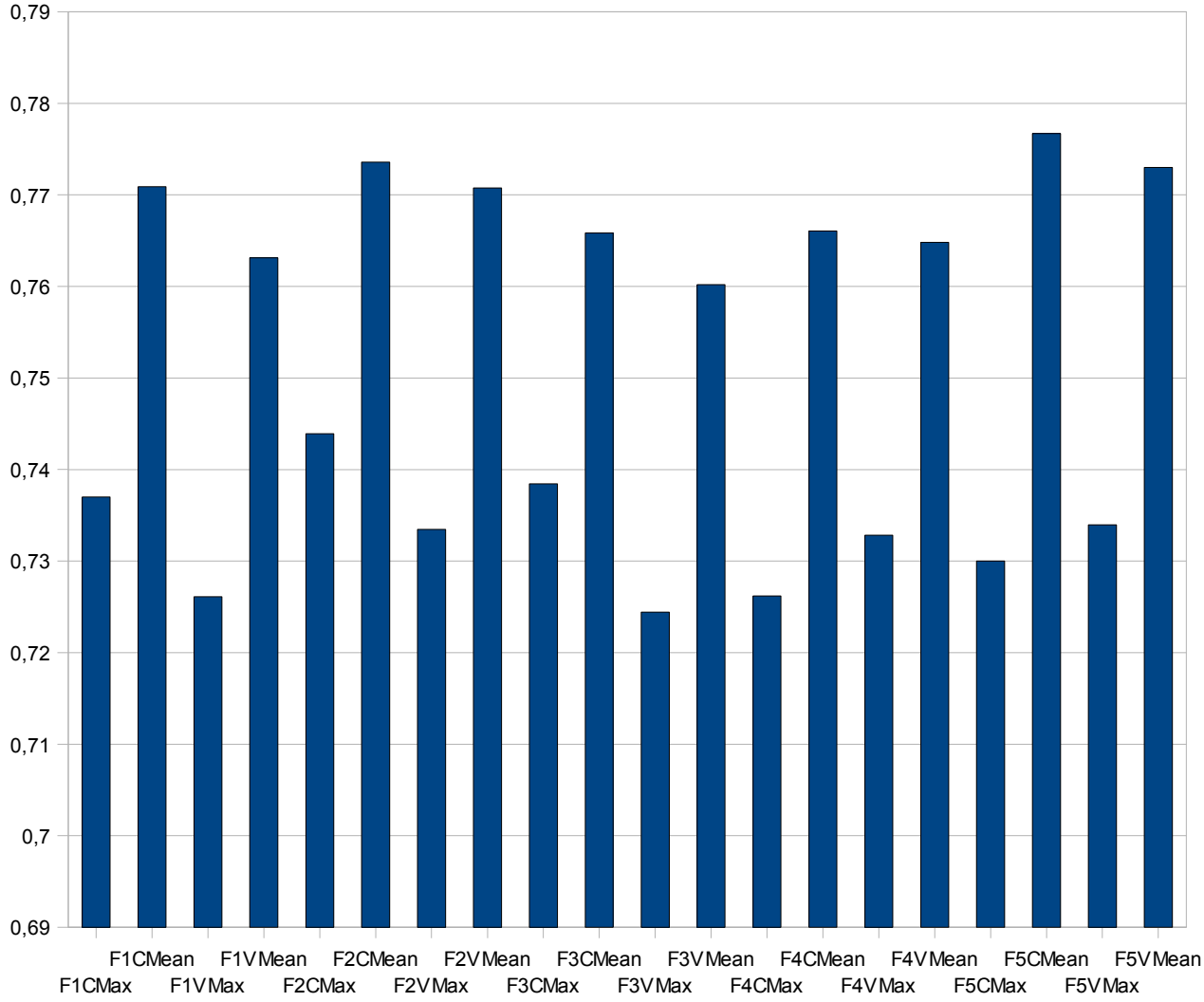

Illustration 5: Gennemsnitlige fejl af forventede bedømmelser for den hukommelsesbaserede metode. X-aksens værdier henviser til dels anvendte filsæt (F1: t1.base og t1.test, F2: t2.base og t2.test ...) dels til metode anvendt til estimering til af bedømmelse af den enkelte film (Mean/Max).

I den hukommelsesbaserede metode bliver den gennemsnitlige fejl af forventede bedømmelser for en bruger og for hele datasættet udregnet på to forskellige måder benævnt henholdsvis ”Mean” og ”Max”.

Den gennemsnitlige fejl af forventede bedømmelser benævnt med ”Mean” i Illustration 5 er udregnet på grundlag af estimerede bedømmelser udregnet på baggrund af formlen angivet på side 5.

Den gennemsnitlige fejl på forventede bedømmelser benævnt med ”Max” i Illustration 5 er udregnet på grundlag af ”Mean”-værdier afrundet til nærmeste heltal.

Det ses af Illustration 5, at gennemsnitlige fejl af ”Max”-typen konsekvent er lavere end ”Mean”-gennemsnitsfejlene, hvilket må skyldes, at den estimerede bedømmelse er et godt bud på den faktiske bedømmelse, og at afvigelsen i forhold til den faktiske

bedømmelse derfor oftest mindre ved at afrunde til nærmeste heltal.

Gennemsnitlige fejl af forventede bedømmelser udregnet med den modelbaserede metode

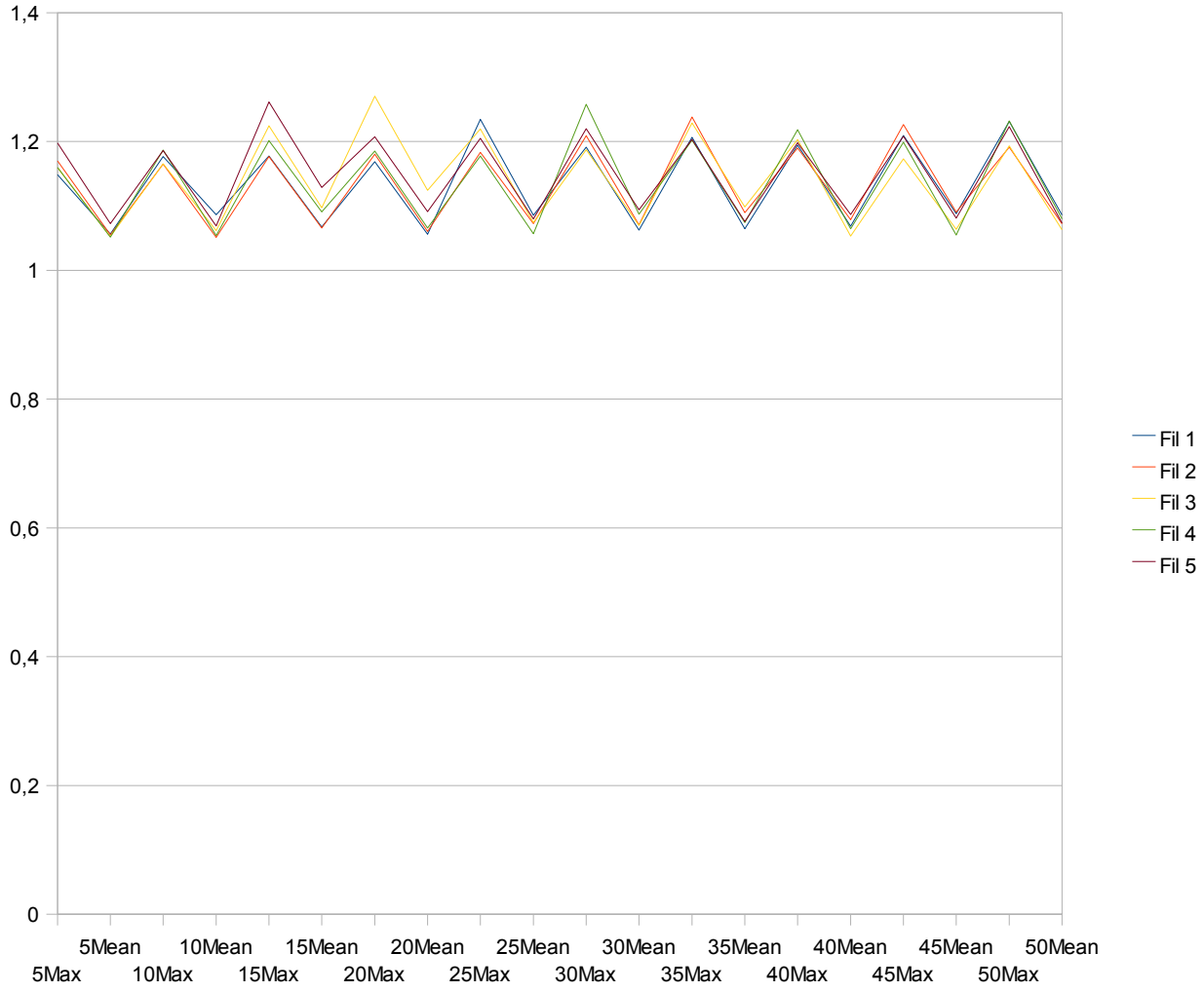

Illustration 6: Gennemsnitlig fejl på forventede bedømmelser beregnet med modelbaseret metode. Beregninger er kørt med 5 filsæt (Fil 1: t1.base og t1.test, Fil 2: t2.base og t2.test...) og for hvert filsæt er der lavet beregninger på flere forskellige näive bayesianske netværk, hvor clusternoden har haft 5, 10, 15 ... 50 states. De gennemsnitlige fejlværdier er fundet som afvigelse fra faktisk værdi i forhold til henholdsvis middelværdi (Mean) af estimeret bedømmelse på filmen henholdsvis estimeret bedømmelse af filmen med størst sandsynlighed (Max).

Middelværdi, μ, (benævnt "Mean" i Illustration 6) af bedømmelse på filmen, F, med states, s, 1..5 beregnes ved

$$\mu(s)=\sum_{s=1}^{5} F(s)P(s|e)$$

, hvor *P(s|e)* er sandsynligheden for de enkelte states givet den evidens, e, der er lagt

ind (evidens vil her være brugerens bedømmelser af andre film, han har set).

Beregningen foretages i javakoden i `model.NDN.getMeanBeliefRate()`.

Den forventede bedømmelse (tilstand/state) af en film, der har højest sandsynlighed, givet evidens benævnes "Max" i Illustration 6, og værdien findes ved at sweepe igennem nodens states, og opsamle den stateværdi med størst sandsynlighed for at forekomme. Beregningen foretages i javakoden i metoden `model.NDN.GetMaxBeliefRate()`.

Der er flere mærkelige aspekter i resultaterne fra den modelbaserede analyse:

1. Den gennemsnitlige fejl af bedømmelser er tilsyneladende upåvirket af antal states på clusternoden.
2. Gennemsnitsfejl af typen "Mean" ligger konsekvent under gennemsnitsfejl af typen "Max".
3. Der er kun små udsving i data for de afviklede kørsler.

Ad 1)
Programmet logger også information om BIC-score[5] (**B**ayesian **I**nformation **C**riterion) og kørslen på filsæt nr. 1 gav følgende værdier:

5 states på clusternode => BIC: -153909.02532523975
10 states på clusternode => BIC: -215547.93957584433
15 states på clusternode => BIC: -280950.73388108326
20 states på clusternode => BIC: -345055.027922402
25 states på clusternode => BIC: -412849.07671807444
30 states på clusternode => BIC: -481701.04911878996
35 states på clusternode => BIC: -545376.4245103493
40 states på clusternode => BIC: -613211.406534896
45 states på clusternode => BIC: -680423.1902039265
50 states på clusternode => BIC: -746105.238963702

Ud fra dette kan der ikke argumenteres for at arbejde med mange states (ihvertfald ikke med mere end 10 med det givne antal cases til læring, idet BIC i princippet kunne have et maksimum, når der er 6, 7, 8 eller 9 states på clusternoden) da forbedringen af de estimerede sandsynlighedsfordelinger ikke kan opveje omkostningen ved at arbejde med den mere komplekse model. Dette resultat er overraskende, da filmene er karakteriseret med ét eller flere klassifikationer inden for 18 forskellige kategorier – drama, aktion, thriller, western... Denne klassificering anvendes ikke i det bayesianske netværk, men det virker mærkeligt, at en clusternode med forholdsvist få tilstande giver den største BIC-værdi.
Ser man i øvrigt på BIC-score på tværs af de 5 filsæt, får man fx.

Filsæt 1 clusternode med 20 states => BIC: -345055.027922402
Filsæt 2 clusternode med 20 states => BIC: -346490.52312145283
Filsæt 3 clusternode med 20 states => BIC: -345083.5947025612
Filsæt 4 clusternode med 20 states => BIC: -346540.98858327954
Filsæt 5 clusternode med 20 states => BIC: -344479.65143415856

Heraf vil man konkludere, at de 5 filsæts sandsynlighedsfordelinger, der er indlært af EM-algoritmen i Hugin API, skal rangeres på følgende vis (bedst først):

1. sandsynlighedsfordeling til filsæt 5
2. sandsynlighedsfordeling til filsæt 1
3. sandsynlighedsfordeling til filsæt 3
4. sandsynlighedsfordeling til filsæt 2
5. sandsynlighedsfordeling til filsæt 4

5 BIC-score: **B**ayesian **I**nformation **C**riterion – anvendes til at karakterisere bayesianske netværk, når de sammenlignes med hinanden. Ved modellering af samme univers bør det bayesianske netværk som har højest BIC-score foretrækkes frem for de øvrige.

Den gennemsnitlige fejl for estimerede filmbedømmelser for de 5 filsæt er mindst for filsæt 1, herefter filsæt 2, 4, 5 og 3 – se Illustration 7 - hvilket ikke understøtter valget af sandsynlighedsfordeling til filsæt 5 som det bedste.

	Fil 1	Fil 2	Fil 3	Fil 4	Fil 5
Max	1,17	1,18	1,27	1,19	1,21
Mean	1,06	1,06	1,12	1,07	1,09

Illustration 7:"Max"- og "Mean"-gennemsnitlig fejl af estimeret filmbedømmelse for näive bayesiansk netværk med 20 tilstande på clusternode.

Ad 2)
Jeg havde forventet, at gennemsnitsfejl af typen ”Mean” var et godt bud, men at gennemsnitsfejl af typen ”Max” sikkert ville være bedre. ”Mean” fejlgennemsnittet er beregnet på grundlag af de enkelte films beregnede gennemsnitsbedømmelser (givet brugerens øvrige filmbedømmelser), mens ”Max”-fejlgennemsnittet er udregnet på grundlag af den mest sandsynlige estimerede bedømmelse af den enkelte film (givet brugerens øvrige filmbedømmelser). Ud fra dette var forventningen, at de to værdier lå tæt på hinanden, og at ”Max”-fejlgennemsnittet ville være lavest, idet tallene som lå til grund for dette var uden decimaler og af den grund måske var tættere på den faktiske værdi. Men det er ikke tilfældet.
Det ville have været interessant at se om en afrunding af tallene, der ligger til grund for ”Mean”-fejlgennemsnittet, ville få dette tal gjort endnu pænere svarende til metoden, der er anvendt på den hukommelsesbaserede metode.
Ad 3)
Som nævnt under punkt 1 og 2 er der god grund til at være mistænkelig overfor resultaterne, og ser man i logfiler og outputfiler fra kørslerne (findes på vedlagte CD under direktoriet movielens\movielensjava\results) finder man flere mærkelige aspekter af kørslerne. Der logges ingen fejl. Sandsynlighedsfordelingerne i NET-filerne ser ud til at indeholde fornuftige værdier. EM-læringen ser ud til at være korrekt gennemført – Hugin API´ets egen log (hedder ”filnavn”.log) viser faldende log-likelihood værdier for hver iteration af EM-algoritmen. Alt ser ud til at være afviklet korrekt, og alligevel er der næsten ingen effekt at spore i resultaterne fra kørslen. Jeg har forsøgt at stoppe en kørsel efter filen med det näive bayesianske netværk (NET-filen) og filen med casedata til EM-læringen (data-filen) var lavet og gennemførte herefter henholdsvis en EM-læring i den egenudviklede kode med anvendelse af Hugin API´et og en EM-læring via Hugin GUI (filer fra dette findes på CD i direktoriet movielens\movielensjava\hugintest). Resultaterne af disse kørsler var ens indtil 19. iteration i EM-algoritmen, hvorefter der opstår forskelle i log-likelihood værdierne. Jeg kan ikke umiddelbart forklare disse forskelle – se Bilag side 21 - men de underliggende sandsynlighedsfordelinger i NET-filerne fra kørslerne via henholdsvis Hugin API og Hugin GUI er selvsagt forskellige (Hugin GUI giver i øvrigt identisk output på to gentagne kørsler med identisk input).

Kørselstider

Kørselstiderne kan ikke bestemmes helt præcist, idet der har været afviklet andre programmer på maskinen samtidig med udførelse af programmet til estimering af brugerbedømmelser har været afviklet. En logning – se billag s. 22 - sent om natten giver dog et grundlag for at sammenligne de to metoders performance i forhold til hinanden.

Handling	Forbrugt tid mm:ss,hs

Fil 1 clusternode 50 states EM-læring	38:59,17
Fil 1 clusternode 50 states propagation + retrieve beliefs	00:12,92
Fil 2 hukommelsesbaseret - vektormetoden	03:13,12
Fil 2 hukommelsesbaseret - correlationsmetoden	03:03,90
Fil 2 clusternode 5 states EM-læring	13:41,60
Fil 2 clusternode 50 states propagation + retrieve beliefs	00:06,44

Af ovenstående tabel ses det, at når modellen er indlært i den modelbaserede metode bruges der kun mellem 3-7% af den tid, som det tager at beregne bedømmelses estimater med den hukommelsesbaserede metode.

Delkonklusion

Der er udviklet et program, som med den hukommelsesbaserede metode kan estimere brugerbedømmelser på to forskellige måder. Programmet kan desuden estimere brugerbedømmelser vha den modelbaserede metode ved anvendelse af Hugin API´et. Der stilles i afsnittet store spørgsmålstegn ved de resultater, der beregnes med den modelbaserede metode, men det har ikke været muligt at fastslå, at resultaterne er fejlbehæftede. Andre mulige årsager til underlige data fra Hugin API ´et kunne være, at konvergenskriteriet for EM-læringsalgoritmen var for groft, at der var fejl i inddata til EM-læringsalgoritmen, for få inddata til EM-læringsalgoritmen eller, at Hugin API´et skulle have opsat andre betingelser for programafviklingen end de anvendte. Det har imidlertid ikke været muligt at finde nogle fejl på disse områder. Resultaterne fra den hukommelsesbaserede metode er af en væsentligt bedre kvalitet men afviklingstiden for den modelbaserede metode er som forventet betydeligt bedre – kun 3-7% af den tid det tager med den hukommelsesbaserede metode.

Konklusion

Rapporten beskriver collaborativ filtrering vha en hukommelsesbaseret metode contra en modelbaseret metode. Formålet med rapporten var via et praktisk forsøg at finde ud af, om den modelbaserede metode kunne overgå den hukommelsesbaserede metode. Teorierne for begge metoder er kort præsenteret og der er lavet en gennemgang af det overordnede programdesign i den egenudviklede kode. Der er stillet spørgsmålstegn ved, om Hugin API´et er korrekt anvendt, idet gennemsnitsfejlene af estimere bedømmelser er upåvirket af antallet af antallet af states på clusternoden. Det har imidlertid ikke været muligt at finde en fejl i den egenudviklede kode.
Resultaterne af programafviklingerne peger på den hukommelsesbaserede metode som den mest præcise, mens den modelbaserede metode som forventet langt overgår den hukommelsesbaserede metode, når det kommer til performance. Her skal man dog tage i betragtning at den modelbaserede metode i et kørende system jævnligt skal have opdaterede sandsynlighedsfordelinger, og at indlæringstiden for disse er forholdsvis lang. Alt i alt vil jeg konkludere, at den modelbaserede metode leverev forholdsvist pæne resultater med en god performance, og derfor vil den være interessant at arbejde videre med, når det er verificeret, at resultaterne fra den er korrekte.

Efterrationalisering – og det videre forløb

Derfor tog det lidt ekstra tid

Tidsforbruget til udarbejdelsen af rapporten har været noget større end forventet, idet specielt det praktiske programmeringsarbejde har budt på udfordringer. Nærværende afsnit beskriver valg af produkter til udarbejdelse af rapport og udførelse af beregninger, samt nogle af de udfordringer der i øvrigt har været i forbindelse med udarbejdelsen af rapporten. Afsnittet kan virke overflødigt som dokumentation af problemet der var ønsket undersøgt i problemformuleringen, men her er det alligevel medtaget, da det dels vil indeholde lidt relevant information for folk der skal i gang med en tilsvarende opgave, dels også illustrerer øvrige "omkostninger" ved at udarbejde en teknisk rapport som denne.
Mine udfordringer ud over de fagtekniske i projektforløbet kan kort beskrives ved:

- der bliver udført tunge beregninger hvilket nødvendiggjorde investering i ny PC (Tidsforbrug på AMD Phenom(tm) 8650 Triple-Core Processor, 2,6Ghz, 6GB RAM er vurderet til ca. 4døgn for en komplet kørsel, men der har ikke været udført en uafbrudt kørsel. Der blev kun afviklet kode i en tråd, så maskinen havde ledig kapacitet under det meste af kørslen).
- Hugin Expert 7.0 full version blev installeret på ovennævnte PC og anvendt til arbejdet med bayesianske net (tidligere var Hugin Expert 7.0 også blevet installeret på en mindre bærbar PC).
- Programkoden er skrevet i java 6.0 ved anvendelse af Eclipse 3.4.2 - http://www.eclipse.org/platform. Dette var et simpelt valg for mig, selvom dokumentationen af Hugin API udelukkende var rettet mod C og C++.
- Hugin API har ikke været simpelt at arbejde med, og alle anvendelser af API-metoder blev derfor sammenlignet med Hugin GUI for at verificere korrekt anvendelse.
- Det blev nødvendigt at lægge en fornuftig logning ind i javakoden (log4j ver.1.2.15 - http://logging.apache.org/log4j/1.2/index.html - blev valgt til dette, og det tog lidt tid at konfigurere værktøjet til at logge error- og progresslogninger i to forskellige filer).
- Programkode er udviklet og testet ved anvendelse af unittests– (junit 4_4.3.1 - http://www.junit.org/- til Eclipse blev anvendt).
- Det viste sig, at Hugin API´et ikke kunne behandle alle filmene i en kørsel. Det tog en del tid at finde frem til dette, da API´et ikke direkte melder om fejl, når det forsøges at køre EM-læring på den fulde datamængde. Problemet ses lettest via Hugin GUI´en, hvor der meldes om fejl i dataformat, når man forsøger at loade casedata til EM-læring ind på netværket.
- Når det bayesianske netværk dannes med Hugin API´et skal man dels sørge for at kalde node-metoderne `node.getTable()` og `node.getExperienceTable()`, idet disse tabeller skal være tilknyttet nodes for, at læringsdata kan køres ind på modellen. Så vidt vides dannes disse tabeller ikke automatisk under indlæsning af cases.
- Data i nodetabellerne skal randomiseres, så EM-algoritmen ikke returnerer symmetriske tabeller, som ikke er anvendelige til estimere af brugerbedømmelser i forbindelse med indlægning af evidens. Hugin API´et indeholder så vidt vides ikke en metode til dette, men API´et er heldigvis ikke særligt kritisk overfor statedata genereret med `java.util.Math.random()`.
- Når evidence er lagt på og estimeret bedømmelse/belief er trukket ud, skal domænet initialiseres inden man kan foretage en ny iteration.

Det var oprindeligt planen, at rapporten skulle suppleres med en disc, hvorpå der fandtes en executable jar-fil, der kunne generere alle filer og resultater præsenteret i rapporten, men da afviklingen af koden bl.a. kræver adgang til en korrekt installeret Hugin Expert leveres koden uden jar-fil (Opsætningen af fx classpath er ikke helt triviel og der skal også sendes en parameter til javaVM).
UML-diagrammer præsenteret i rapporten er dannet i open source produktet ArgoUML - http://argouml.tigris.org/. Produktet er simpelt at arbejde med, kan præsentere de fleste begreber i UML og autogenerere javakode (er dog ikke anvendt), men tegning af sekvensdiagrammer er meget besværlig, da man tilsyneladende ikke kan editere elementerne, når de først er lagt ind i diagrammet.
Rapporten er skrevet i open source produktet OpenOffice.org 2.4.2 - http://da.openoffice.org/ - og produktet er velegnet både til præsentation af formler og grafer.

Det videre forløb

Set fra et programmeringssynspunkt skal der laves en fornuftig GUI eller programmerne skal pakkes i en jar-fil og indpasses i en web-applikation. Det udviklede program er forholdsvist følsom overfor fejl i input idet der fx ikke kan laves beregninger af brugerbedømmelser af film, som ikke er registreret i film-inputfilen. Min holdning er, at der ikke skal laves for meget kode til at rette op på den slags åbenlyse fejl, og ud fra det synspunkt mangler den eksisterende kode ikke nogen betydelig udvidelse for bedre fejlhåndtering.

Det har under udviklingen af programkoden vist sig at være godt at arbejde med unit-tests, idet flowkontrollen i programkoden, der udregner de matematiske udtryk, ikke er så simpel, at den altid kan forventes at virke i første forsøg. Da den hukommelsesbaserede metode i høj grad er afhængig af en effektiv programafvikling, vil det specielt være relevant at lave en analyse af kompleksiteten og måske også en optimering af algoritmerne til beregning af estimerede bedømmelser inden for denne metode. En sådan analyse vil også fortælle noget om, hvordan metoden vil performe når der kommer flere film, brugere og brugerbedømmelser ind i systemet. Dette arbejde har jeg ikke fundet tid til at gøre.

Det bayesianske netværk er opbygget med en antagelse om, at data er MCAR, og det er sikkert ikke tilfældet, idet brugerne ikke har fået besked om at se 100 vilkårlige film og bedømme hver enkelt af disse, men derimod har afgivet bedømmelser af de film, som de selv har valgt at se. Derfor vil brugerne på forhånd have gjort nogle valg af hvilke film de ønskede at se ud fra den markedsføring, der har været foretaget af den enkelte film. En håndtering af dette kunne laves ved at tilføje et ekstra ”lag” knuder på modellen, således modellen også kunne tage højde for brugernes fravalg af film. Dette lag skal sikkert placeres under filmknuderne, men indførelsen af laget vil kunne øge modellens kompleksitet og gøre den betydeligt tungere at lave beregninger på.

Litteraturliste

Collab: John S. Breese, David Heckerman, Carl Kadie, Empirical Analysis of Predictive Algorithms forCollaborative Filtering, 1998
Calc: Fraleigh, John B, Calculus with analytic geometry, 1985
BNDG: Jensen, Finn V. og Nielsen, Thomas D., Bayesian Networks and Decision Graphs, 2007

Bilag 1 – Uddrag af output fra Hugin-logs

Hugin GUI	Hugin API via egenudviklet kode
EM learning with 943 cases: **Log-likelihood for iteration 1: -141276** **Log-likelihood for iteration 2: -124741** **Log-likelihood for iteration 3: -117860** **Log-likelihood for iteration 4: -113580** **Log-likelihood for iteration 5: -110622** **Log-likelihood for iteration 6: -108458** **Log-likelihood for iteration 7: -106812** **Log-likelihood for iteration 8: -105523** **Log-likelihood for iteration 9: -104492** **Log-likelihood for iteration 10: -103651** **Log-likelihood for iteration 11: -102955** **Log-likelihood for iteration 12: -102371** **Log-likelihood for iteration 13: -101875** **Log-likelihood for iteration 14: -101450** **Log-likelihood for iteration 15: -101083** **Log-likelihood for iteration 16: -100763** **Log-likelihood for iteration 17: -100482** **Log-likelihood for iteration 18: -100234** **Log-likelihood for iteration 19: -100014** Log-likelihood for iteration 20: -99816.8 Log-likelihood for iteration 21: -99614.7 ... Log-likelihood for iteration 128: -74263.9 EM learning - final results: Log-likelihood: -74256.8 AIC: -134211 BIC: -279571	**EM learning with 943 cases:** **Log-likelihood for iteration 1: -141276** **Log-likelihood for iteration 2: -124741** **Log-likelihood for iteration 3: -117860** **Log-likelihood for iteration 4: -113580** **Log-likelihood for iteration 5: -110622** **Log-likelihood for iteration 6: -108458** **Log-likelihood for iteration 7: -106812** **Log-likelihood for iteration 8: -105523** **Log-likelihood for iteration 9: -104492** **Log-likelihood for iteration 10: -103651** **Log-likelihood for iteration 11: -102955** **Log-likelihood for iteration 12: -102371** **Log-likelihood for iteration 13: -101875** **Log-likelihood for iteration 14: -101450** **Log-likelihood for iteration 15: -101083** **Log-likelihood for iteration 16: -100763** **Log-likelihood for iteration 17: -100482** **Log-likelihood for iteration 18: -100234** **Log-likelihood for iteration 19: -100014** Log-likelihood for iteration 20: -99816.3 Log-likelihood for iteration 21: -99616.1 ... Log-likelihood for iteration 125: -75739.7 EM learning - final results: Log-likelihood: -75732.4 AIC: -135686 BIC: -281047

Bilag 2 – Uddrag af log4j.log til brug for performancesammenligning

[2009-06-08 03:11:02,765] INFO709644[main] - Main.buildAndLoad(Main.java:126) - NBN1cs50 Learning Start
[2009-06-08 03:50:01,104] INFO047983[main] - naivebn.NBN.loadCases(NBN.java:171) - NBN1cs50BIC: -746105.238963702
[2009-06-08 03:50:01,930] INFO048809[main] - Main.buildAndLoad(Main.java:130) - NBN1cs50 Learning Slut
[2009-06-08 03:50:01,930] INFO048809[main] - Main.evidenceHandling(Main.java:134) - NBN1cs50 Propagating Start
[2009-06-08 03:50:14,848] INFO061727[main] - Main.evidenceHandling(Main.java:146) - NBN1cs50 Propagating Slut
[2009-06-08 03:50:16,107] INFO062986[main] - Main.beregn(Main.java:96) - fil2hukommelseVektor start
[2009-06-08 03:53:29,225] INFO256104[main] - Main.beregn(Main.java:117) - fil2hukommelseVektor slut
[2009-06-08 03:53:29,225] INFO256104[main] - Main.beregn(Main.java:96) - fil2hukommelseCorrel start
[2009-06-08 03:56:33,123] INFO440002[main] - Main.beregn(Main.java:117) - fil2hukommelseCorrel slut
[2009-06-08 03:56:33,203] INFO440082[main] - Main.buildAndLoad(Main.java:126) - NBN2cs5 Learning Start
[2009-06-08 04:10:14,683] INFO261562[main] - naivebn.NBN.loadCases(NBN.java:171) - NBN2cs5BIC: -153720.2209443922
[2009-06-08 04:10:14,802] INFO261681[main] - Main.buildAndLoad(Main.java:130) - NBN2cs5 Learning Slut
[2009-06-08 04:10:14,803] INFO261682[main] - Main.evidenceHandling(Main.java:134) - NBN2cs5 Propagating Start
[2009-06-08 04:10:21,238] INFO268117[main] - Main.evidenceHandling(Main.java:146) - NBN2cs5 Propagating Slut