Bootstrapping og simulering

Tilleggslitteratur for STK1100

Geir Storvik

April 2014*

1 Introduksjon

Simulering av tilfeldige variable (stokastisk simulering) er et nyttig verktøy innenfor statistikk, men også mer generelt. Dette notatet er et supplement til STK1100. Læreboka til Devore & Berk (2011) diskuterer bootstrapping, men stoffet er fordelt over ulike kapitler og avsnitt og er noe overfladisk. Tilsvarende er stokastisk simulering stort sett kun diskutert gjennom oppgaver. Dette notatet er ment å samle hovedideene bak bootstrapping og samtidig si noe mer om potensialet for stokastisk simulering. Samtidig gir det eksempler på hvordan bootstrapping og stokastisk simulering kan utføres i MATLAB .

2 Bootstrapping

Anta vi har observert $x_1, \ldots, x_n \sim F$ der F er en kumulativ fordelingsfunksjon[1]. Vi ønsker å estimere en ukjent parameter θ med $\hat{\theta} = \hat{\theta}(\mathbf{x})$ der $\mathbf{x} = (x_1, \ldots, x_n)$. Noen av de spørsmål vi gjerne stiller er:

1. Er $\hat{\theta}$ forventningsrett, og hvis ikke hva er skjevheten til estimatet?
2. Hva er usikkerheten til $\hat{\theta}$?

*Oppdatert av Geir Storvik i april 2016 og Ørnulf Borgan i april 2017 og april 2019.

[1]Det er litt enklere å snakke om F istedet for punktsannsynligheten p eller sannsynlighetstettheten f da vi med F ikke behøver å skille mellom diskrete og kontinuerlige fordelinger.

Disse er knyttet til fordelingsegenskapene til $\hat{\theta}$ som avhenger av F. Med dette mener vi oppførselen til $\hat{\theta}$ hvis vi gjentar vårt eksperiment med å samle inn data $\mathbf{x}$ fra fordelingen beskrevet med F.

Et problem her er at F er ukjent, noe som medfører at også fordelingsegenskapene til $\hat{\theta}$ er ukjente. For å kunne gi noen svar på spørsmålene ovenfor, må vi derfor gjøre noen tilnærminger. I kapitlene 6 og 7 i Devore & Berk (2011) diskuteres at normaltilnærming (begrunnet gjennom sentralgrenseteoremet) ofte kan benyttes. Men hva hvis antagelsene som ligger til grunn for disse tilnærmingene ikke er tilstede? *Bootstrapping* er da et alternativ.

2.1 Bootstrapping og standardfeil

Ideen bak *bootstrapping* (Efron 1982) kan illustreres gjennom et eksempel. La $X_i, i = 1, ..., n$ være levetidene til n komponenter. Anta vi har observert (eksempel 7.12 i Devore & Berk 2011)

41.53 18.73 2.99 30.34 12.33 117.52 73.02 223.63 4.00 26.78

En ofte brukt modell for levetider er den eksponensielle fordeling, $f(x; \lambda) = \lambda e^{-\lambda x}$. Innenfor denne modellen er $E(X_i) = \mu = 1/\lambda$. Dermed er $E[\bar{X}] = \mu$ og en rimelig estimator for λ er

$$\hat{\lambda} = 1/\bar{X}$$

For observasjonene ovenfor får vi estimatet $\hat{\lambda} = 1/55.087 = 0.0182$. Hva er standardfeilen til dette estimatet?

La oss først se på variansen til $\hat{\lambda}$. Merk at variansen er et mål på variasjonen i $\hat{\lambda}$ ved gjentatte eksperimenter der vi samler inn $n = 10$ observasjoner i hvert eksperiment. Hvis vi kunne repetere eksperimentet mange ganger ville vi derfor kunne *estimere* variansen til $\hat{\lambda}$ ved den empiriske variansen over disse repeterte eksperimentene. Standardfeilen kan så fås ved å ta kvadratroten av den empiriske variansen.

Problemet ligger i at vi ikke klarer å repetere eksperimentene i virkeligheten. Men vi kan *simulere* tilsvarende eksperimenter på datamaskinen.

Anta den eksponensielle fordeling er den riktige fordeling og at vi kjenner λ. Vi kan da gjennomføre $B = 1000$ simuleringseksperimenter med følgende MATLAB kode (vi må gi verdier for `n` og `lambda`):

```
B = 1000;
for b=1:B
    xstar = exprnd(1/lambda,1,n);
```

```
    lambdasim(b) = 1/mean(xstar);
end
SE=sqrt(var(lambdasim))
```

Vi vil da få ut standardfeilen til de simulerte $\hat{\lambda}$-ene (vektoren `lambdasim`). Merk at vi selv kan bestemme B slik at ved å gjøre B stor nok får vi så stor nøyaktighet som vi vil.

Et ekstra problem er imidlertid at vi ikke kjenner den riktige fordeling til observasjonene våre. Anta vi er villige til å anta *modellen* om at de er eksponensielt fordelte slik at kun λ er ukjent. Fra data har vi imidlertid et *estimat* på λ. En *tilnærming* er da å bruke $f(x;\hat{\lambda})$ istedet for $f(x;\lambda)$. MATLAB kode for dette blir da:

```
x = [41.53 18.73 2.99 30.34 12.33 117.52 73.02 223.63 4.00 26.78];
n = 10;
mu_hat = mean(x)
lambda_hat = 1/mu_hat
B = 1000;
lambdasim = zeros(1,B);
for b=1:B
    xstar = exprnd(mu_hat,1,n);
    lambdasim(b) = 1/mean(xstar);
end
SE = sqrt(var(lambdasim))
```

Fem kjøringer av disse komandoene ga standardfeil verdiene:

0.00695 0.00699 0.00727 0.00689 0.00685

Merk at vi vil få litt forskjellige verdier hver gang vi kjører. Dette kommer av tilfeldighetene som ligger i simuleringene våre. En økning av B til 10000 ga verdiene:

0.00710 .00708 0.00694 0.00734 0.00702

Vi ser nå at vi får et mye mer nøyaktig resultat.

Merk at det er *to kilder til usikkerhet*:

- Usikkerhet i de opprinnelige observasjoner $x_1, ..., x_n$
- Usikkerhet i våre simuleringer

I praksis vil usikkerheten som ligger i data være lagt større enn usikkerheten i våre simuleringer slik at $B = 1000$ typisk vil være nok.

Prosedyren vi har gjennomført i eksempelet ovenfor kalles for *parametrisk bootstrapping*. Generelt går denne prosedyren ut på at hvis data er generert fra en modell med kumulativ fordelingsfunksjon $F(x;\theta)$ og vi har en estimator $\hat{\theta} = \hat{\theta}(\mathbf{x})$ så kan vi estimere standard feilen til $\hat{\theta}$ ved følgende trinn:

1. Repeter for $b = 1, ..., B$:
   (a) Simuler $x_1^*, ..., x_n^*$ u.i.f. fra $F(x;\hat{\theta})$
   (b) Sett $\theta^{*b} = \hat{\theta}(\mathbf{x}^*)$, der $\mathbf{x}^* = (x_1^*, ..., x_n^*)$
2. Estimer $\sigma_{\hat{\theta}}$ ved $\sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\theta^{*b} - \bar{\theta}^*)^2}$ der $\bar{\theta}^* = \frac{1}{B}\sum_{b=1}^{B}\theta^{*b}$.

I parametrisk bootstrapping estimerer vi standardfeilen under antagelsen om at modellen $F(x;\theta)$ er riktig. Hvis virkelige modellen ikke er alt for forskjellig fra $F(x;\theta)$, vil svarene vi får være fornuftige. Hvis vi er usikre på dette kan *ikke-parametrisk bootstrapping* være et alternativ. Idéen i dette tilfellet er å gjøre minimale antagelser på F i utgangspunktet. Merk at

$$F(x) = P(X \leq x)$$

En mulig estimator for F i det tilfellet er den *empirisk kumulative fordelingsfunksjon* som er gitt ved

$$\widehat{F}(x) = \frac{1}{n}(\#x_i \leq x) \tag{1}$$

Merk at under $\widehat{F}$ så er $P(X = x_i) = \frac{1}{n}$ for $i = 1, \ldots, n$, og at trekking fra $\widehat{F}$ svarer til å *trekke fra $x_1, ..., x_n$ med tilbakelegging*. Dette er en viktig egenskap ved $\widehat{F}$ som vi vil utnytte når vi skal utføre de nødvendige beregninger involvert i bootstrapping.

La oss igjen se på eksemplet vårt med levetider av komponenter. Nedenfor følger MATLAB kode for å utføre ikke-parametrisk bootstrapping:

```
B = 1000;
lambdasim = zeros(1,B);
for b=1:B
    xstar = randsample(x,n,true);
    lambdasim(b) = 1/mean(xstar);
end
SE = sqrt(var(lambdasim))
```

Fem gjentatte kjøringer av disse kommandoene ga standardfeil

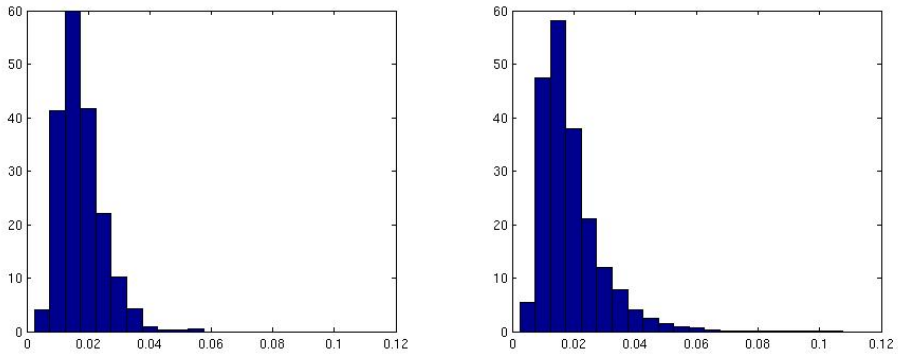
0.01001 0.00995 0.01024 0.01008 0.00984

Vi ser at standardfeilen har økt noe. Dette kan tyde på at dataene ikke helt passer i forhold til den ekspoensielle fordelingen (hvis modellantagelsene er riktige vil resultatene fra ikke-parametrisk og parametrisk bootstrapping være rimelig like).

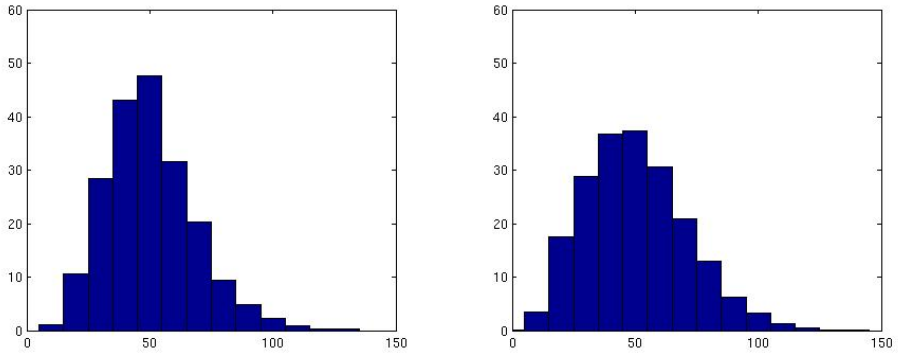
2.2 Bootstrapping og forventningsskjevhet

Simuleringene av $\hat{\theta}$ vi får gjennom bootstrapping kan brukes til mer enn bare å estimere standardfeil. Figuren nedenfor viser histogrammer av de simulerte $\hat{\theta}$-ene, parametrisk bootstrapping til venstre og ikke-parametrisk bootstrapping til høyre. Her ser vi at fordelingene er ganske skjeve. Dette tyder på at sentralgrenseteoremet ikke gjelder her. Det er to grunner til det:

- Det er få observasjoner ($n = 10$)
- $\hat{\lambda}$ er ikke et gjennomsnitt



Her kan et alternativ være å se på $\hat{\mu} = \bar{X}$ i stedet. Plottet nedenfor viser Bootstrap simuleringer av denne. Vi får nå noe som ligner mer på normalfordelingen, men fremdeles er det noe ikke-symmetri som indikerer at $n = 10$ er noe lite i dette tilfellet der vi starter med en ganske skjev fordeling på dataene.



La oss imidlertid gå tilbake til $\hat{\lambda}$, men nå konsentrere oss om forventningsskjevhet. Vi er altså interesser i å si noe om

$$\text{Bias}(\hat{\theta}) = E[\hat{\theta}] - \theta$$

Denne forventningsskjevheten kan estimeres fra bootstrap simuleringene våre ved

$$\widehat{\text{Bias}}(\hat{\theta}) = \bar{\theta}^* - \hat{\theta}$$

For vårt tidligere eksempel kan dette utføres ved MATLAB kommandoen:

```
bias = mean(lambdasim)-lambdahat
```

En simulering med parametrisk bootstrapping ga $\widehat{\text{Bias}}(\hat{\theta}) = 0.00214$ mens ikke-parametrisk bootstrapping ga 0.00309. Vi kan nå oppsummere våre resultater i følgende tabell (basert på første kjøring med $B = 1000$):

Bootstrapping	Estimat	SE	Bias
Parametrisk	0.01815	0.00695	0.00214
Ikke-parametrisk	0.01815	0.01001	0.00309

Dette kan nå brukes til å anslå forventet kvadratisk feil for $\hat{\theta}$. For parametrisk bootstrapping blir dette

$$\text{MSE} = 0.00695^2 + 0.00214^2 = [4.830 + 0.4580] \cdot 10^{-5} = 5.288 \cdot 10^{-5}$$

mens for ikke-parametrisk bootstrapping er de tilsvarende tallene

$$\text{MSE} = 0.01001^2 + 0.00309^2 = [1.002 + 0.095] \cdot 10^{-4} = 1.097 \cdot 10^{-4}$$

I begge tilfeller ser vi at forventningsskjevheten bidrar svært lite til den totale feilen noe som tilsier at forventingsskjevheten er liten i dette tilfellet.

3 Stokastisk simulering og Monte Carlo integrasjon

I forbindelse med bootstrapping brukte vi simulering av tilfeldige variabler som verktøy for å få ut de svar vi ønsket. Når tilfeldighet inngår i simuleringer kalles det ofte for *stokastisk simulering*. To aktuelle faktorer er interessante å diskutere i den forbindelse:

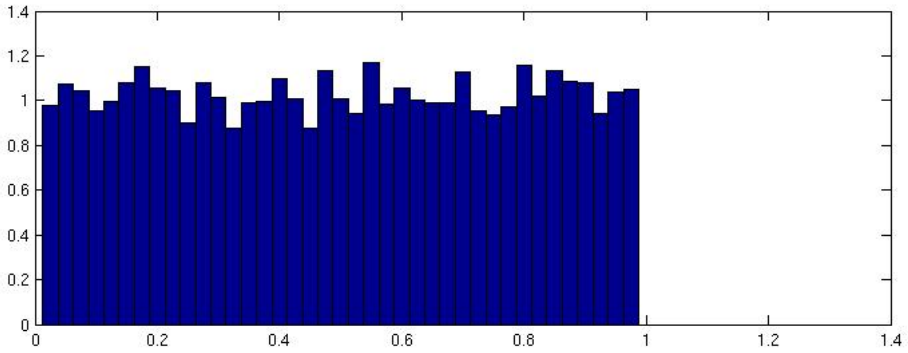
- Hvordan kan en datamaskin utføre simuleringer fra en vilkårlig fordeling?
- Hva annet kan vi bruke stokastisk simulering til?

3.1 Simulering fra en vilkårlig fordeling

Formelt er datamaskinen en deterministisk boks (selv om det kanskje ikke alltid virker slik). Det beste vi derfor kan gjøre er å konstruere algoritmer som kan få datamaskinen til å simulere hva vi kaller pseudotilfeldige tall, dvs de tall vi får ut ligner på den fordeling vi er ute etter. Utgangspunktet for all stokastisk simulering er simulering av variabler fra den uniforme fordeling på intervallet $[0, 1]$. De fleste datamaskiner og tilhørende software har idag ferdige funksjoner som generer pseudotilfeldige tall som for alle praktiske formål ligner på tilfeldige tall fra den uniforme fordelingen over $[0, 1]$. Kommandoene nedenfor viser hvordan dette kan gjøres i MATLAB , innkludert plotting av histogram.

```
u = rand(1000);
x = 0:0.025:1
n = histc(u,x);
bar(x,n/(sum(n)*0.025),1);
```

Figuren nedenfor viser histogrammet av de genererte variablene og illustrerer at vi får variabler som passer godt med den uniforme fordelingen.



3.2 Inversjonsmetoden

Anta nå vi ønsker å simulere fra en kontinuerlig fordeling med kumulativ fordelingsfunksjon F. La $U \sim \text{uniform}[0, 1]$ og la $X = F^{-1}(U)$. Da er

$$\begin{aligned} P(X \leq x) =& P(F^{-1}(U) \leq x) \\ =& P(U \leq F(x)) = F(x) \end{aligned}$$

som viser at X da har kumulativ fordelingsfunksjon F.

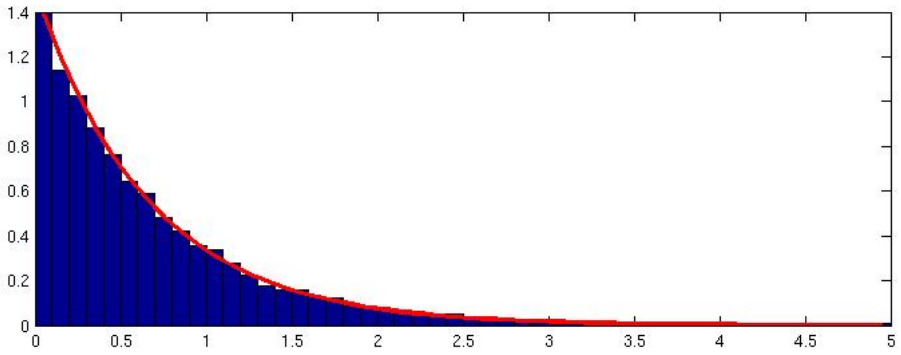
Eksempel Anta vi ønsker å trekke fra den eksponensielle fordeling $f(x;\lambda) = \lambda e^{-\lambda x}$. Da er $F(x;\lambda) = 1-e^{-\lambda x}$ og $F^{-1}(u) = -\lambda^{-1}\log(1-u)$. Dermed kan vi bruke følgende algoritme for å generere en variabel fra den eksponensielle fordeling

1. Generer $u \sim \text{Uniform}[0,1]$.
2. Sett $x = -\lambda^{-1}\log(1-u)$

I MATLAB med $\lambda = 1.5$ kan flere variable genereres simultant ved kommandoene:

```
lambda = 1.5;
u = rand(1,10000);
x = -log(1-u)/lambda;
```

Nedenfor er et histogram av de simulerte x-ene sammen med plot av den sanne sannsynlighetstetthet $f(x;\lambda)$.



For de fleste kjente fordelinger vil det imidlertid finnes ferdige funksjoner som gjør dette for oss, for den eksponensielle fordeling så vi tidligere at vi i MATLAB kan bruke funksjonen `exprnd`. Det er imidlertid andre fordelinger der ikke slike funksjoner er tilgjengelige

Eksempel Cauchy fordeling Cauchy fordelingen er definert ved

$$f(x) = \frac{1}{\pi(1+x^2)}$$

Det ser ikke ut til å være noen ferdig implementerte funksjoner i standard MATLAB for å generere variable fra denne fordelingen. Vi har imidlertid at

$$F(x) = \tfrac{1}{2} + \pi^{-1}\tan^{-1}(x)$$

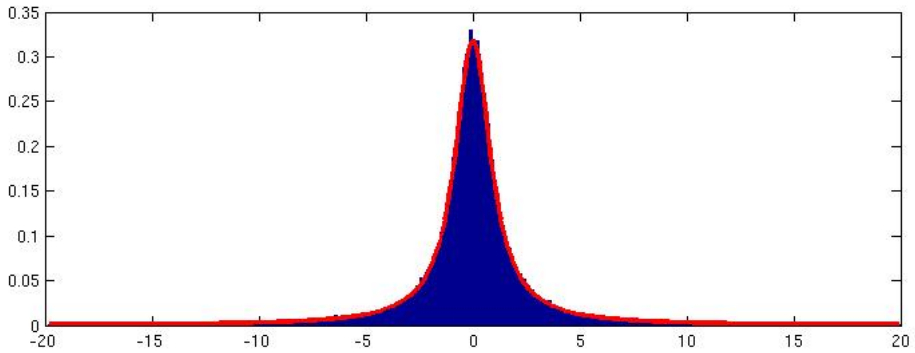
som gir

$$F^{-1}(u) = \tan[\pi(u - \tfrac{1}{2})]$$

Dermed kan vi generere tilfeldige variable fra Cauchy fordelingen ved følgende MATLAB kommandoer:

```
u = rand(1,10000);
x = tan(pi*(u-0.5));
```

Nedenfor er et histogram[2] av de simulerte x-ene sammen med plot av den sanne sannsynlighetstettheten $f(x)$.



3.3 Forkastningsmetoden

Anta nå vi ønsker å simulere fra en fordeling med sannsynlighetstetthet

$$f(x) = 20x(1-x)^3, \quad 0 < x < 1$$

Dette er et spesialtilfelle av *beta fordelingen* som er en mye brukt fordeling for observasjoner som er begrenset til intervallet $[0, 1]$. Et problem her er at den kumulative fordelingsfunksjonen $F(x)$ ikke har noe analytisk uttrykk, noe som gjør inversjonsmetoden vanskelig å benytte. Et alternativ da er *forkastningsmetoden*.

Forkastningsmetoden trenger en *forslagsfordeling* $G(x)$ med tilhørende sannsynlighetstetthet $g(x)$ slik at $f(x)/g(x) \leq c$ for alle x. Metoden er da definert gjennom følgende algoritme:

1. Simuler $y \sim G(\cdot)$.

[2]Mulig kommando: `histogram(x,'BinLimits',[-20,20], 'Normalization','pdf')`

2. Generer $u \sim \text{Uniform}[0,1]$

3. Hvis $u \leq f(y)/(cg(y))$, sett $x = y$, hvis ikke returner til trinn 1

Proposisjon 1
Variabelen som kommer ut av forkastningsmetoden vil ha sannsynlighetstetthet $f(x)$. Sannsynligheten for å akseptere en generert y er $1/c$.

Bevis Se oppgave 5.85 i Devore & Berk (2011)

Merk at c er et mål på hvor *effektiv* algoritmen er. Jo større c er, jo mindre er sannsynligheten for å akseptere og jo flere forsøk må vi gjøre. Vi ønsker derfor at c skal være minst mulig. Samtidig har vi begrensningen $c \geq f(x)/g(x)$. Et vanlig valg er derfor å velge

$$c = \max_x \frac{f(x)}{g(x)}$$

Merk spesielt at vi må ha at $g(x) > 0$ for alle verdier der $f(x) > 0$.

Eksempel Betrakt igjen eksemplet

$$f(x) = 20x(1-x)^3, \quad 0 < x < 1$$

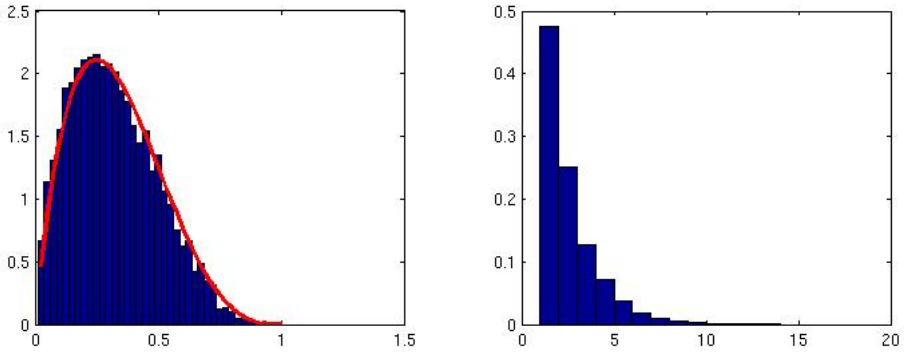
Siden vi her har en fordeling over intervallet $[0,1]$ kan en mulig forslagsfordeling være den uniforme fordeling. I det tilfellet er

$$c = \max_x \frac{f(x)}{g(x)} = \max_x 20x(1-x)^3$$

Her får vi den største verdien for $x = 1/4$ som gir $c = \frac{135}{64} = 2.1094$. I MATLAB kan algoritmen implementeres på følgende måte:

```
n = 10000;x = zeros(1,n);cinv = 64/135;
for i=1:n
  y = rand(1);
  u = rand(1);
  while (u > 20*y*(1-y)*(1-y)*(1-y)*cinv)
   y = rand(1);
   u = rand(1);
  end
  x(i) = y;
end
```

Figuren nedenfor viser til venstre histogram av de simulerte x-er sammen med den sanne $f(x)$. Vi ser også her at det er godt samsvar mellom de simulerte verdier og den modell de skal være simulert ifra. Til høyre er et histogram over antall simuleringer som måtte gjennomføres før en verdi ble akseptert.



3.4 Monte Carlo integrasjon

Vi vil nå se på hvordan stokastisk simulering kan brukes til å løse et matematisk problem. Anta vi er interessert i å beregne

$$\theta = \int_{-\infty}^{\infty} \cos(x) e^{-x^2/2} dx.$$

Merk at vi kan omskrive integralet til

$$\begin{aligned}\theta &= \int_{-\infty}^{\infty} \sqrt{2\pi}\cos(x)\frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \\ &= \int_{-\infty}^{\infty} \sqrt{2\pi}\cos(x) f(x) dx\end{aligned}$$

der $f(x)$ er sannsynlighetstettheten til standardnormalfordelingen. Dermed er

$$\theta = E[\sqrt{2\pi}\cos(X)],$$

der X nå er en tilfeldig variabel som følger standardnormalfordelingen. Anta nå vi kan simulere $X_1, ..., X_M$ fra $f(x)$. Vi kan da bruke vår sannsynlighetsteori til å konstruere en *estimator* for θ ved

$$\widehat{\theta} = \frac{1}{M}\sum_{i=1}^{M} \sqrt{2\pi}\cos(X_i)$$

I MATLAB kan vi nå få et estimat på θ ved følgende kommandoer:

```
M = 10000;
x = normrnd(0,1,1,M);
y = sqrt(2*pi)*cos(x)
thetahat = mean(y);
```

Fem kjøringer av disse kommandoene ga verdiene

1.5149 1.5206 1.5336 1.5116 1.5132

Hver av disse verdiene er et estimat på θ. Disse kan så sammenliknes med hva man får ut av en numerisk integrasjonsmetode, 1.5203, som i dette tilfellet kan sees på som en fasit. Merk at estimatene ligger i nærheten av denne verdien, men varierer noe. Vi kan øke M til 10000000. Da ender vi opp med verdiene

1.5205 1.5196 1.5208 1.5203 1.5204

Vi ser nå at estimatene varierer mye mindre rundt den riktige verdien. Fra standard regler (med $Y_i = \sqrt{2\pi}\cos(X_i)$) har vi at

$$V(\widehat{\theta}) = \frac{1}{M} V(Y_i)$$

og

$$\sigma_{\widehat{\theta}} = \frac{1}{\sqrt{M}} \sigma_{Y_i}$$

σ_{Y_i} vil være vanskelig å beregne, men kan også estimeres (med s_{Y_i}) fra våre simuleringer:

```
Sy = std(y)
```

En ny kjøring med $M = 10000000$ ga $\widehat{\theta} = 1.5206$ og $s_{Y_i} = 1.1204$. Dette gir $s_{\widehat{\theta}} = 0.000354$ som estimat for $\sigma_{\widehat{\theta}}$. For å vurdere nøyaktigheten på estimatet av θ, kan vi beregne et (tilnærmet) 95% konfidensintervall ved

$$\widehat{\theta} \pm 1.96 \cdot s_{\widehat{\theta}}$$

jf. side 394 i Devore & Berk (2011). En "feilmargin" for estimatet er dermed

$$\pm 1.96 \cdot s_{\widehat{\theta}} = \pm 1.96 \cdot 0.000354 = \pm 0.00069$$

Eksempelet ovenfor illustrerer en generell prosedyre kalt *Monte Carlo integrasjon* eller *Monte Carlo metoden*: Anta vi er interessert i et integral

$$\theta = \int_{-\infty}^{\infty} g(x)dx$$

Dette kan vi skrive om til

$$\theta = \int_{-\infty}^{\infty} \frac{g(x)}{f(x)} f(x) dx$$

der vi krever at $f(x) > 0$ hvis $g(x) > 0$. Denne omskrivningen gjelder for enhver funksjon $f(x)$ med disse begrensningene. Vi vil imidlertid begrense oss ytterligere til å anta at $f(x)$ er en sannsynlighetstetthetsfunksjon, dvs $f(x) \geq 0$ og $\int_{-\infty}^{\infty} f(x)dx = 1$. Under denne antagelsen er

$$\theta = E\left(\frac{g(X)}{f(X)}\right)$$

der X nå er en tilfeldig variabel med sannsynlighetstetthetsfunksjon $f(x)$. Anta nå vi kan simulere $X_1, ..., X_M$ u.i.f. (uavhengig og identisk fordelt) fra $f(x)$ og la $Y = \frac{g(X)}{f(X)}$. Da kan vi estimere θ ved

$$\widehat{\theta} = \frac{1}{M} \sum_{i=1}^{M} Y_i$$

Vi har direkte at $E(\widehat{\theta}) = \theta$. Videre har vi at

$$V(\widehat{\theta}) = \frac{1}{M} V(Y)$$

og

$$\sigma_{\widehat{\theta}} = \frac{1}{\sqrt{M}} \sigma_Y$$

Vi kan dermed styre usikkerheten i estimatet ved å velge passende antall simuleringer M.

Et annet aspekt ved denne metoden er at vi også kan velge $f(x)$, som ofte blir kalt *forslagsfordelingen*. Ulike valg av denne vil gi ulike verdier av $\sigma_{g(X)/f(X)}$. Noen ganger kan usikkerheten reduseres kraftig ved smarte valg av forslagsfordeling.

Eksempel Anta vi er interessert i

$$\theta = \int_{2}^{\infty} \frac{1}{\pi(1+x^2)} dx$$

Merk at $f(x) = \frac{1}{\pi(1+x^2)}$ er sannsynlighetstettheten til Cauchy fordelingen slik at

$$\theta = P(X \geq 2)$$

hvis X er en tilfeldig variabel fra Cauchy fordelingen. Den opplagte metode for å estimere θ gjennom Monte Carlo integrasjon er dermed å simulere $X_1, ..., X_M$ uif fra Cauchy-fordelingen, la $Y_i = 1$ hvis $X_i \geq 2$ og 0 ellers og så la

$$\hat{\theta} = \frac{1}{S} \sum_{i=1}^{M} Y_i$$

Referanser

Devore, J. L. & Berk, K. N. (2011), *Modern Mathematical Statistics with Applications*, Springer.

Efron, B. (1982), *The Jackknife, the Bootstrap, and Other Resampling Plans*, number 38 *in* 'CBMS-NSF Regional Conference Series in Applied Mathematics', Siam, Philadelphia.