Stockholms universitet

Modellering av utvecklingsfaktorerna med artificiella neuronnät

Axel Åström

Masteruppsats i försäkringsmatematik
Master Thesis in Actuarial Mathematics

Masteruppsats 2019:8
Försäkringsmatematik
Juni 2019

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Matematisk statistik
Stockholms universitet
Masteruppsats **2019:8**
http://www.math.su.se/matstat

Modellering av utvecklingsfaktorerna med artificiella neuronnät

Axel Åström*

Juni 2019

Sammanfattning

I detta arbete utgår vi från Macks chain ladder modell och en modifiering av denna med en annan struktur för betingad varians av kumulativa utbetalda belopp och visar hur de kan omdefinieras för att kunna ta hänsyn till information om individuella skador. Det görs genom att utvecklingsfaktorerna skattas med artificiella neuronnät där vi i det här fallet har använt oss av så kallade flerlagersperceptroner. Syftet med detta är att undersöka hur väl de två olika metoderna estimerar diverse ultimobelopp för de två simulerade datamängderna. Vidare ville vi också ta reda på om det fanns några eventuella fördelar med att inkludera mer information om de individuella skadorna. Resultatet blev att båda metoderna verkar lämpliga att använda sig av för att skatta framtida utbetalningar för de båda datamängderna. Dessutom gav metoden som bygger på den modifierade chain ladder modellen bättre skattningar jämfört med den andra metoden för båda datamängderna. Det var oväntat eftersom datamängderna var anpassade efter respektive metods antaganden.

*Postadress: Matematisk statistik, Stockholms universitet, 106 91, Sverige. E-post:axel.per.astrom@gmail.com. Handledare: Filip Lindskog.

Abstract

In this study, we start from Mack's chain ladder model and a modification based on a different conditional variance structure for cumulative paid amounts and show how they can be redefined in order to take into account information about individual claims. This is done by estimating the development factors with artificial neural networks, where in this case we have used so-called multilayer perceptrons. The purpose of this is to investigate how well the two different methods estimate various ultimate claims amount for the two simulated data sets. Furthermore, we also wanted to find out if there were any possible benefits of including more information about the individual claims. The result was that both methods seem appropriate to use to estimate future payments for the two data sets. In addition, the method based on the modified model gave better estimates compared to the other method for both data sets. This were unexpected since the data sets was adapted to the respective method's assumptions.

Förord och tack

Detta arbete utgör ett examensarbete på masternivå om 30 hp inom försäkringsmatematik vid matematiska institutionen på Stockholms Universitet.

Jag vill tacka min handledare Filip Lindskog på matematiska institutionen för hans värdefulla feedback. Ett särskilt tack även till min flickvän Emelie och resten av min familj som stöttat mig under studietiden.

Innehåll

1. Inledning 5

2. Bakgrund 6

2.1 Reservsättning 6
2.2 Chain ladder-metoden 6
2.3 Artificiella neuronnät 9

3. Modell 17

3.1 Antaganden 18
3.2 Modellera med neuronnät 19

4. Datasimulering 21

5. Simulering av nätverk 25

5.1 Förbehandling av egenskaperna 25

6. Resultat 26

7. Diskussion. 35

Referenser 37

Appendix 38

1. Inledning

Inom klassisk reservsättning är det vanligt att man använder metoder som utgår från att datan är aggregerad över flera försäkringsportföljer i form av så kallade skadetrianglar. Ett avgörande antagande för att detta ska vara lämpligt är att dessa portföljer är tillräckligt homogena. En stor anledning till varför intresset inte varit större för att undersöka hur de individuella skadorna har utvecklat sig beror mycket på att det inte alltid har funnits tillräckligt med datorkraft för att hantera många beräkningar. Därför har chain ladder-metoden, som troligtvis är en av de mest populära metoderna för att beräkna reserverna används mycket genom åren, eftersom den är enkel att förstå och att genomföra rent beräkningsmässigt. Men med hjälp av den tekniska utveckligen har det blivit allt mer vanligt att studera reservsättning på mikronivå. Detta innebär att man använder sig av information om de individuella skadorna till skillnad från makronivån, som är beskriven ovan. Makromodeller kan ibland leda till en missvisande analys då eventuellt viktig information inte har hanterats.

I det här arbetet ska vi utgå från Macks chain ladder-modell och en modifiering av denna med en annan struktur för betingad varians av kumulativa utbetalda belopp. Vidare modifierars modellerna genom att det ursprunliga regressionsantagandet ersätts med en icke linjär neuronnätsregressionsmodell, eftersom vi vill få metoder som kan inkludera information om individuella skador. Huvudsyftet med arbetet är att förstå och förklara nuvarande teori som finns inom ämnet. Metoderna tillämpas även på simulerad data där målet är att ta reda på om de är lämpliga för att estimera reserven. Vi vill även jämföra hur resultaten skiljer sig från de med chain ladder-metoden. Därefter är det av intresse att analysera resultaten för att undersöka eventuella fördelar med att inkludera mer information om de individuella skadorna.

Arbetet inleds med att vi presenterar teori för artificiella neuronnät, mer precis flerlagersperceptroner. Vidare definieras de två chain ladder liknande metoderna. Båda metoderna beror på tre stycken faktorer som i det här arbetet kallas A, B respektive C där varje faktor har en indelning i klasser. Efter det går vi in på hur datamängderna har simulerats och avslutningsvis tillämpas metoderna på datan. Resultatet blev att båda metoderna verkar lämpliga för att estimera reserven för båda datamängderna, både när det kommer till det totala värdet samt för klasserna inom de olika faktorerna var för sig. Dessutom pekar resultatet mot att metoden som utgår från den modifierade chain ladder modellen gav bättre skattningar jämfört med den andra metoden och chain ladder-metoden. I början kan det verka konstigt eftersom datamängderna är simulerade så att de ska vara anpassade efter respektive metods antaganden.

2. Bakgrund

I den här delen kommer vi att förklara diverse teori som är lämplig för att vi senare ska kunna definiera modellen som är av intresse i det här arbetet.

2.1 Reservsättning

Reservsättning handlar om att man vill beräkna hur mycket pengar, det vill säga reserv, som ett försäkringsbolag behöver avvara för att kunna täcka framtida kostnader för ej slutreglerade skador. En anledning till det är att man vill veta hur mycket ersättning som kommer att utbetalas för respektive skadeår. Inom sakförsäkring är det vanligt att det uppstår en fördröjningstid mellan när skadan inträffar respektive rapporteras. Därefter sker en eller flera utbetalningar tills dess att skadan har slutreglerats. Själva utbetalningsmönstret kan skilja sig relativt mycket mellan olika typer av försäkringar och skador.

Reserven brukar delas upp i två stycken poster. Den ena kallas för IBNR och står för "incurred but not reported" vilket motsvarar reserven för de skador som inträffat under det specifika skadeåret men ännu inte rapporteras. Den andra kallas för RBNS som står för "reported but not settled". Detta är reserven för skador som har rapporterats men som inte är slutreglerade eftersom man inte vet den totala skadekostnaden än. För att skatta dessa reserver finns det olika metoder men en av de vanligaste är chain ladder-metoden. Det är en metod som skattar hela reserven och inte respektive post var för sig.

2.2 Chain ladder-metoden

Chain ladder metoden är en av de vanligaste metoderna som används för att beräkna reserven. För den gäller det att $C_{i,j}$ betecknar de kumulativa utbetalda beloppen för skador som inträffat under skadeår i efter j utvecklingsår. Vi har att $1 \leq i \leq I$ och $1 \leq j \leq I$. Det gäller även att modellen är fördelningsfri och den är baserad på utvecklingsmönstret för de kumulativa utbetalningarna. Modellen definieras under följande antaganden.

$$E[C_{i,j+1}|C_{i,j}] = f_j C_{i,j}, j = 1, ..., I-1 \tag{2.1}$$

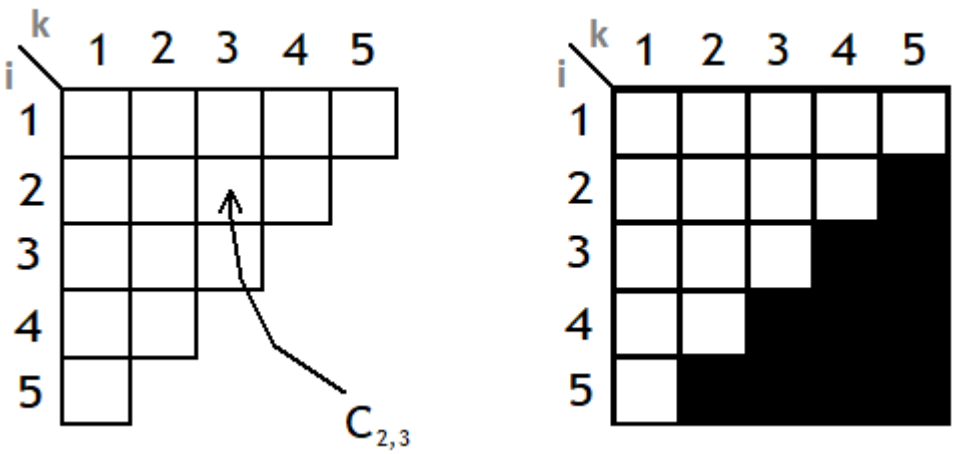
$$\{C_{i,j}, j = 1, ..., I\} \text{ är oberoende av } \{C_{k,j}, j = 1, ..., I\} \text{ för alla } i \neq k \tag{2.2}$$

$$Var(C_{i,j+1}|C_{i,1}, ..., C_{i,j}) = C_{i,j}\sigma_j^2 \tag{2.3}$$

Parametrarna f_j kallas för utvecklingsfaktorer och σ_j^2 är proportionalitetskonstanter. Utifrån antagandena ovan kan vi få fram att

$$E[C_{i,I}|C_{i,I-i+1}] = C_{i,I-i+1} \prod_{j=I-i+1}^{I-1} f_j \tag{2.4}$$

vilket motsvarar den totala förväntade kostnaden, det vill säga ultimo, för skador som inträffat under skadeår i. Historisk observerad data $C_{i,j}$ finns för $i+j < I+1$ och vanligtvis presenteras den i en skadetriangel som kan se ut enligt Figur 1 nedan.[1]



Figur 1: Exempel på skadetriangel med $I = 5$ där det svarta området motsvarar framtida utbetalningar som ska predikteras, medan de andra cellerna innehåller historisk observerad data. Figuren är hämtad från [11], s. 26.

Utvecklingsfaktorerna f_j kan predikteras enligt.

$$\hat{f}_j = \frac{\sum_{i=1}^{I-j} C_{i,j+1}}{\sum_{i=1}^{I-j} C_{i,j}} \tag{2.5}$$

Vi antar att skadekostnaden kommer att ha lika stor procentuell förändring mellan utvecklingsår j och $j+1$ oavsett vilket skadeår som är av intresse.[1] Men utöver att prediktera ultimo är vi intresserade av att veta det genomsnittliga avstånd mellan $\hat{C}_{i,I}$ och det faktiska värdet $C_{i,I}$. Detta kan mätas genom att man beräknar medelkvadratfelet för prediktorn $\hat{C}_{i,I}$ och det faktiska värdet $C_{i,I}$. Det definieras enligt

$$msep(\hat{C}_{i,I}, \hat{C}_{i,I}) = E[(C_{i,I} - \hat{C}_{i,I})^2] \tag{2.6}$$

Hellre vill vi beräkna det betingade MSEP som under antagandena för chain ladder fås av

$$msep(C_{i,I}, \hat{C}_{i,I}) = C_{i,I}^2 \sum_{k=I+1-i}^{I-1} \frac{\sigma_k^2}{f_k^2} (\frac{1}{C_{i,k}} + \frac{1}{\sum_{j=1}^{I-k} C_{j,k}}) \tag{2.7}$$

Parametrarna f_k skattas enligt formel (2.3) och σ_k^2 skattas enligt formeln (2.8) nedan.

$$\sigma_k^2 = \frac{1}{I-k-1} \sum_{i=1}^{I-k} C_{i,k} (\frac{C_{i,k+1}}{C_{i,k}} - f_k)^2, \qquad 1 \leq k \leq I-2. \tag{2.8}$$

Vi märker att formel (2.8) inte kan hjälpa oss att få en skattning för σ_{I-1}^2 eftersom det inte är möjligt att skatta varians med en ensam kvot $C_{1,I}/C_{1,I-1}$. Problemet kan lösas på två sätt, antingen sätter vi $\sigma_{I-1}^2 = 0$ om vi tror att skadeåret är slutbetalt eller extrapolerar vi enligt nedanstående formel.

$$\sigma_{I-1}^2 = \min(\sigma_{I-2}^4/\sigma_{I-3}^2, \min(\sigma_{I-2}^2, \sigma_{I-3}^2)) \tag{2.9}$$

Det går även att visa att för chain ladder så är $msep(C_{i,I}, \hat{C}_{i,I})$ lika med medelkvadratfelet för reservprediktorn $R_i = C_{i,I} - C_{i,I+1-i}$ där $i = 2, ..., I$. Således kan standardfelet för den predikterade reserven för skadeår i fås genom att man beräknar $s.e.(C_{i,I}) = \sqrt{msep(C_{i,I})}$. Tidigare nämnde vi också att det är av intresse att titta på den totala reserven R. MSEP för R fås enligt denna formel.[1]

$$(s.e.(R))^2 = \sum_{i=2}^{I} \Big\{ \sigma_i^2 + C_{i,I} (\sum_{j=i+1}^{I} C_{j,I}) \sum_{k=I+1-i}^{I-1} \frac{2\sigma_k^2/f_k^2}{\sum_{n=1}^{I-k} C_{n,k}} \Big\} \tag{2.10}$$

Däremot existerar det i vissa fall skadetrianglar som inte uppfyller antagandena för chain ladder-metoden. I fallet då antagandet som ges i formel (2.3) inte inte uppfylls kan en modifierad chain ladder-metod användas, den har en annan struktur för betingad varians av kumulativa utbetalda belopp. Vi väljer framöver att kalla den metoden för den viktade chain ladder-metoden och för den skattas utvecklingsfaktorerna f_j enligt.

$$\hat{f}_{j0} = \sum_{i=1}^{I-j} \frac{C_{i,j}^2}{\sum_{i=1}^{I-j} C_{i,j}^2} \frac{C_{i,j+1}}{C_{i,j}} \tag{2.11}$$

Parametrarna $\hat{f}_{j0}$ är de $C_{i,j}^2$-viktade medelvärdet av de individuella utvecklingsfaktorerna $C_{i,j+1}/C_{i,j}$. Estimatorn för f_j, enligt formel (2.5), är det $C_{i,j}$-viktade medelvärdet.[1]

2.3 Artificiella neuronnät

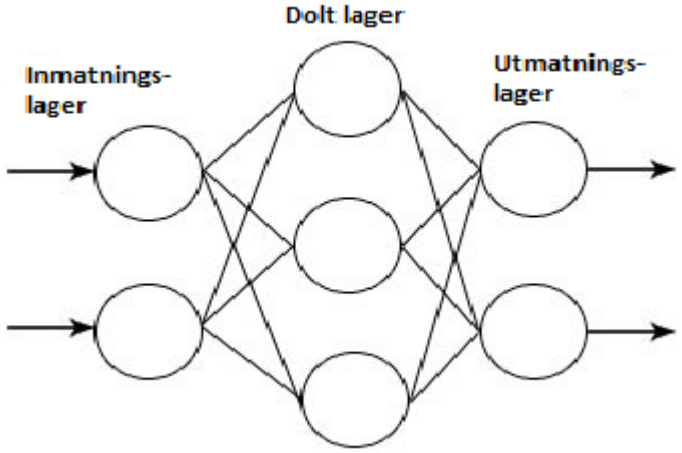
Artificiella neuronnät är en familj av modeller som är inspirerade av hur den mänskliga hjärnan är konstruerad. Modellerna består av så kallade neuroner som är anslutna till varandra på olika sätt likt ett nätverk. Genom att ändra nätverkstrukturen eller använda olika icke linjära transferfunktioner kan man, med hjälp av en inlärningsalgoritm, få modeller för olika typer av olinjära beroenden. Exempelvis kan de användas för att ge prognoser om framtida händelser.[3]

2.3.1 Perceptronen

Perceptronen är ett mycket enkelt neuronnät som beskrevs av Frank Rosenblatt under 1950-talet. Idag är det vanligare att använda andra modeller av artificiella neuroner. Men innan vi går in på någon av dessa lite mer moderna modeller kan det vara värt att först försöka förstå hur perceptronen är uppbyggd. Den består av en neuron som kan ta emot en eller flera binära ingångsvärden x_j där varje ingång har en egen vikt w_j. Vidare summeras produkten av ingångsvärden och vikten för alla ingångar. Det gäller då att neuronen utmatar antingen värdet ett eller noll beroende på om summan $\sum_j w_j x_j$ är större eller mindre än ett valt tröskelvärde.[2]

2.3.2 Flerlagersperceptroner

Genom att kombinera flera perceptroner kan man lösa problem som en perceptron inte klarar av. Man får en flerlagersperceptron som är ett neuronnät med ett ingående lager, ett eller flera dolda lager och ett utmatningslager, enligt Figur 2 nedan.



Figur 2: Exempel på en flerlagersperceptron som har ett dolt lager. Figuren är hämtad från [2].

Varje neuron mottar ingångsdata från en eller flera ingångar som vardera har en vikt likt fallet med en perceptron. Det är ett framåtmatande nätverk vilket innebär att signalerna bara får överföras till nästa lager, det vill säga framåt. Varje neuron har även en transferfunktion $r(t)$ och nedan i Tabell 1 ges exempel på några av de vanligaste.[3]

Namn	$r(t)$
Tröskel	$\begin{cases} 1, t < 0 \\ 0, t \geq 0 \end{cases}$
Sigmoid	$1/(1+e^{-t})$
Hyperbolisk tangent	$(e^{2t}-1)/(e^{2t}+1)$
Identitet	t

Tabell 1: Vanliga transferfunktioner för neuroner i en flerlagersperceptron.

Både sigmoidfunktionen och den hyperboliska tangentfunktionen i Tabell 1 påminner om en utjämnad version av en tröskelfunktionen. Det leder till att själva inlärningsalgoritmen förenklas, eftersom en liten ändring i exempelvis vikterna möjligtvis leder till en liten ändring i utmatningen, vilket inte gäller för tröskelfunktionen.[2]

Utmatningen från neuron j ges av $r(b_j^l + \sum_k w_{j,k}^l \alpha_k^{l-1})$ där $w_{j,k}^l$ betecknar vikten som går från neuron k i det tidigare lagret $(l-1)$ till neuron j i lager l. Parametrarna α_k^{l-1} är utmatningen från neuron k i det förra lagret. Det finns också en parameter som kallas för bias b_j^l som finns för varje neuron för att möjliggöra en justering av alla ingångsvärden.[2]

2.3.3 Bakåtpropagation

Inlärningsalgoritmen för flerlagerperceptroner kallas bakåtpropagation och det är en tränad inlärningsalgoritm. Det innebär att nätverket byggs upp genom att känd mätdata och svar matas in. Om inte resultaten överensstämmer beräknas felet och därefter justeras vikterna respektive bias. Detta upprepas och om nätverkets design är rätt kommer nätet att konvergera mot de sanna resultaten.

Bakåtpropagation görs vanligen genom iterativ tillämpning av två specifika steg. Första steget kallas framåtpropagation, där matas träningsdata in i nätverket. Det innebär att inmatningens sanna utmatning är given. Det handlar om att undersöka hur mycket nätverkets utmatning skiljer sig från det faktiska värdet.

Utöver detta kan det även vara värt att nämna att vikterna respektive bias, till en början, är slumpmässigt valda. Vidare inleds bakåtpropagationssteget som handlar om minska felet för träningsdatan genom att justera vikterna och bias. Det görs genom att gradienten beräknas för en förlustfunktion L med avseende på en matris W bestående av, ovan nämnda, vikter och bias som finns i nätverket. Vanligtvis väljs L som medelkvadratfelet. Det gäller att ett steg tas för varje iteration i gradientens negativa riktning åt det hållet som förlustfunktionen L minskar mest, enligt följande

$$w_k \rightarrow w'_k = w_k - \frac{\eta}{n} \sum_{j=1}^{n} \frac{\partial L_{X_j}}{\partial w_k} \tag{2.12}$$

$$b_l \rightarrow b'_l = b_l - \frac{\eta}{n} \sum_{j=1}^{n} \frac{\partial L_{X_j}}{\partial b_l} \tag{2.13}$$

där n betecknar träningsdatans storlek. Parametern η definieras som inlärningsfrekvensen, vars uppgift är att kontrollera steglängden för stegen som tas i bakåtpropagationen. Efter detta börjar proceduren om igen med framåtpropagationssteget, det vill säga en ny epoch inleds, och detta upprepas tills att förlustfunktionen L har nått ett minimumvärde.[2]

2.3.4 Elastisk bakåtpropagation och RMSprop

Vid vanlig bakåtpropagation är man i behov av att specificera ett parametervärde för inlärningsfrekvensen η. Vanligtvis väljs då en relativt kort steglängd eftersom det annars finns en risk att ett eventuellt minimum missas. Det i sin tur kan göra inlärningsprocessen väldigt långsam. Ett alternativ är att använda sig av elastisk bakåtpropagation som är en mer tidseffektiv träningsalgoritm. För den gäller det att alla vikter respektive bias har en egen inlärningsfrekvens som man inte behöver specificera utan de kan variera mellan de olika iterationerna. Våra vikter respektive bias uppdateras genom att man endast kollar på gradientens tecken och om tecknet är detsamma mellan två successiva iterationer, kan inlärningsfrekvensen ökas. Annars minskas inlärningsfrekvensen och detta upprepas till man når ett minimum.[3]

En annan möjlig variant av gradient descent som kan användas är RMSprop optimeraren. Den gör att inlärningsfrekvensen kan öka vilket leder till att nätvärket konvergerar snabbare vid inlärningsalgoritmen. Den skiljer sig från vanlig gradient descent när det kommer till hur själva gradienten beräknas.[6]

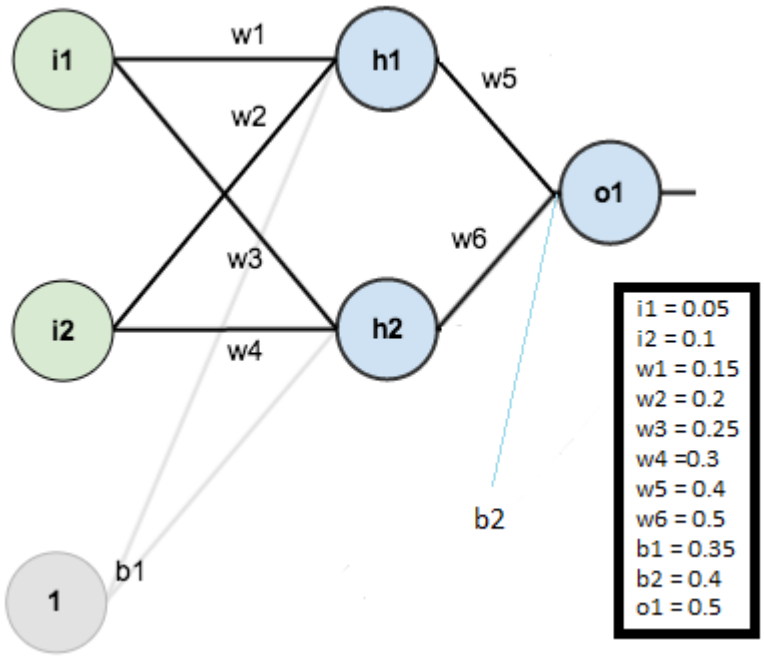
2.3.5 Stokastisk gradient descent

Stokastisk gradient nedstigning är en optimeringsprocedur där gradienten av förlustfunktionen L inte beräknas för hela träningsdatan vid varje iteration. Utan man plockar slumpmässigt ut en mindre del av datan av storlek m för att estimera gradienten ΔW där W är en matris bestående av vikterna och biasparametrarna som finns i neuronnätet. Anledningen till att det är av intresse att beräkna den beror på att man vill veta hur parametrarna ska ändras för att minska värdet för förlustfunktionen maximalt. Gäller då att gradienten $\Delta W = \frac{\sum_{j=1}^{n} \Delta W_{X_j}}{n}$ kan skattas av formlen $\frac{\sum_{j=1}^{m} \Delta W_{X_j}}{m}$ där $X_1, ..., X_m$ representerar den slumpmässigt valda träningsdatan medan n är träningsdatans totala storlek liksom ovan. Anledningen till varför man väljer att inte undersöka hela datan är för att inlärningsprocessen ska gå snabbare.[2]

2.3.6 Exempel på artificiella neuronnät

För att öka förståelsen för neuronnät ska vi illustrera två exempel. Ett exempel på hur bakåtpropagationen går till och ett exempel där vi tränar och testar ett nätverk med simulerad data.

Exempel 1: I det här exemplet tänkte vi undersöka en epoch i bakåtpropagtionen när träningsdatan är av storlek $n = 1$ för ett enkelt neuronnät som illustreras i Figur 3 nedan. Transferfunktionen är en så kallad sigmoid som ges i Tabell 1.[4]



Figur 3: Ett exempel på en flerlagersperceptron med ett dolt lager tillsammans med träningsdatan och initialt givna vikter. Figuren är hämtad från [4].

Den totala inmatningen till neuron h_1 ges av:

$$\text{net}_{h_1} = w_1 i_1 + w_2 i_2 + b_1 = 0.3375$$

Således har vi allt som behövs för att beräkna utmatningen från h_1 som beräknas enligt följande:

$$\text{out}_{h_1} = \frac{1}{1 + e^{-\text{net}_{h_1}}} = 0.5937$$

På samma sätt som ovan får vi att $\text{out}_{h_2} = 0.5968844$. Vi vill nu beräkna out_{o_1} som är nätverkets utmatning vilket kan göras så här:

$$\text{net}_{o_1} = w_5 \text{out}_{h_1} + w_6 \text{out}_{h_2} + b_2 = 0.9357502$$

$$\text{out}_{o_1} = \frac{1}{1 + e^{-\text{net}_{o_1}}} = 0.7182404$$

Vi går nu vidare och beräknar felet när vi låter förlustfunktionen L vara medelkvadratsumman $\frac{1}{2n}\sum_{j=1}^{n}(\text{out}_{sant} - \text{out}_{o_1})^2$. Den blir lika med 0.02381444 eftersom den faktiska utmatningen är 0.5.

Bakåtpropagationssteget inleds och målet med det är att uppdatera vikter i neuronnätet så att nätverkets utmatning överensstämmer bättre med värdet out_{sant}. Vi börjar med att först skapa oss en uppfattning om hur mycket en ändring i

exempelvis vikten w_5 påverkar det totala felet L genom att beräkna $\frac{\partial L}{\partial w_5}$. Det vill säga gradienten med avseende på vikten w_5. Med kedjeregeln får vi att:

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial \text{out}_{o_1}} \frac{\partial \text{out}_{o_1}}{\partial \text{net}_{o_1}} \frac{\partial \text{net}_{o_1}}{\partial w_5}$$

Vi väljer att lösa det del för del.

$$\begin{aligned}
L &= \frac{(\text{out}_{sant} - \text{out}_{0_1})^2}{2} \\
\frac{\partial L}{\partial \text{out}_{o_1}} &= \text{out}_{o_1} - \text{out}_{sant} = 0.2182404 \\
\text{out}_{o_1} &= \frac{1}{1 + e^{-\text{net}_{o_1}}} \\
\frac{\partial \text{out}_{o_1}}{\partial \text{net}_{o_1}} &= \text{out}_{o_1}(1 - \text{out}_{o_1}) = 0.2023711 \\
\text{net}_{o_1} &= w_5 \text{out}_{h_1} + w_6 \text{out}_{h_2} + b_2 \\
\frac{\partial \text{net}_{o_1}}{\partial w_5} &= \text{out}_{h_1} = 0.59327
\end{aligned}$$

Således får vi att $\frac{\partial L}{\partial w_5} = 0.0262021$ och på liknande sätt kan vi beräkna $\frac{\partial L}{\partial w_6} = 0.02636173$. Nu är det av intresse att beräkna gradienten med avseende på vikterna som används vid inmatning till neuronerna i det dolda lagret, det vill säga w_1, w_2, w_3 och w_4. Nedan illusterar vi hur $\frac{\partial L}{\partial w_1}$ kan beräknas. Liksom innan använder vi oss av kedjereglerna vilket ger oss följande:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \text{out}_{h_1}} \frac{\partial \text{out}_{h_1}}{\partial \text{net}_{h_1}} \frac{\partial \text{net}_{h_1}}{\partial w_1}$$

Löser detta del för del och för den första delen får vi följande:

$$\frac{\partial L}{\partial \text{out}_{h_1}} = \frac{\partial L}{\partial \text{net}_{o_1}} \frac{\partial \text{net}_{o_1}}{\partial \text{out}_{h_1}} = \frac{\partial L}{\partial \text{out}_{o_1}} \frac{\partial \text{out}_{o_1}}{\partial \text{net}_{o_1}} \frac{\partial \text{net}_{o_1}}{\partial \text{out}_{h_1}} = 0.01766622$$

Har använt att $\frac{\partial \text{net}_{o_1}}{\partial \text{out}_{h_1}} = w_5$ samt att de övriga beräkningarna redan är gjorda vilket leder till att

$$\begin{aligned}
\frac{\partial \text{out}_{h_1}}{\partial \text{net}_{h_1}} &= \text{out}_{h_1}(1 - \text{out}_{h_1}) = 0.2413007 \\
\text{net}_{h_1} &= w_1 i_1 + w_3 i_2 + b_1 \\
\frac{\partial \text{net}_{h_1}}{\partial w_1} &= i_1 = 0.05
\end{aligned}$$

Resultatet blir att $\frac{\partial L}{\partial w_1} = 0.0002131436$. På samma sätt kan vi beräkna övriga gradienter och dessa presenteras i Tabell 2 nedan. Vidare används dem i följande formel för att uppdatera vikterna i nätverket:

$$w_i' = w_i - \eta \frac{\partial L}{\partial w_i}$$

Parmetern η låter vi vara lika med 0.5 och liksom tidigare presenteras resultaten, tillsammans med de tidigare vikterna, i Tabell 2.

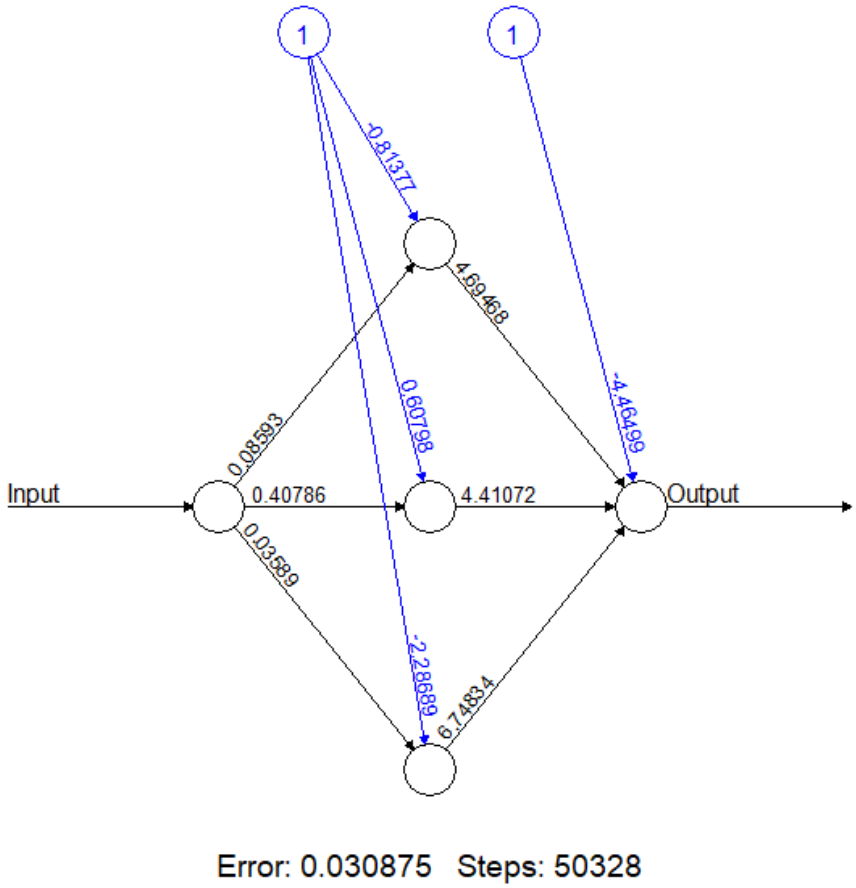
i	$\frac{\partial L}{\partial w_i}$	w_i	w_i'
1	0.0002131436	0.15	0.1498934
2	0.0002656706	0.2	0.1997869
3	0.0004262872	0.25	0.2498672
4	0.0005313413	0.3	0.2997343
5	0.0262021	0.4	0.3868989
6	0.02636173	0.5	0.4868191

Tabell 2: Tabell över de beräknade gradienterna, de initiala vikterna och de uppdaterade vikterna.

Avslutningsvis ska vi beräkna närverkets utmatning med de justerade vikterna och undersöka hur felet har ändrats. Utmatningen out_{o_1} blir lika med 0.7150633. Det innebär att medelkvadratfelet är lika med 0.02312611, vilket är en minskning jämfört med tidigare eftersom det var lika med 0.02381444. Vidare fortsätter bakåtpropagationen på samma sätt som ovan, med flera epochs, tills förlustfunktionen L når ett minimum.

Exempel 2: Vi ska nu använda oss av ett neuronnät, det vill säga en flerlagersperceptron, för att illustrera hur det kan användas för inlärning av kvadratroten. Nätet består av en inmatningsneuron, ett dolt lager med tre neuroner och avslutningsvis en utmatningsneuron. Transferfunktionen för neuronerna i det dolda lagret är sigmoidfunktionen medan transferfunktionen för utmatningsneuronen är identitetsfunktionen. Vidare användas elastisk bakåtpropagation eftersom vi exempelvis inte behöver hitta ett lämpligt värde för inlärningsfrekvensen η.

Som träningsdata väljer vi att simulera 100 stycken likformigt fördelade tal mellan 0 och 100 och dessa betecknas $X_i, i = 1, ..., 100$. Därefter tar vi kvadratroten ur dem och således får vi vår sanna utmatning out_{sann}. Resultatet blir då att vårt tränade nätverk får vikterna och bias som illustreras i Figur 4 nedan.



Figur 4: Ett neuronnät som är tränat för att kunna hantera kvadratroten för värden mindre än 100.

Det som nu är av intresse är att testa hur väl nätverket fungerar. Vi testar det

genom att mata in värden från 1 till 100 och därefter jämförs nätverkets utmatning med de faktiska värdena.

Inmatning	Sann-utmatning	Nät-utmatning
1	1	0.9449646
4	2	2.0349859
9	3	3.0156860
16	4	3.9625719
25	5	5.0078410
36	6	6.0263618
49	7	6.9913928
64	8	7.9886656
81	9	9.0186072
100	10	9.9439464

Tabell 3: Tabell över hur den faktiska utmatningen skiljer sig från utmatningen som ges av neuronnätet som illustreras i Figur 4.

Från Tabell 3 kan vi då se att värdena överensstämmer med varandra ganska bra vilket också var syftet med själva exemplet.

3. Modell

I det här arbetet är vi intresserade av att utöka den klassiska chain ladder-modellen så att vi kan inkludera information om individuella skador, för att det ska bli möjligt att förstå strukturella skillnader mellan olika typer av fodringar. Kort sagt görs detta genom att vi först justerar några antaganden i modellen och därefter modelleras utvecklingsfaktorerna med neuronnät.

Innan vi går in på antaganden kan det vara värt att gå igenom lite relevant notation. Först har vi egenskapsrummet χ där $x \in \chi$ beskriver egenskaperna hos de enskilda fordringarna. Det består, i det här arbetet, av information om tre stycken faktorer.

- Faktor $A \in \{1, 2, 3\}$ som består av tre stycken klasser.
- Faktor $B \in \{1, 2, 3, 4\}$ som består av fyra stycken klasser.
- Faktor $C \in \{1, 2, ..., 10\}$ som består av tio stycken klasser.

Det finns totalt 120 unika egenskapskombinationer. Ingen större vikt kommer läggas på att ge exempel på vad respektive faktor skulle kunna ge information om. Det som är av störst betydelse är att vi inte inkluderar skadeår i eftersom det

skulle strida mot antagandet i chain ladder-metoden gällande att f_j inte beror på skadeåret. Vidare definieras $C_{i,j}(x)$ som de kumulativa fordringsbetalningarna för skador som inträffat skadeår i efter j utvecklingsår och har egenskaperna $x \in \chi$. Dessutom är vi i behov av att införa nya utvecklingsfaktorer $f_1(x), ..., f_{J-1}(x)$ som tar hänsyn till informationen om faktorerna A, B och C.

Innan vi går in på diverse antaganden som behövs måste ett eventuellt problem som kan uppstå belysas. Det handlar om fallet då det existerar $x \in \chi$ där $C_{i,j-1}(x) = 0$. Huvudsakligen kan det inträffa på grund av två anledningar. Inga skador har rapporterats för skadeår i efter de j första utvecklingsåren givet att vi har egenskaperna x. Fallet kan också vara att skador har rapporterats för egenskaperna x men ännu har inga fordringsbetalningar gjorts efter de j första utvecklingsåren.[5]

3.1 Antaganden

Eftersom vi vill ändra modellen så att den ska kunna ta hänsyn till information $x \in \chi$ måste vi också justera respektive lägga till vissa antaganden. Först vill vi presentera några stående antaganden som kommer att gälla från och med nu, trots att de inte kommer att upprepas kontinuerligt i fortsättningen av arbetet:

- Egenskapsrummet χ är ändligt. I vårt fall är $|\chi| = 120$ möjliga egenskapsvärden.
- kumulativa utbetalningar $C_{i,j}(x) > 0$ för alla $1 \leq i \leq I$, $0 \leq j \leq J$ och $x \in \chi$.

Antar även att de kumulativa fordringsbetalningar $C_{i,j}(x)$ för olika skadeår $1 \leq i \leq I$ eller för olika egenskaper $x \in \chi$ är oberoende. Det existerar utvecklingsfaktorer $f_0(x), ..., f_{J-1}(x) > 0$ och proportionalitetskonstanter $\sigma_0^2, ..., \sigma_{J-1}^2 > 0$ för alla $1 \leq i \leq I$, $1 \leq j \leq J$ och $x \in \chi$ så att:

$$E[C_{i,j}(x)|F_{i+j-1}] = f_{j-1}(x)C_{i,j-1}(x) \tag{3.1}$$

$$Var(C_{i,j}(x)|F_{i+j-1}) = \sigma_{j-1}^2 C_{i,j-1}(x) \tag{3.2}$$

Detta överensstämmer med den klassiska chain ladder-metoden där F_{i+j-1} definieras som känd information, det vill säga den övre skadetriangeln. Dessutom är det viktigt att poängtera att anledningen till att (3.1) och (3.2) ser ut som de gör beror på antagandet gällande att $C_{i,j}(x) > 0$. För om datan som är av intresse i det här arbetet inte uppfyller det är vi i behov av använda någon annan skattningsmetod eller hantera problemet genom att aggregera vissa klasser inom

de olika faktorerna. Under antagandena ovan givet att $I > J$ samt att $i + j > I$ så har vi även att:

$$E[C_{i,J}|F_I] = C_{i,I-i}(x) \prod_{j=I-1}^{J-1} f_j(x) \tag{3.3}$$

3.2 Modellera med neuronnät

Vi vill modellera utvecklingsfaktorerna som ges i formel (3.1). Det kommer att göras med flerlagersperceptroner vars inmatning kommer att vara $x \in \chi$ där det gäller att χ har en dimension av storlek d. Nätverket har ett dolt lager bestående av q neuroner och transferfunktionen $r(t)$ är en hyperbolisk tangent. Utmatningen $z_k(x)$ från det dolda lagret ges av

$$x \to z_k(x) = \tanh\left(w_{k,0} + \sum_{l=1}^{d} w_{k,l} x_l\right) \tag{3.4}$$

Avslutningsvis består utmatningslagret av en neuron som har transerferfunktionen $r(t) = t$ vilket resulterar i att nätverkets utmatning som beskriver en utvecklingsfaktor ges av:

$$x \to \log f_{j-1} = \beta_0 + \sum_{k=1}^{q} \beta_k z_k \tag{3.5}$$

Det innebär att det kommer att existera ett nätverk för varje utvecklingsperiod $1 \leq j \leq J$ som kan ha olika många neuroner q i det dolda lagret. Dessutom kommer två liknande metoder att användas för att träna neuronnäten.

Metod A: Den första metoden kallas för metod A. Den utgår från att responsen $Y_{i,j}(x)$ som används vid träningen av ett nätverk är lika med $C_{i,j}(x)/\sqrt{C_{i,j-1}(x)}$. Det är av intresse att definiera förlustfunktionen L_j^A. Ett naturligt val under våra modellantaganden är det så kallade viktade summerade kvadratfelet:

$$L_j^A = \frac{1}{\sigma_{j-1}^2} \sum_{i=1}^{I-j} \sum_{x} \left(\frac{C_{i,j}(x)}{\sqrt{C_{i,j-1}(x)}} - f_{j-1}(x)\sqrt{C_{i,j-1}(x)} \right)^2 \tag{3.6}$$

Det leder till att motsvarande regressionsfunktion kan utläsas enligt:

$$x \to f_{j-1}^A(x)\sqrt{C_{i,j-1}(x)} = \exp\left(\beta_0 + \sum_{i=1}^{q} \beta_k z_k(x) + \frac{1}{2}\log C_{i,j-1}(x)\right) \tag{3.7}$$

Värdena för $\frac{1}{2}\log C_{i,j-1}(x)$ hanteras som offset.[5] Målet med bakåtpropagationen är att minimerar L_J^A ovan steg för steg givet ett värde för q. Det görs för

att vi ska hitta den optimala nätverksparametern $\alpha = (\beta_0, ..., \beta_q, w_{1,0}, ..., w_{q,d})'$. Således anser vi att det kan vara lämpligt att visa hur gradienten kan beräknas. Kedjeregeln, som används i exempel 1, ger oss att:

$$\begin{aligned}
\bigtriangledown_\alpha L_j^A(\alpha) &= \bigtriangledown_\alpha \frac{1}{\sigma_{j-1}^2} \sum_{i=1}^{I-j} \sum_x \left(\frac{C_{i,j}(x)}{\sqrt{C_{i,j-1}(x)}} - f_{j-1}(x)\sqrt{C_{i,j-1}(x)} \right)^2 \\
&= \frac{-2}{\sigma_{j-1}^2} \sum_{i=1}^{I-j} \sum_x C_{i,j-1}(x) \left(\frac{C_{i,j}(x)}{C_{i,j-1}(x)} - f_{j-1}(x) \right) \bigtriangledown_\alpha f_{j-1}(x) \\
\bigtriangledown_\alpha f_{j-1}(x) &= f_{j-1}(x) \bigtriangledown_\alpha (\beta_0 + \sum_{k=1}^{q} \beta_k z_k(x))
\end{aligned}$$

Väljer att sätta $\mu(x)$ lika med $\beta_0 + \sum_{k=1}^{q} \beta_k z_k(x)$ vilket för $k = 0, ..., q$ och $Z_0 = 1$ ger oss att

$$\frac{\partial \mu(x)}{\partial \beta_k} = z_k(x), \quad \frac{\partial \mu(x)}{\partial w_{k,l}} = \beta_k(1 - z_k(x)^2)x_l$$

där $x_0 = 1$. Det innebär att optimeringsproblemet lösas med bakåtpropagation.[5]

Metod B: Den andra metoden väljer vi att kalla för metod B. Responsen som används vid inlärningsalgoritmen är, i det här fallet, lika med $C_{i,j}(x)/C_{i,j-1}(x)$ vilket innebär att följande förlustfunktion L_j^B är lämplig att använda:

$$L_j^B = \frac{1}{\sigma_{j-1}^2} \sum_{i=1}^{I-j} \sum_x \left(\frac{C_{i,j}(x)}{C_{i,j-1}(x)} - f_{j-1}(x) \right)^2 \tag{3.8}$$

motsvarande regressionsfunktion ges av:

$$x \to f_{j-1}^B(x) = \exp\left(\beta_0 + \sum_{i=1}^{q} \beta_k z_k(x) \right) \tag{3.9}$$

Efter detta vill vi, i likhet med metod A, visa hur gradienten kan beräknas med kedjeregeln.

$$\begin{aligned}
\bigtriangledown_\alpha L_j^A(\alpha) &= \bigtriangledown_\alpha \frac{1}{\sigma_{j-1}^2} \sum_{i=1}^{I-j} \sum_x \left(\frac{C_{i,j}(x)}{C_{i,j-1}(x)} - f_{j-1}(x) \right)^2 \\
&= \frac{-2}{\sigma_{j-1}^2} \sum_{i=1}^{I-j} \sum_x \left(\frac{C_{i,j}(x)}{C_{i,j-1}(x)} - f_{j-1}(x) \right) \bigtriangledown_\alpha f_{j-1}(x)
\end{aligned}$$

Beräkningen fortsätter på samma sätt som för metod A, vilket återigen betyder att bakåtpropagationen är lämpligt.

Jämförelse mellan A och B: Anledningen till att responsen som används i metod A är lika med $C_{i,j}(x)/\sqrt{C_i, j-1}(x)$ beror på att om vi utgår från att vår data uppfyller chain ladder antagandena (2.1) och (2.2) så gäller det att

$$C_{i,j+1} = f_j C_{i,j} + \sigma_j \sqrt{C_{i,j}} \epsilon_{i,j+1} \tag{3.10}$$

där $\epsilon_{i,j+1}$ har väntevärde noll och varians lika med ett, samt oberoende av $C_{i,j}$. Responsen i metod A kommer att leda till att vi får en skattning vars varians inte kommer att bero på storleken av de kumulativa utbetalningarna C_i, j enligt

$$\frac{C_{i,j+1}}{\sqrt{C_{i,j}}} = f_j \sqrt{C_{i,j}} + \sigma_j \epsilon_{i,j+1} \tag{3.11}$$

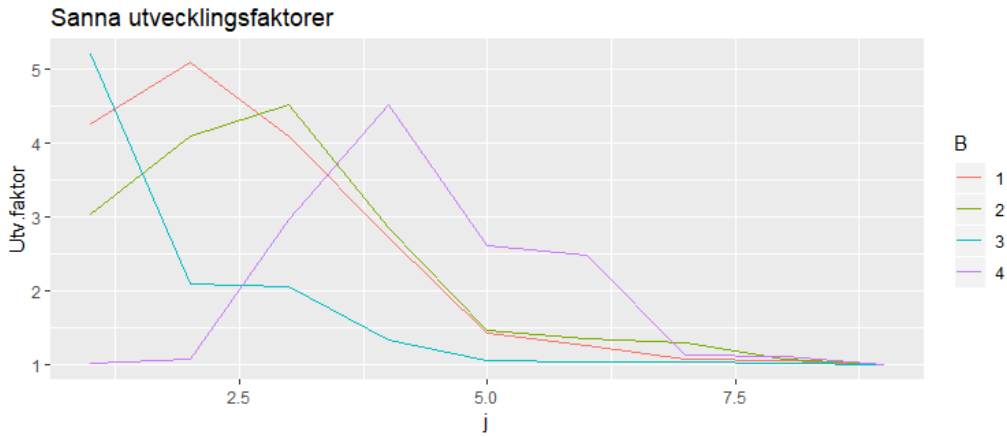
Det är inte är fallet då responsen $Y_{i,j}(x)$ är lika med $C_{i,j}(x)/C_{i,j-1}(x)$ eftersom det leder till att

$$\frac{C_{i,j+1}}{C_{i,j}} = f_j + \frac{\sigma_j \epsilon_{i,j+1}}{\sqrt{C_{i,j}}} \tag{3.12}$$

Analysen görs därav för två stycken olika datamängder som beskrivs i avsnittet datasimulering nedan.[7]

4. Datasimulering

Analysen kommer att ske för två stycken datamängder vid namn data I och data II. De skiljer sig ifrån varandra när det kommer till hur vi simulerar fram respektive utvecklingsår j. Båda består av information om $C_{i,j}(x)$ för alla i, j och $x \in \chi$. Det är även bestämt att $C_{i,1}(x)$ är lognormalfördelade med väntevärdet 1808.042 och standardavikelsen 2370.041. Dessutom har ingen större vikt lagts på hur de sanna värdena för $f_j(x)$ skulle bestämmas. När de togs fram var målet endast att storleken på $f_j(x)$ var rimlig med avseende på vad vi visste om $f_{j-1}(x), f_{j-2}(x), ..., f_1(x)$. Det vill säga att samtliga $f_1(x), ..., f_9(x)$ verkar se ut som en funktion över utvecklingsåren som når ett maximum och därefter går den mot noll vilket illustreras i Figur 5 nedan.



Figur 5: Graferna illustrear de sanna utvecklingsfaktorerna $f_j(x)$ när vi fixerat faktor A respektive faktor C till klass 1 respektive klass 7.

Data I: Den första datamängden väljer vi att kalla för data I. I det fallet har vi simulerat fram så kallad chain ladder data och tanken med det är att den ska överensstämma med metod A.[7] Vi har fått fram datan genom att simulera fram respektive utvecklingsår enligt:

$$C_{i,j+1}(x) = f_j(x)C_{i,j}(x) + \sigma_j\sqrt{C_{i,j}(x)}e_{i,j+1}(x) \tag{4.1}$$

$$\begin{aligned}
&e_{i,j+1}(x) = \exp\left(\mu(W_{i,j}(x)) + \sigma(W_{i,j}(x))Z_{i,j}\right) - W_{i,j}(x)\\
&\sigma(W_{i,j}(x)) = \sqrt{log(1 + W_{i,j}^{-2}(x))}\\
&\mu(W_{i,j}(x)) = \log(W_{i,j}(x)) - \sigma^2(W_{i,j}(x))/2
\end{aligned}$$

Vi har att $W_{i,j}(x) = f_j(x)C_{i,j}^{1/2}/\sigma_j$ samt att $Z_{i,j}$ är standardnormalfördelade. Dess-utom har vi använt att $\sigma_j = 10$ för alla j.

Data II: För data II har vi valt att simulera fram ett utvecklingsår med följande formel

$$C_{i,j+1}(x) = f_j(x)C_{i,j}(x) + \sigma_j C_{i,j}(x)\epsilon_{i,j+1} \tag{4.2}$$

Vi har att $\epsilon_{i,j+1} = Z_{i,j+1} - k$ samtidigt som $Z_{i,j+1} \sim \text{lognormal}(\mu = 0, \sigma^2 = 0.4812118)$ och $k = E[Z_{i,j+1}] = \exp(\mu + \frac{\sigma^2}{2})$. Således kommer det för metod B gälla att variansen för skattningen av de kumulativa utbetalningarna $C_{i,j+1}(x)$ inte beror på något tidigare känt värde för $C_{i,j}(x)$. Eftersom det kommer att resultera i att

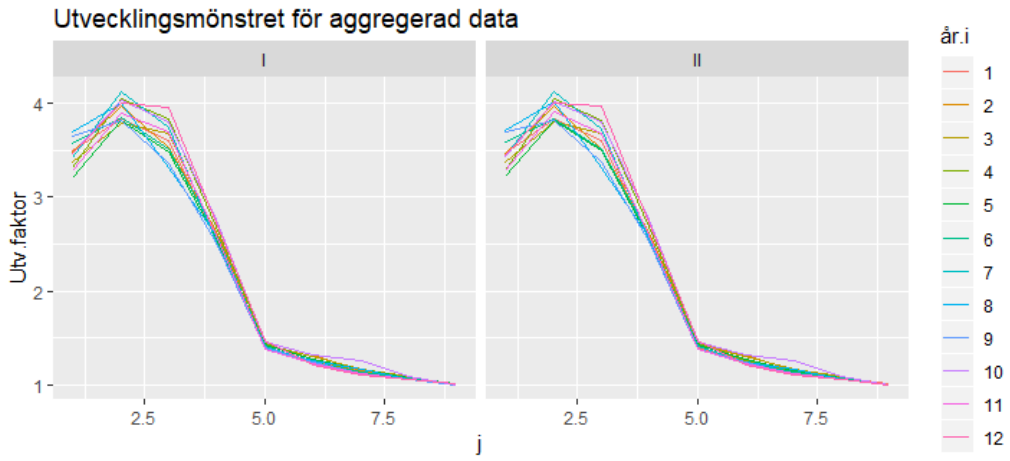
$$\frac{C_{i,j+1}(x)}{C_{i,j}(x)} = f_j(x) + \sigma_j\epsilon_{i,j+1} \tag{4.3}$$

Vill även förtydliga att både data I och data II kommer att användas för både metod A och metod B. Slutligen, innan vi lämnar själva datasimuleringsdelen, kan det vara lämpligt att illustrera en tabell och figurer över den simulerade datan för att möjligtvis öka förståelsen för dess uppbyggnad.

A	B	C	år.i	c1	c2	c3	c4	c5
1	1	1	1	3167.70701	16653.4143	83415.159	273998.366	616548.52
1	1	1	2	784.06997	4287.8618	21203.193	69511.045	153499.18
1	1	1	3	1268.13705	6736.6036	33125.827	109539.222	238559.52
1	1	1	4	1626.32636	8919.9151	44656.692	144329.242	325877.99
1	1	1	5	5795.98507	30138.8656	151360.574	492341.549	1095070.77
1	1	1	6	775.17949	4489.8336	22582.475	74089.499	166329.94
1	1	1	7	1090.73472	5849.4612	29177.823	91257.182	206025.57
1	1	1	8	1077.59515	5353.2277	27099.596	88762.266	199697.44
1	1	1	9	863.78021	4175.4598	19754.542	65952.141	145876.06
1	1	1	10	5830.11082	31676.0988	152845.175	495445.863	1102232.43
1	1	1	11	1070.70538	5285.1889	26179.374	86486.656	191572.75
1	1	1	12	960.05614	5608.4978	27559.809	93467.340	208874.93
1	1	2	1	1323.72847	7051.0051	27816.115	85126.006	241139.08
1	1	2	2	2974.07954	15201.1493	58448.505	184009.568	517869.93
1	1	2	3	292.75045	1529.4967	5777.681	17685.317	50251.93
1	1	2	4	1019.89632	5584.0325	22853.436	74456.435	205195.10
1	1	2	5	109.44151	665.8864	2395.474	9318.579	24047.30
1	1	2	6	273.62646	1155.3291	4106.832	12873.104	36914.09
1	1	2	7	77.24491	369.3225	1455.123	4199.158	12754.40
1	1	2	8	1043.09162	4896.8816	18330.897	58083.184	166539.10

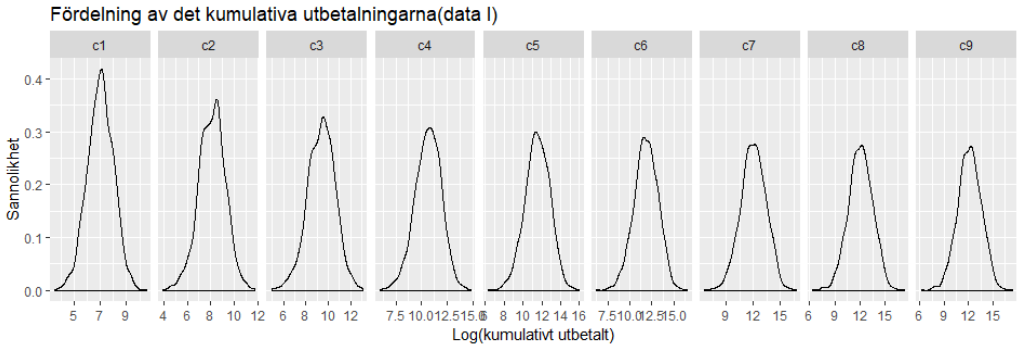
Tabell 4: En tabell som illustrerar hur den simulerade datan är uppbyggd. Däremot har vi valt att endast visa de fem första av de tio utvecklingsåren.(Data I)

Ser att det finns information om de kumulativa utbetakningarna $C_{i,j}(x)$ för varje egenskapskombination respektive skadeår. Detsamma gäller för data II. Efter det undersöks utvecklingsmönstret för den aggregerade datan över samtliga faktorer och klasser eftersom ett av antagandena i chain ladder modellen är att utvecklingsfaktorerna inte ska bero på skadeåren.



Figur 6: Utvecklingsmönstret för de kumulativa utbetalningarna när vi aggregerat den sanna, simulerade datan, över samtliga faktorer och klasser.

Precis som vi förväntade oss uppfylls det antaganden, utifrån vad man se i Figur 6. Det verkar som att skillnaderna mellan skadeåren är relativt små vilket pekar mot att en låg varians har använts vid datasimuleringen av båda datamängderna. Avslutningsvis, för detta avsnitt, tittar vi på fördelningen av de kumulativa utbetalningarna.



Figur 7: Fördelningen för de kumulativa utbetalningarna när vi undersöker samtliga egenskapskombinationer och skadeår.(Data I)

Från Figur 7 och Figur A1 ser vi att täthetsfunktionerna för respektive utvecklingsår verkar vara lognormalfördelade, eftersom de ser ut att vara normalfördelade samtigt som värdena på x-axeln är logaritmerade.

5. Simulering av nätverk

Simuleringen görs i programmet R och koden är skriven i språket R.[8] Det är två programpaket som är av väsentlig betydelse. Det ena heter keras och det används för att modellera utvecklingsfaktorerna enligt metod A.[9] Andra paketet heter neuralnet, vilket enbart används för metod B.[10] På grund av de inbyggda funktionernas begränsningar är vi i behov av att ändra förlustfunktionerna L_j^A och L_j^B som definierades i (3.6) respektive (3.8). Skillnaden är att vi kommer att använda oss av de oviktade summerade kvadratfelen.

$$L_j^{A0} = \sigma_{j-1}^2 L_j^A = \sum_{i=1}^{I-j} \sum_x \left(\frac{C_{i,j}(x)}{\sqrt{C_{i,j-1}(x)}} - f_{j-1}(x)\sqrt{C_{i,j-1}(x)} \right)^2 \tag{5.1}$$

$$L_j^{B0} = \sigma_{j-1}^2 L_j^B = \sum_{i=1}^{I-j} \sum_x \left(\frac{C_{i,j}(x)}{C_{i,j-1}(x)} - f_{j-1}(x) \right)^2 \tag{5.2}$$

Det är inget problem eftersom det explicita valet av σ_{j-1}^2 inte påverkar kalibreringen vid bakåtpropagationen.

För metod A har vi valt att använda oss av en bakåtpropagation vid namn RMSprop samt har vi valt att alltid genomföra 100 epochs vid inlärningsalgoritmen av respektive neuronnät. För metod B tillämpas elastisk bakåtpropagation som upprepas tills att minskningen av förlustfunktionens värde är mindre än 0.01. Samtliga neuronnät kommer att bestå utav $q = 15$ neuroner i sitt dolda lager. Anledningen till att samma funktioner inte har använts för båda metoderna vilket beror på att vi inte lyckades inkludera offsetparametrar i neuralnetpakets funktioner. Man kan däremot fråga sig varför vi då inte använder kerasfunktionerna för metod B. Kort sagt beror det på att de är ganska tidsineffektivt jämfört med neuralnet paketets funktioner.

Vi vill även förtydliga att storleken på träningsdatan kommer minskas när utvecklingsår j blir större. Exempelvis kommer inlärningsalgoritmen för neuronnätet kopplat till utvecklingsfaktorerna $f_1(x)$ ha elva stycken responsen värden $Y_{i,j}(x)$, som är lika med $C_{i,j}(x)/\sqrt{C_{i,j-1}(x)}$ eller $C_{i,j}(x)/C_{i,j-1}(x)$, för varje egenskapskombination. I fallet när neuronnätet för $f_8(x)$ ska tränas så används endast fyra responsvärden $Y_{i,j}(x)$ för respektive $x \in \chi$.

5.1 Förbehandling av egenskaperna

För att metoden gradientnedstigning, som används i bakåtpropagationen, ska resultera i en rimlig regressionsfunktion är man ibland i behov av att förbehandla

komponenterna i egenskaperna $x \in \chi$. Problemet som måste lösas är att samtliga faktorer består av kategoriska komponenter. Det vill säga att det inte finns något tydligt avstånd mellan exempelvis klass ett och klass två inom faktor A. Jämfört med om vi förslagsvis hade ålder där det är lika stort avstånd mellan 24 och 25 som det är för 56 till 57. Således behöver vi ändra de kategoriska komponenterna till numeriska innan nätverksmodelleringen kan påbörjars. Det görs med hjälp av dummykodning vilket innebär att en faktor bestående av r klasser ersätts av en $(r-1)$-dimensionell binär vektor som indikerar respektive komponent. Exempelvis illustrerar Tabell 5 nedan hur det ser ut i fallet för faktor A.

x_1	Dummy x_1^*	
A1	0	0
A2	1	0
A3	0	1

Tabell 5: Dummykodning för faktor $A \in \{1, 2, 3\}$

Från Tabell 5 kan vi exempelvis se att klass ett för faktor A används som referensklassen. Typiskt brukar man välja den klassen som består av flest observationer. Vår data består av lika många observationer för samtliga egenskapskombinationer, vi har valt att alltid låta klass ett för respektive faktor vara referensklassen. Resultatet av detta blir att de tre egenskapsfaktorerna ersätts av en egenskapsvektor vars dimension är lika med $2 + 3 + 9 = 14$. Det är även av intresse att nämna att kontinuerliga egenskapsfaktorer också kan behöva förbehandlas, eftersom dess värden behöver ligga på en jämförbar skala för att bakåtpropagationen ska leda till en rimlig kalibrering. En möjlig lösning är därför att tillämpa den så kallade MinMaxScalern som ges av formel (5.3) nedan.

$$x_l \to x_l^* = 2\frac{x_l - \min x_l}{\max x_l - \min x_l} - 1 \tag{5.3}$$

6. Resultat

I detta avsnitt kommer olika tabeller respektive figurer som illustrerar resultaten av simuleringen att presenteras. För alla figurer och tabeller i resultatdelen har metod A och metod B upprepats 100 gånger vardera. Det är i de flesta fallen medelvärdet av dessa 100 försök som visas om inget annat sägs eller om vi undersöker fördelningar. Att simuleringen upprepas 100 gånger innebär att varje neuronnät har tränats 100 gånger för båda metoderna. Det betyder att varje utvecklingsfaktor $f_j(x)$ har estimerats 100 gånger med metod A och 100 gånger med metod B.

Först är det lämpligt att undersöka hur väl metod A, metod B samt den klassiska chain ladder-metoden skattar ultimovärdet för samtliga skadeår $i = 4, 5, ..., 12$.

		CL-metoden		NN-metod A		NN-metod B	
Skadeår	Sann ultimo	Ultimo	Standard error	Ultimo	Standard error	Ultimo	Standard error
4	67951634	67897960	93546	68086963	723113	67866979	706831
5	47272848	47591579	833780	47672227	1021091	47250022	851658
6	56091689	57256869	1247007	56073618	1140849	56105792	1036813
7	49126513	52868127	1725474	51056022	1958825	48882705	1168928
8	60103895	62380311	2084331	62139525	2699255	59698868	1453370
9	42820181	48897615	2083120	45601311	4316297	42845403	1393946
10	95364778	71587698	4201131	89411407	4662519	94739966	2184161
11	50093426	54517248	4017377	51617077	7093206	50064476	2277020
12	79335812	71975293	5548105	71737145	5076111	78919879	5094126
Total	548160776	534972700	10594256	543395295	11463488	546374089	6859459

Tabell 6: En tabell som illustrerar ultimobeloppen för respektive skadeår när vi har aggregerat data I över alla faktorer och dess klasser. Dessutom gäller det att simuleringen har upprepats 100 gånger vid inlärningsprocessen av neuronnäten för både metod A och metod B.

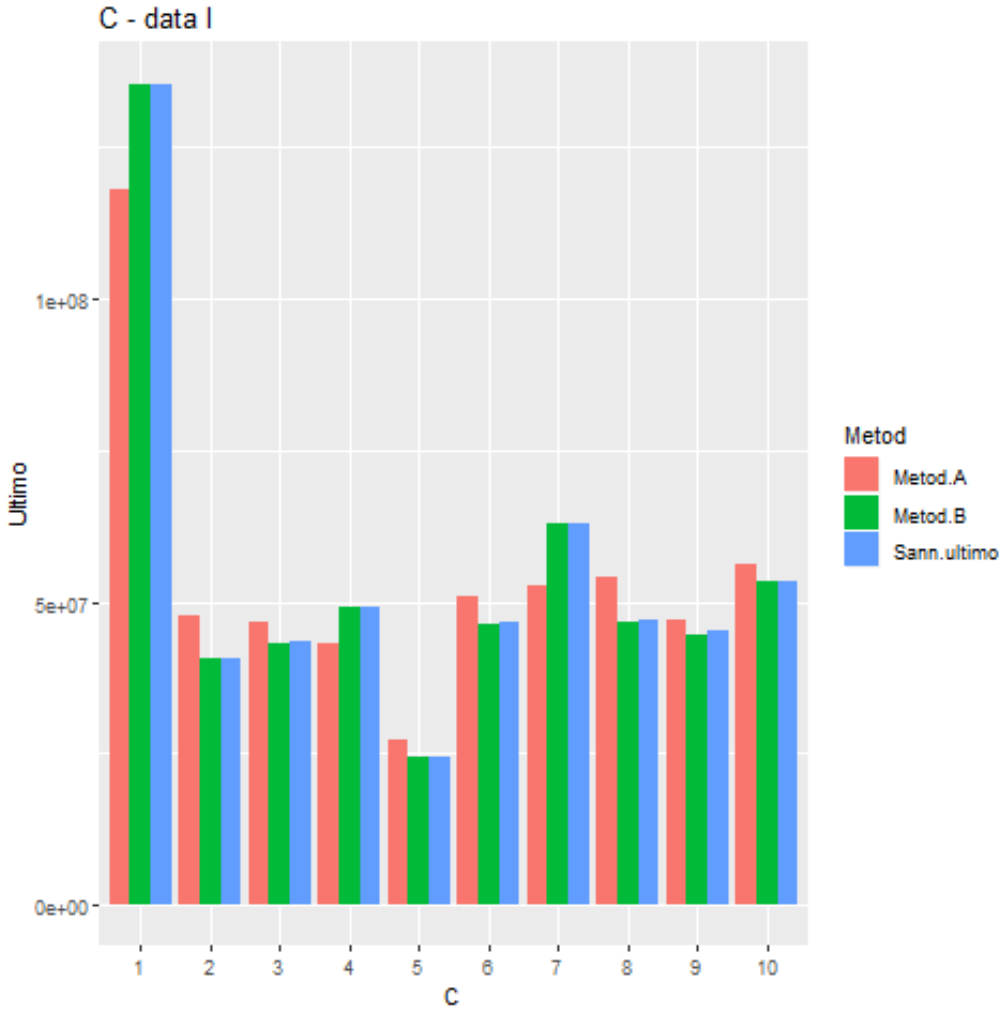
		CL-metoden		NN-metod A		NN-metod B	
Skadeår	Sann ultimo	Ultimo	Standard error	Ultimo	Standard error	Ultimo	Standard error
4	66616543	66728027	109612	66869201	411059	66712186	361072
5	46883316	47370921	913455	47446559	974683	46935086	351650
6	56401259	57196530	1186082	56116184	508834	56376037	503854
7	49252071	52827793	1782608	51453647	2954356	49254703	516666
8	59079669	62295098	2007841	62864023	906769	59195752	637402
9	43279580	49358581	2240056	46463531	3537483	43393902	499960
10	93863046	70428549	3831383	89897124	6966478	93954205	1030205
11	49872444	54115738	4127977	51088818	2438248	49917068	574102
12	78362056	71472712	5141427	70562936	15077560	78476424	914845
Total	543609983	531793948	10250001	542762021	16152846	544215363	1982830

Tabell 7: En tabell som illustrerar ultimobeloppen för respektive skadeår när vi har aggregerat data II över alla faktorer och dess klasser. Dessutom gäller det att simuleringen har upprepats 100 gånger vid inlärningsprocessen av neuronnäten ör både metod A och metod B. Vidare gäller det att en viktad chain ladder-metod har har används för att skatta beloppen för kolumnen CL-metoden.

Från Tabell 6 och Tabell 7 kan vi säga att samtliga tre metoder verkar lämpliga för skattning av ultimo men utifrån tabellerna ser det ut som att metod B ger den bästa skattningen av det sanna värdet. Kolumnerna över standard error från Tabell 6 och Tabell 7 pekar också mot motsvarande resultat i och med att lägre värden innebär att skattningarna är mer pålitliga. Det vill säga att det verkar som att metod B är den metoden vars skattningar har minst spridning från det sanna värdet $C_{i,I}$. Detta är oväntat i och med att data I är anpassad efter metod A. En möjlig förklaring är att neuronnäten inte har konvergerat tillräckligt för metod A. Eftersom eventuellt för få epochs har upprepats vid bakåtpropagationen, vilket är något som undersöks senare i resultatdelen.

Intresserar oss nu för att ta reda på hur pass lämpliga metod A respektive metod B är om man fokuserar på faktorerna A, B och C var för sig. Inleder med att göra

ett stapeldiagram som jämför det totala ultimovärdet för skadeår $i = 4, 5, ..., 12$ inom de tio klasserna som finns för faktor C.



Figur 8: Ett stapeldiagram som illustrerar det totala ultimovärdet för skadeår 4-12 inom de tio klasserna som finns för faktor C. De röda staplarna illustrerar det sanna värdet för data I, medan de gröna staplarna är skattningar av resultaten när vi använt oss av ett neuronnät enligt metod A samtidigt som de blå staplarna illusterar motsvarande för metod metod B. Vidare gäller det även att de gröna respektive blå staplarna är medelvärdet av resultaten när vi upprepat simuleringarna 100 gånger.

Från Figur 8 och Figur A2 ser vi att både metod A och metod B verkar ha fångat upp variationerna i den sanna datan relativt bra för både data I och data II. Däremot är det återigen metod B som ser ut att vara mest lämpad för att skatta

de två datamängderna utifrån de givna skadetrianglarna. Efter detta kollar vi på en tabell, för respektive datamängd, som visar hur de två metoderna har estimerat ultimovärdet för respektive skadeår för de tre klasserna inom faktor A.

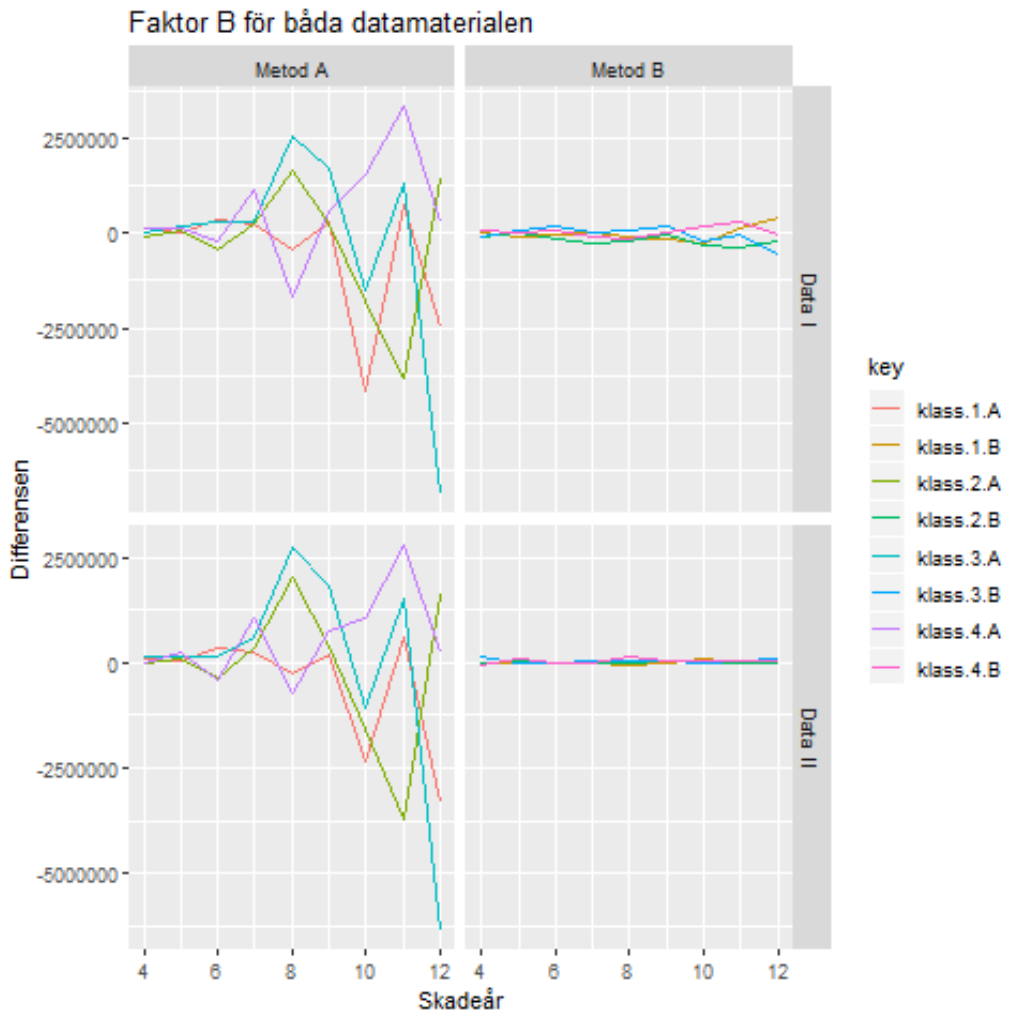
	Klass 1			Klass 2			Klass 3		
År	A	B	Sann	A	B	Sann	A	B	Sann
4	30167043	30082191	30106165	19102179	19095372	19105854	18817741	18689417	18739614
5	19359588	19114696	19031435	11302606	11241421	11326570	17010033	16893905	16914843
6	20712046	20744244	20689887	12895911	13622291	13500906	22465661	21739257	21900895
7	18333232	16618217	16708209	12975348	12566921	12607740	19747442	19697567	19810564
8	23593455	25571710	25611445	18672921	14817895	14789687	19873149	19309263	19702764
9	14519208	14276014	14163666	16267805	13452062	13454445	14814298	15117327	15202070
10	23357666	21743754	21726649	14728704	16288004	16337348	51325037	56708209	57300781
11	13724297	12417468	12389261	18575625	17756303	17730014	19317155	19890705	19974150
12	30856078	35429946	36244910	19463527	19275872	19731263	21417540	24214061	23359639

Tabell 8: En tabell som visar hur ultimo varierar mellan de två olika metoderna som vi använder för inlärning av neuronnäten. När vi endast intresserar oss för hur de skiljer sig åt mellan de tre klasserna inom faktor A för respektive skadeår.(Data I)

	Klass 1			Klass 2			Klass 3		
År	A	B	Sann	A	B	Sann	A	B	Sann
4	29130158	29064336	28921423	19034916	18981336	19021234	18704127	18666514	18673885
5	19346740	19019220	18946480	11056585	11006725	11010616	17043233	16909141	16926220
6	20425187	20429926	20450329	13487985	14396691	14372855	22203012	21549419	21578075
7	18822951	16843067	16843322	12691643	12416638	12446101	19939053	19994998	19962648
8	23935280	25164362	25039577	18799599	14923451	14938952	20129144	19107939	19101139
9	14888727	14279083	14180816	16516059	13702801	13719370	15058744	15412018	15379394
10	23683178	21751079	21747242	14287358	15931342	15940356	51926588	56271785	56175448
11	14069013	12891585	12857751	18129929	17267569	17380152	18889875	19757914	19634540
12	30336379	35090302	34977574	19216111	19385666	19455681	21010446	24000456	23928801

Tabell 9: En tabell som visar hur ultimo varierar mellan de två olika metoderna som vi använder för inlärning av neuronnäten. När vi endast intresserar oss för hur de skiljer sig åt mellan de tre klasserna inom faktor A för respektive skadeår.(Data II)

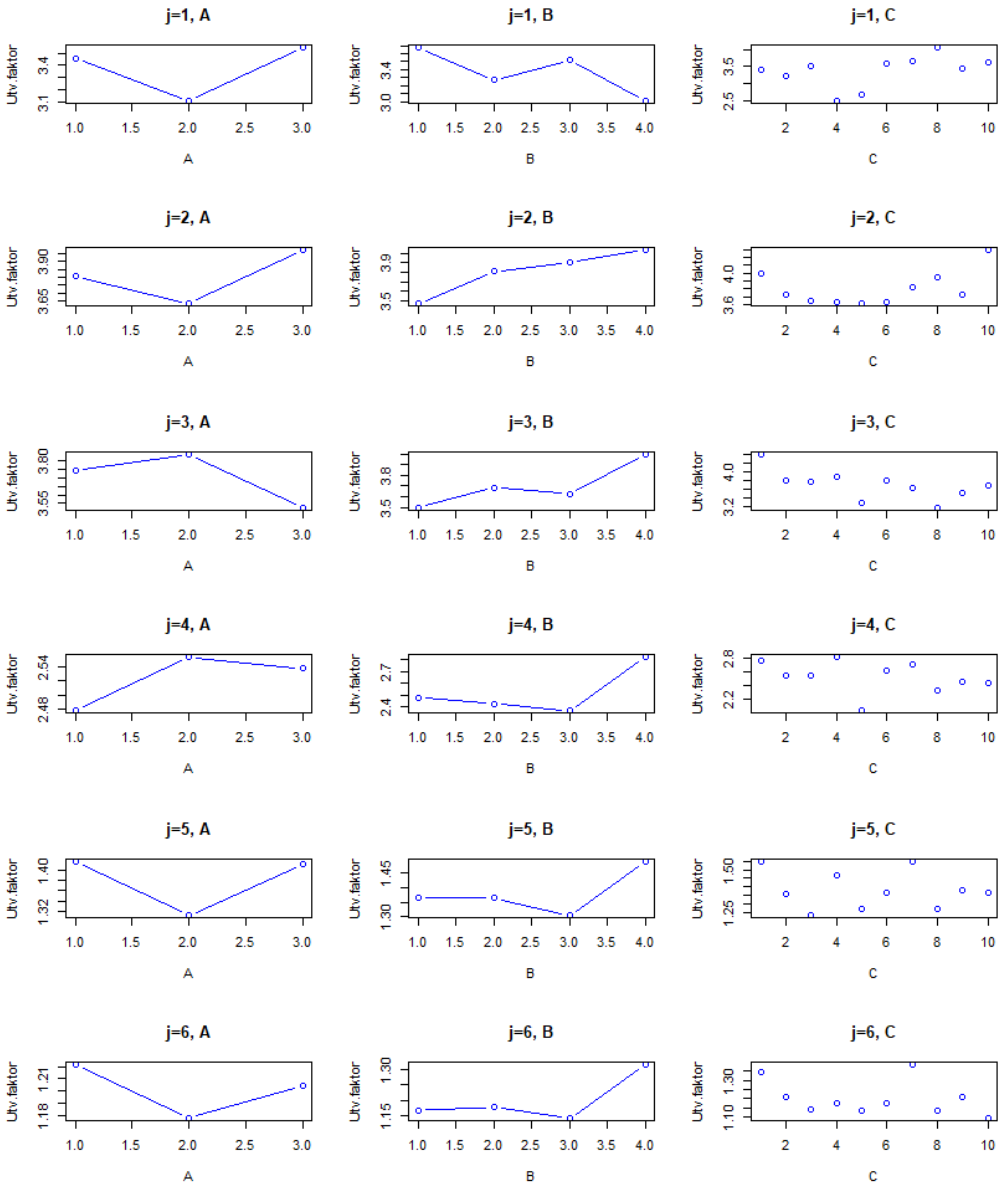
Vi kan i Tabell 8 och Tabell 9 se att båda metoderna verkar lämpliga eftersom dess resultat överensstämmer relativt bra med de sanna värdena. Dessutom ser det återigen ut som att metod B är att föredra framför metod A för både data I och data II. Det som nu återstår innan själva utvecklingsfaktorerna $f_j(x)$ ska undersökas är att illustrera något som visar hur metoderna har skattat klasserna inom faktor B. Vi gör det genom grafer som visar differensen mellan metodernas ultimoskattningar och de sanna ultimovärdena över skadeåren som är av intresse.

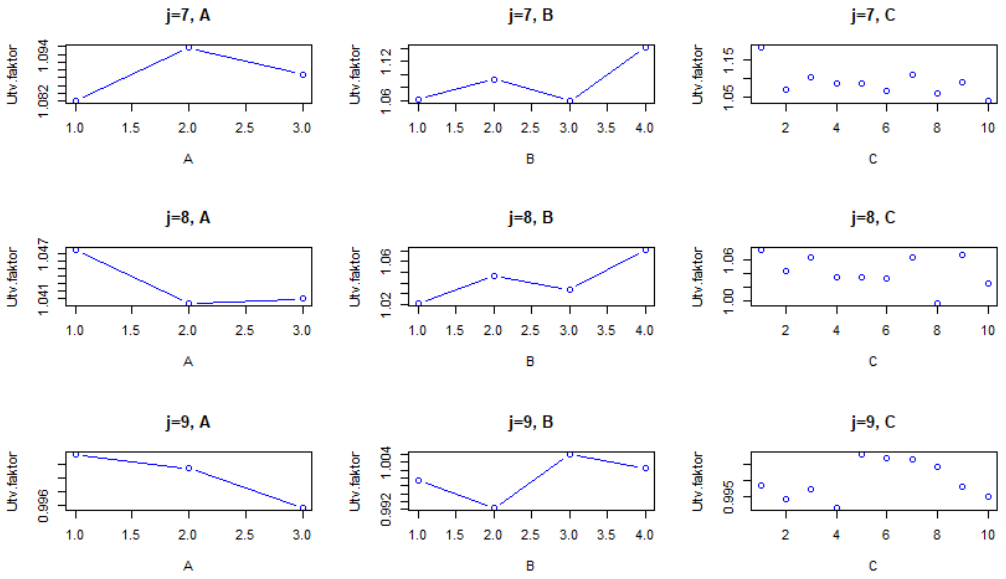


Figur 9: En plott som illustrerar hur det estimerade ultimovärdet skiljer sig från det sanna värdet över skadeåren för respektive klass inom faktor B.

Figur 9 visar tydligt att metod B har skattat ultimovärdena, för respektive skadeår samt för respektive datamängd, bättre än metod A. Utifrån figurerna och tabellerna ovan, ser vi även att metod B har en tendens att skatta data II bättre. Det är vad man kan förvänta sig i och med att data II är anpassad efter metod B. Däremot är det svårt att se detsamma rent visuellt för metod A.

Vi ska nu undersöka de estimerade utvecklingsfaktorerna $\hat{f}_j^A$ och $\hat{f}_j^B$ för båda datamängderna, för att eventuellt kunna upptäcka något mönster. Gör det genom att titta på plottar över medelvärdet av de skattade $f_j(x)$ för de olika klasserna inom respektive faktor för samtliga utvecklingsår. Nedan har vi valt att illustrera detta för metod B och data I medan de övriga resultaten presenteras i appendix.

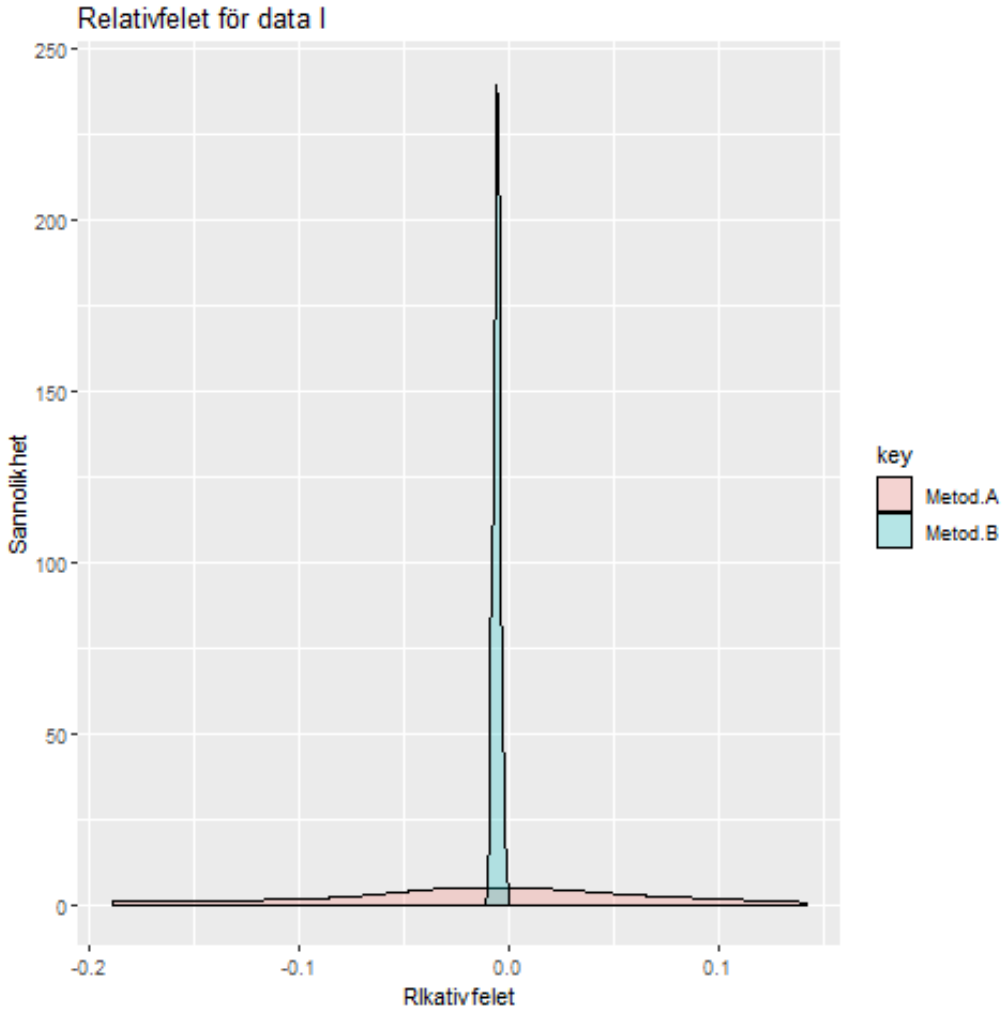

j=1, A
j=1, B
j=1, C
j=2, A
j=2, B
j=2, C
j=3, A
j=3, B
j=3, C
j=4, A
j=4, B
j=4, C
j=5, A
j=5, B
j=5, C
j=6, A
j=6, B
j=6, C
Utv.faktor
A
B
C



Figur 10: De estimerade utvecklingsfaktorerna $\hat{f}_j(\mathrm{x})$ för de enskilda egenskapsfaktorerna A, B respektiva C. I detta fall är det metod B och data I som är av intresse.

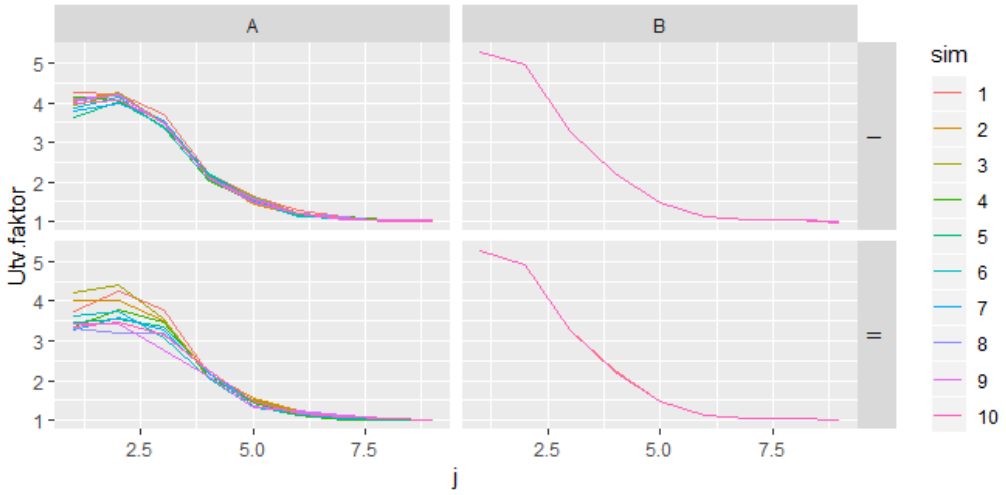
Först kan vi från graferna i Figur 10 se att utvecklingsfaktorerna verkar vara störst för klass ett inom faktor C. Det är en av förklaringarna till resultatet i Figur 8 gällande varför ultimovärdet är mycket större för klass ett inom faktor C. Utöver detta pekar även graferna som beskriver utvecklingsfaktorerna inom faktor B mot att klass fyra borde vara den klassen som har störst reserv inom faktorn i förhållande till de kända utbetalda värdena.

Avslutningsvis vill vi också ta reda på om neuronnäten som fås av metoderna har konvergerat, det vill säga att förlustfunktionen har nått sitt minimum. Ett lämpligt sätt att undersöka det är att kolla på hur skattningarna skiljer sig mellan de etthundra upprepningarna. Börjar med att plotta relativfelets fördelning för den totala reserven. Relativfelet definieras som kvoten av differensen mellan skattad utmatning respektive sann utmatning och det sanna värdet.



Figur 11: Fördelningen av relativfelet för den totala reserven för data I

Från Figur 11 och Figur A6 i appendix kan vi se att väntevärdet är ungefär lika med noll. Fördelningen för metod A har en större spridning, vilket pekar mot att neuronnäten för metod A inte har konvergerat efter 100 epochs. Vidare plottas även utvecklingsfaktorerna $\hat{f}_j(x)$ för $j = 1, 2, ..., 9$ för de tio första simuleringarna när $x = (1, 1, 1)$. Det vill säga klass ett inom faktorerna A, B och C.



Figur 12: Skattningarna av utvecklingsfaktorerna $\hat{f}_j(x)$ när vi fixerat alla faktorer till klass 1 och upprepat respektive metod tio gånger för både data I och data II.

Liksom innan verkar det, utifrån Figur 12, som att neuronnäten i metod B har blivit tillräckligt kalibrerat. Det gäller inte för metod A eftersom vi kan se att skattningarna av utvecklingsfaktorerna skiljer sig relativt mycket mellan simuleringarna.

7. Diskussion

Vi har alltså utökat den klassiska chain ladder-metoden, som används inom reservsättning, så att information om de olika skadorna kan inkluderas. Resultatet av det som tabellerna respektive figurerna illustrerar är kort sagt att både metod A och metod B verkar lämpliga för att estimera diverse ultimovärden för båda datamängderna. Metod B har i det här fallet lett till bättre skattningar jämfört med metod A respektive chain ladder-metoden och rent visuellt verkar det som att metod B estimerar data II bättre än data I. Det är precis vad man kan förvänta sig i och med att data II är anpassad efter metod B. Men är det svårt att se huruvida det gäller för metod A utifrån våra figurer och tabeller. Vidare kan det vara värt att nämna att de två metoderna också har fördelen att de låter oss analysera reserven för individuella skador. Det ger oss möjligheten att upptäcka förändrade portföljblandningar, exempelvis pekar resultatet mot att vi kan förvänta oss en större reserv för skador som tillhör klass ett inom faktor C. Således förväntas det att en stor del av de totala ersättningsbeloppen ska utbetalas i framtiden för den gruppen. Dessutom kan resultaten möjligtvis användas även inom prissättningen av försäkringarna. Fördelen, om man ska jämföra med chain ladder-metoden, är också att skattningarna ser ut att bli bättre för de senaste skadeåren. Det beror på att vi nu har inkluderat mer information eftersom i fallet med chain ladder har man endast information om hur mycket som är utbetalt och inte om det är skador inom exempelvis klass ett för faktor C som vi sett har en relativt stor reserv.

Det verkar som att både metod A och metod B är att föredra framför chain ladder-metoden för de två simulerade datamängderna. Däremot är vi i behov av att diskutera varför metod B genomgående resulterat till bättre skattningar. Trots att exempelvis data I är tänkt att vara anpassad efter metod A. En trolig anledning är att vid inlärningsalgoritmen för metod B har fler epochs upprepats jämfört med metod A som var förinställd på 100 epochs. Således har neuronnäten för metod B blivit mer kalibrerade och därav kunnat fångat upp utvecklingsmönstret bättre. Man kan fråga sig varför inte fler epochs kördes för metod A och svaret på det är kort sagt tidsaspekteten. Eftersom vid simulering av metod A respektive metod B upptäckte vi att funktionerna som användes för metod A var mer tidskrävande än motsvarande för metod B. Möjligtvis hade resultatet blivit annorlunda om datan hade simulerats med större varians eftersom det eventuellt skulle bli svårare för metod B att estimera data I jämfört med metod A och vice versa. Det kan även vara värt att nämna att responsen som användes vid träningen av nätverken för metod A inte kunde användas för funktionerna som metod B använde sig av eftersom de inte kunde inkludera offset. Däremot hade det kanske varit av intresse att undersöka hur metod B skulle estimerat datan om vi endast upprepat 100 epochs på samma sätt som för metod A. För att se om resonemanget ovan är relevant.

Således verkar det som att det finns mycket mer inom detta ämne som kan vara värt att undersöka. Dels skulle det vara intressant att kolla på verklig data och undersöka hur resultaten skulle skilja sig åt. Det finns också betydelse i att undersöka data där vi har olika storlekar på J för $x \in \chi$ eftersom i det här fallet var $J = 10$ för samtliga egenskapskombinationer. Vidare kan kontinuerliga faktorer även inkluderas i analysen dock vill vi påpeka att det gör funktionerna relativt tidsineffektiva vilket var anledningen till att sådana uteslöts i detta arbete. Avslutningsvis kan vi summera detta till att metoderna verkar användbara förutsatt att man har mycket datorkraft.

Referenser

[1] Mack, T. (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. ASTIN Bulletin 23/2, 213-225 (Hämtad 2019-05-13)

[2] Nielsen, M. (2017). Neural Networks and Deep Learning. Onlinebok tillgänglig på http://neuralnetworksanddeeplearning.com (Hämtad 2019-05-13)

[3] Silva,I. N.da et al.Artificial Neural Networks-A Practical Course.Springer, 2017. Tillgänglig på: https://www.springer.com/us/book/9783319431611 (Hämtad 2019-05-13)

[4] https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/ (Hämtad 2019-05-14)

[5] Wüthrich, M. 2018b. "Neural networks applied to chain–ladder reserving", European Actuarial Journal 8(2):407-436. (Hämtad 2019-05-13)

[6] https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers- (Hämtad 2019-05-13)

[7] Lindholm, M., Lindskog, F., Wahl, F.(2018). Estimation of conditional mean squared error of prediction for claims reserving. Tillgänglig på SSRN: http:// ssrn.com/abstract=3302576 (Hämtad 2019-05-19)

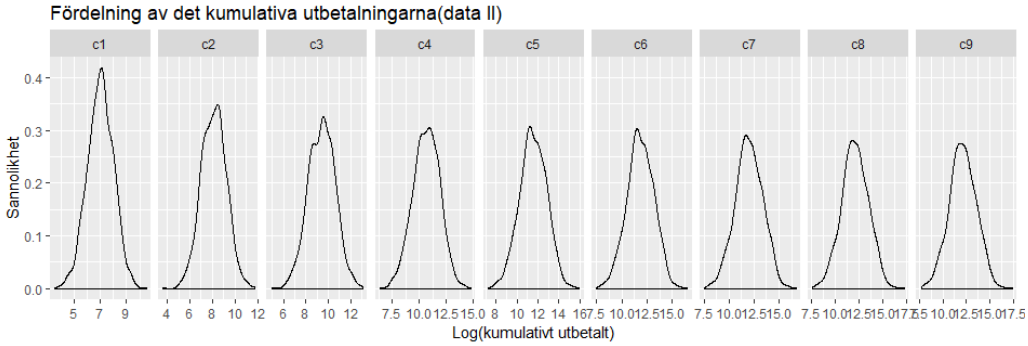
[8] R Core Team. 2019. The R Project for Statistical Computing http: //www.R-project.org/ (Hämtad 2019-05-13)

[9] R Core Team. 2019. R interface to Keras http: //keras.rstudio.com/ (Hämtad 2019-05-13)

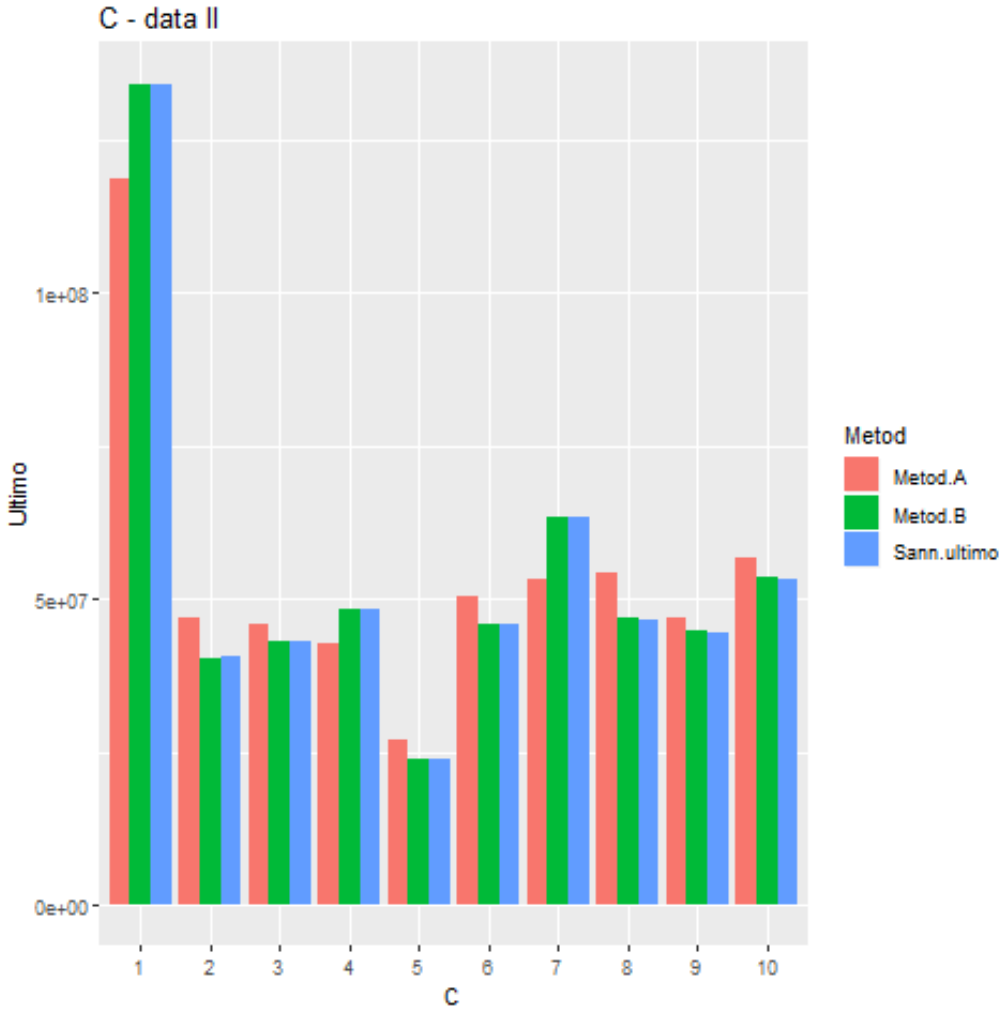
[10] R Core Team. 2019. Package "neuralnet"
http: //cran.r-project.org/web/packages/neuralnet/neuralnet.pdf (Hämtad 2019-05-13)

[11] Lindholm, M., Lindskog, F.(2018). LECTURE NOTES ON BASIC INSURANCE. Tillgänglig på: https://kurser.math.su.se/pluginfile.php/66976/mod_resource/content/9/lectureNotes190506.pdf (Hämtad 2019-06-03)

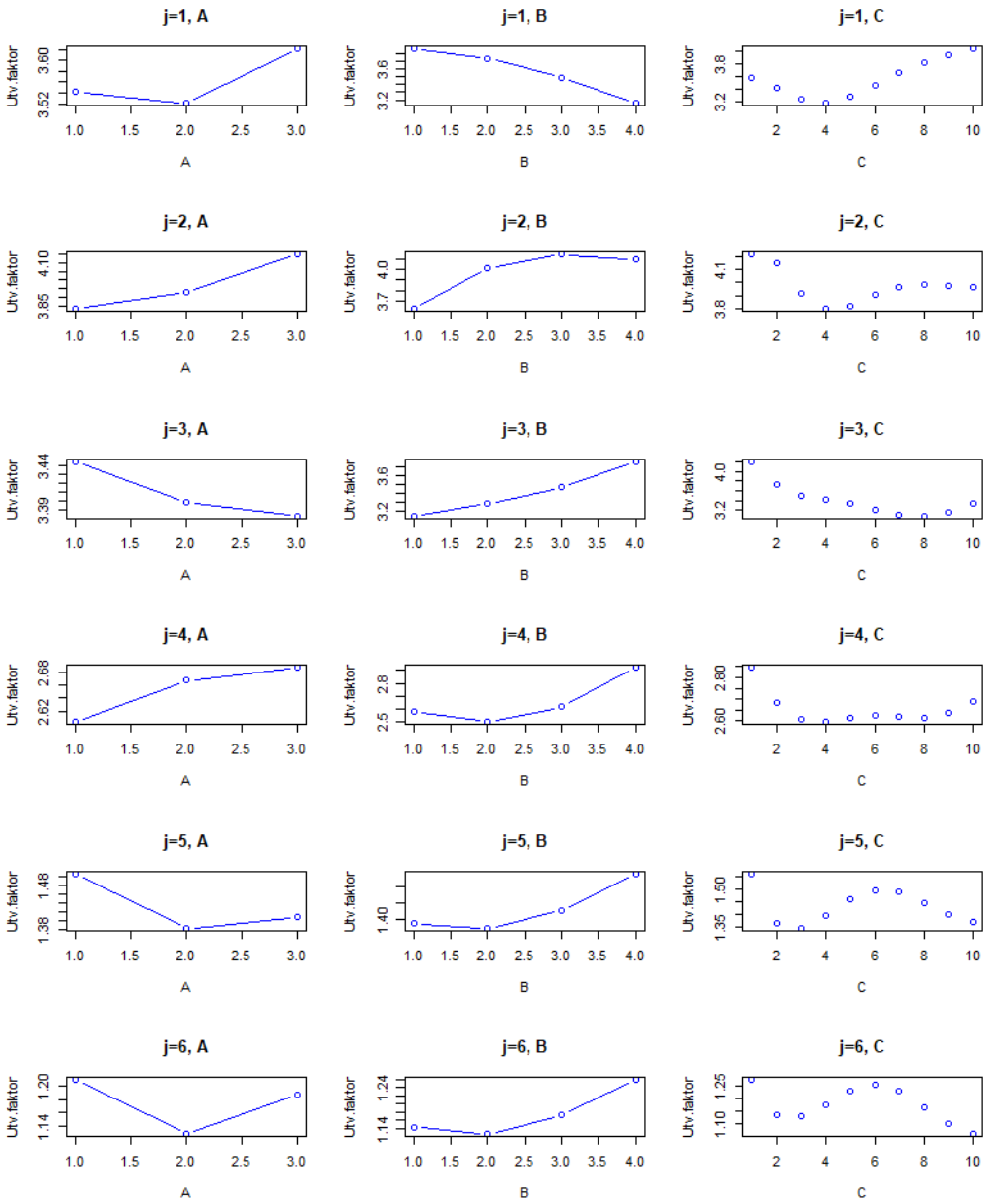
Appendix

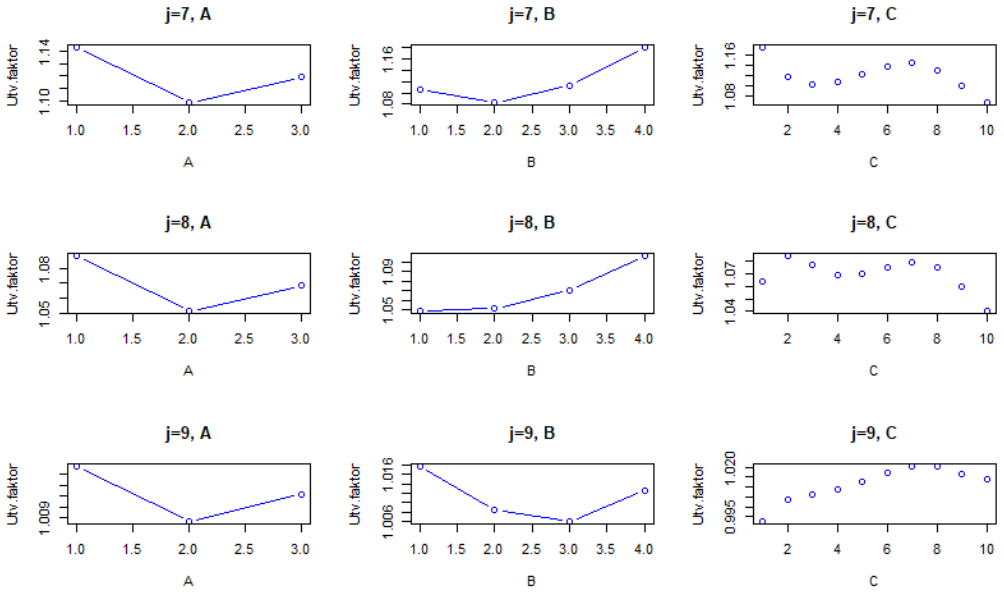


Figur A1: Fördelningen för de kumulativa utbetalningarna när vi undersöker samtliga egenskapskombinationer och skadeår.(Data II)

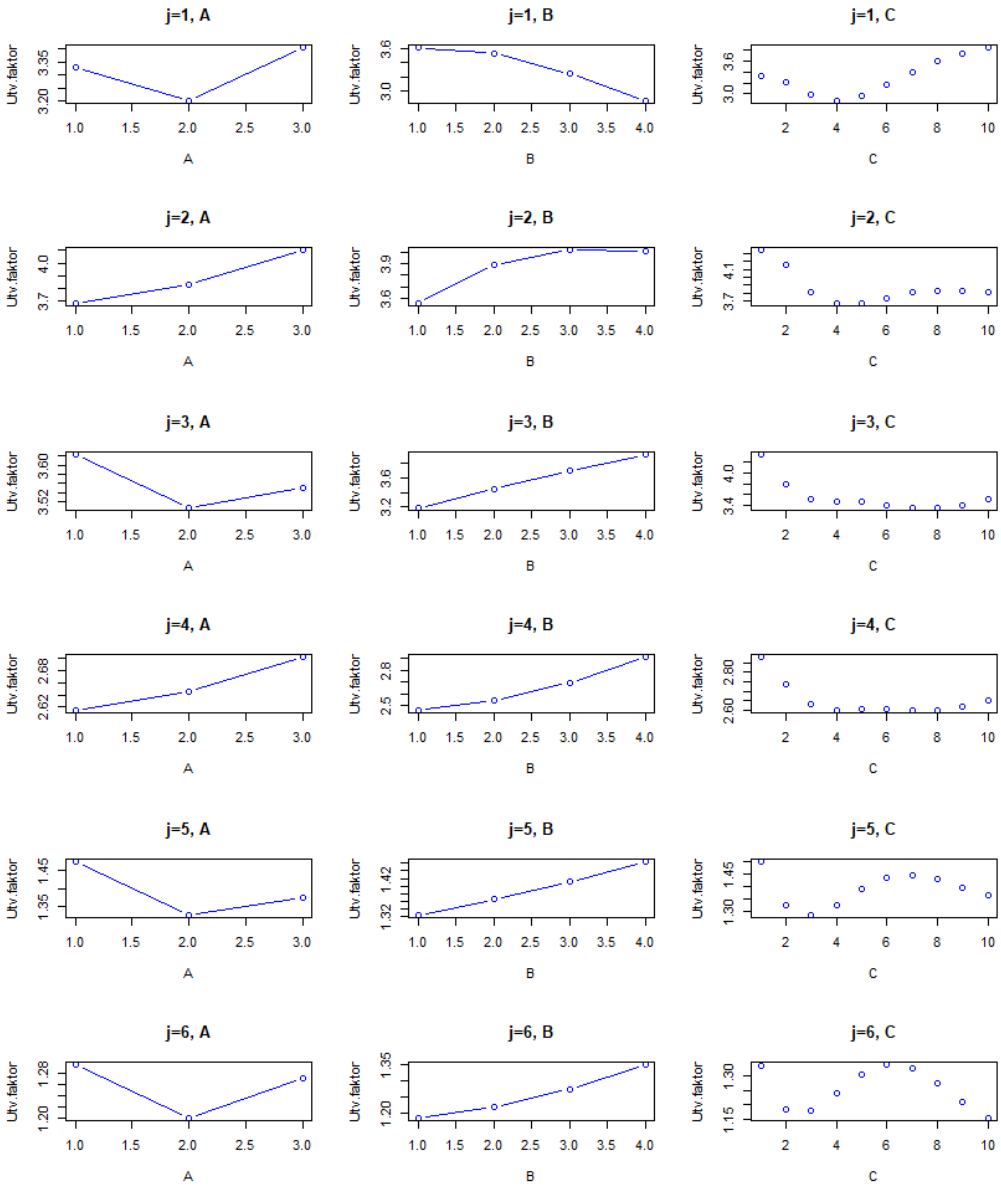


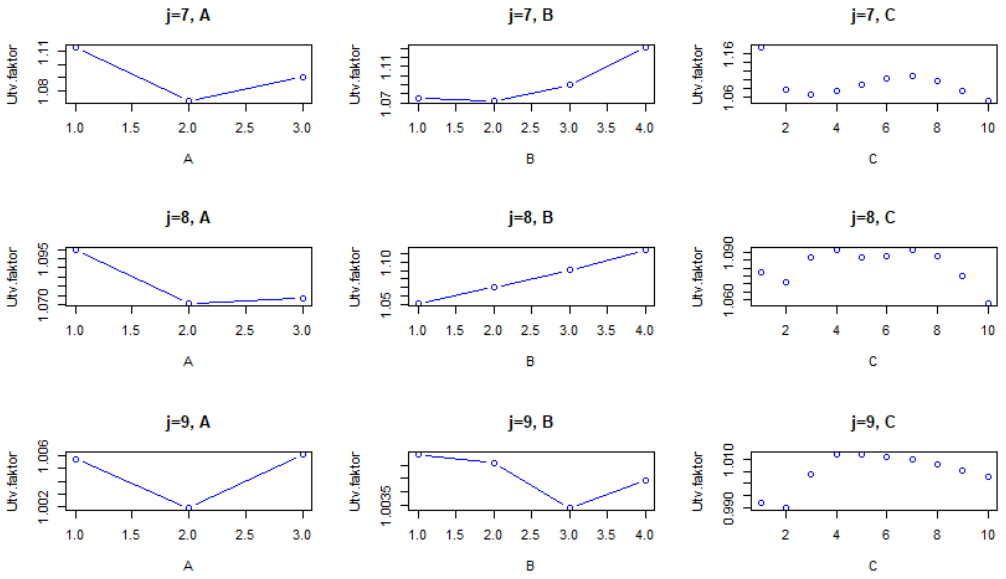
Figur A2: Ett stapeldiagram som illustrerar det totala ultimo värdet för skadeår 4-12 inom de tio klasserna som finns för faktor C. Där den röda stapeln illustrerar det sanna värdet för data II. Medan de gröna staplarna är skattningar av det sanna resultatet när vi använt oss av ett neuronnät enligt metod A samtidigt som de blå staplarna illusterar motsvarande fast med metod B. Vidare gäller det även att de gröna respektive blå staplarna är medelvärdet av resultaten när vi upprepat simuleringarna 100 gånger för både metod A och metod B


j=1, A
j=1, B
j=1, C
j=2, A
j=2, B
j=2, C
j=3, A
j=3, B
j=3, C
j=4, A
j=4, B
j=4, C
j=5, A
j=5, B
j=5, C
j=6, A
j=6, B
j=6, C
Utv.faktor
A
B
C

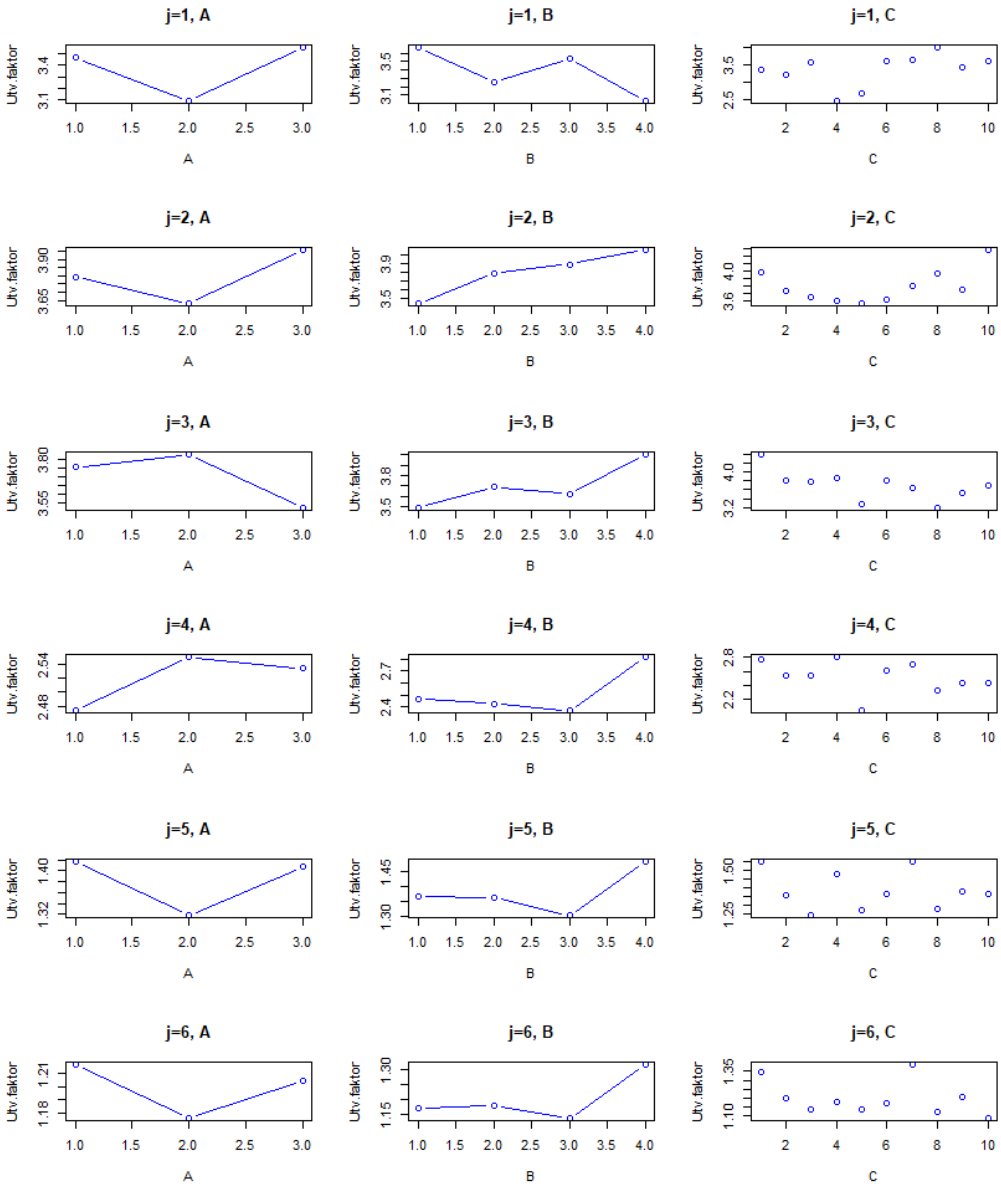


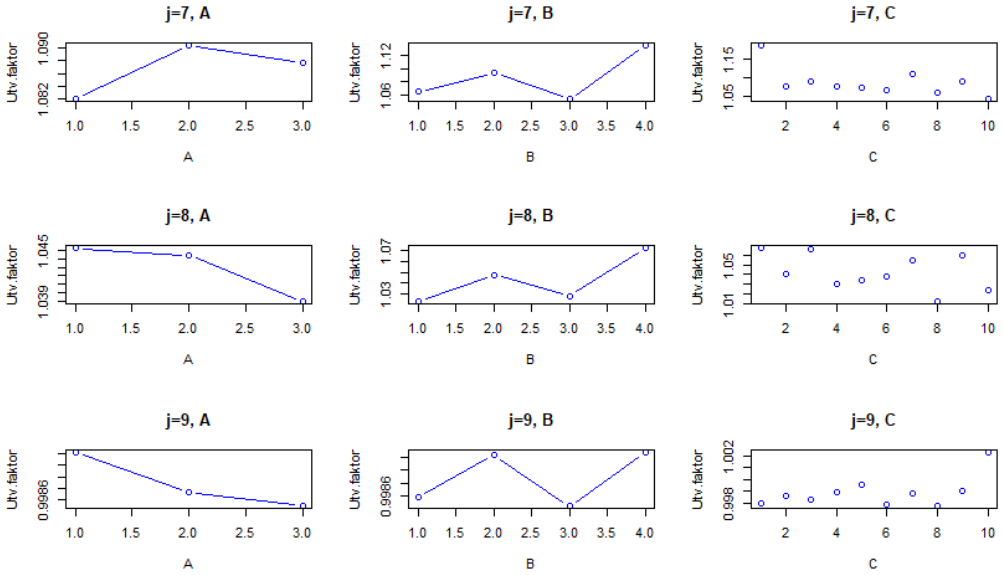
Figur A3: De estimerade utvecklingsfaktorerna $\hat{f}_j(\mathrm{x})$ för de enskilda egenskapsfaktorerna A, B respektiva C. Där det i detta fall är metod A och data I som är av intresse.


j=1, A
j=1, B
j=1, C
j=2, A
j=2, B
j=2, C
j=3, A
j=3, B
j=3, C
j=4, A
j=4, B
j=4, C
j=5, A
j=5, B
j=5, C
j=6, A
j=6, B
j=6, C
Utv.faktor
A
B
C

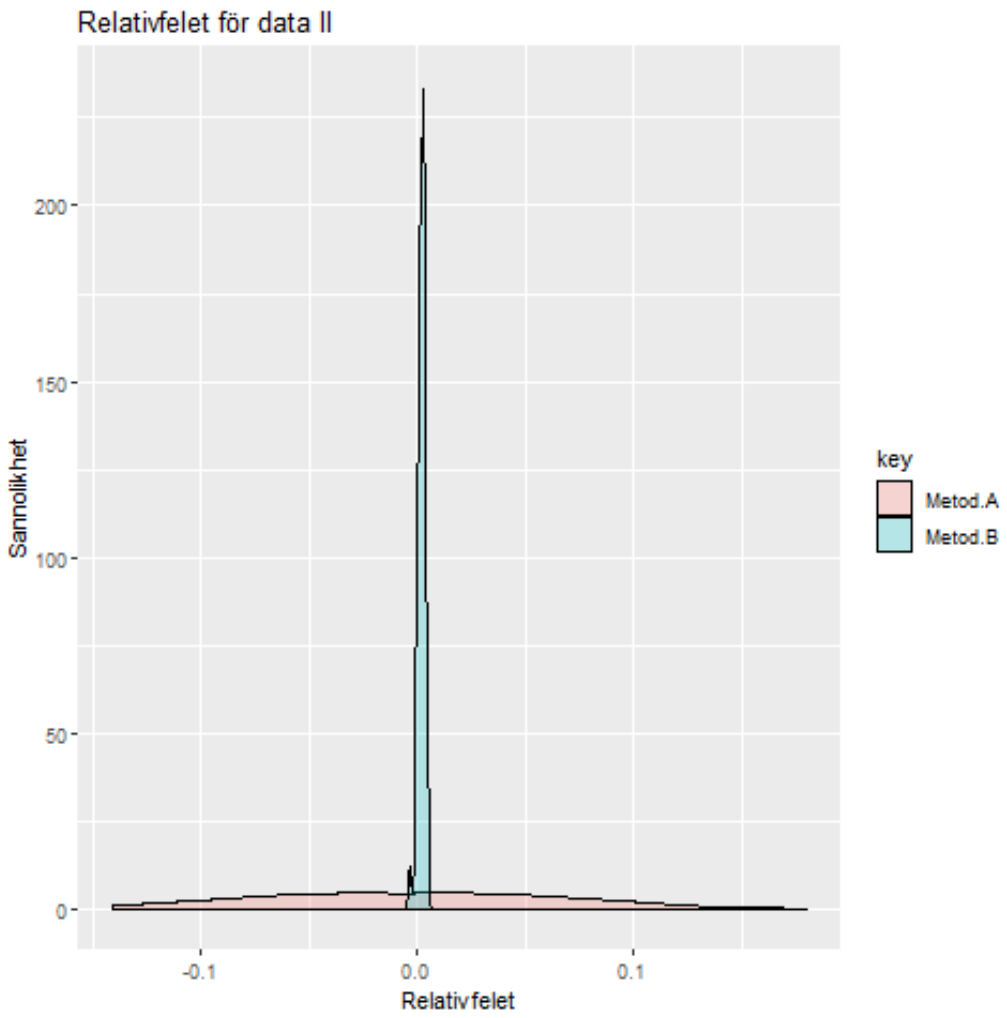


Figur A4: De estimerade utvecklingsfaktorerna $\hat{f}_j(\mathrm{x})$ för de enskilda egenskapsfaktorerna A, B respektiva C. Där det i detta fall är metod A och data II som är av intresse.


j=1, A
j=1, B
j=1, C
j=2, A
j=2, B
j=2, C
j=3, A
j=3, B
j=3, C
j=4, A
j=4, B
j=4, C
j=5, A
j=5, B
j=5, C
j=6, A
j=6, B
j=6, C
Utv.faktor
A
B
C



Figur A5: De estimerade utvecklingsfaktorerna $\hat{f}_j(\mathrm{x})$ för de enskilda egenskapsfaktorerna A, B respektiva C. Där det i detta fall är metod B och data II som är av intresse.



Figur A6: Fördelningen av relativfelet för den totala reserven för data II