Föreläsning 12

Statistisk inlärning och dataanalys (Kungliga Tekniska Högskolan)

Föreläsning 12

12.1 Monte Carlo-metoden

På flera ställen i kursen har vi stött på uttryck på formen

$$\mathrm{E}[h(X)]$$

för någon funktion $h : \mathbb{R} \to \mathbb{R}$ och stokastisk variabel X, dvs. väntevärden (betingade eller inte) som måste beräknas. Inom bayesiansk inferens har vi till exempel aposterioriväntevärden och aposteriorivarianser, men också svanssannolikheter som kan skrivas om som väntevärden

$$\mathrm{P}(\Theta > \theta_0 \mid X = x) = \mathrm{E}[\mathbf{1}_{(-\infty,\theta_0]}(\Theta) \mid X = x],$$

där $\mathbf{1}_A(x)$ avser indikatorfunktionen på A (dvs. funktionen som är ett för $x \in A$ och noll annars). För att bestämma symmetriska aposterioriintervall söker vi a och b så att

$$\mathrm{P}(\Theta \leq a \mid X = x) = \alpha/2 \quad \text{och} \quad \mathrm{P}(\Theta \geq b \mid X = x) = \alpha/2,$$

vilket är ekvivalent med att invertera funktionerna

$$a \mapsto \mathrm{P}(\Theta \leq a \mid X = x) = \mathrm{E}[\mathbf{1}_{(-\infty,a]}(\Theta) \mid X = x]$$

och

$$b \mapsto \mathrm{P}(\Theta \geq b \mid X = x) = \mathrm{E}[\mathbf{1}_{[b,+\infty)}(\Theta) \mid X = x].$$

Liknande frågeställningar uppstår också inom frekventistisk inferens när vi söker kritiska värden för hypotestest eller gränser för konfidensintervall.

Ibland kan dessa väntevärden beräknas genom direkt integration eller summering (även om det kan vara svårt ibland). I vissa fall kan vi använda trick som lagen om total förväntan eller lagen om total varians. Men ibland är det inte möjligt att beräkna dessa väntevärden för hand.

Då krävs det numeriska approximationer av väntevärdet. För diskreta stokastiska variabler innebär detta att utvärdera summan

$$\mathrm{E}[h(X)] = \sum_{x=0}^{+\infty} h(x) f_X(x),$$

vilket ofta är ganska smärtfritt. För kontinuerliga stokastiska variabler har vi däremot integralen

$$\mathrm{E}[h(X)] = \int_{\mathbb{R}} h(x) f_X(x) dx,$$

som kan vara svårare att approximera. Ett tillvägagångsätt är att tillämpa numeriska metoder som kvadraturregler där värdet hos integralen kan beräknas upp till en godtycklig felmarginal. Detta kan vara dock vara tidskrävande både med avseende på implementering och exekvering, speciellt när antalet dimensioner är stort.

Ett problem med kvadraturregler (och andra numeriska integrationsmetoder) är att integralen påverkas mer av de värden där $f_X(x)$ är stor, dvs. de värden på X som har större sannolikhet. Kvadraturregler approximerar alla delar av integranden med samma precision och kan på det sättet vara ineffektiv från en beräkningssynpunkt. Bättre vore att lägga mer fokus på de värden där $f_X(x)$ är stor än på de värden där $f_X(x)$ är liten.

Ett sätt att göra detta är att sampla flera stokastiska variabler $X_1, X_2, \ldots, X_k$ med samma fördelning som X och beräkna summan

$$H_k = \frac{1}{k} \sum_{i=1}^{k} h(X_i).$$

Enligt stora talens lag har vi att H_k konvergerar till $\mathrm{E}[h(X)]$ när $k \to +\infty$, mer specifikt att $\mathrm{P}(|H_k - \mathrm{E}[h(X)]| < \epsilon) \to 1$ för varje $\epsilon > 0$. Med centrala gränsvärdessatsen har vi ett starkare resultat där vi får att H_k har den approximativa fördelningen

$$\mathrm{N}(\mathrm{E}[h(X)], \mathrm{Var}[h(X)]/k)$$

när k är tillräckligt stort. Eftersom $\mathrm{Var}[h(X)]$ inte är känd här (att beräkna denna analytiskt är minst lika svårt som att beräkna $\mathrm{E}[h(X)]$) approximerar vi den med

$$\frac{1}{k}\sum_{i=1}^{k}(h(X_i) - H_k)^2.$$

Vi delar med k för att approximera variansen av H_k och tar kvadratroten för att få medelfelet

$$\sqrt{\frac{1}{k}\cdot\frac{1}{k}\sum_{i=1}^{k}(h(X_i) - H_k)^2} = \frac{1}{k}\sqrt{\sum_{i=1}^{k}(h(X_i) - H_k)^2}.$$

Detta kan användas för att utvärdera noggranheten i skattningen av $\mathrm{E}[h(X)]$.

Metoden ovan brukar kallas för *Monte Carlo-metoden*. Namnet kommer från forskningsprojektet där metoden utvecklades under andra världskriget. Den ursprungliga tillämpningen var nämligen simulering av neutrondiffusion i en atombomb, forskning som var strikt sekretessbelagd och därför krävde ett kodnamn. En av forskarna hade en släkting som brukade spela bort sina (och släktens) pengar på kasinot i Monte Carlo och detta i kombination med metodens användning av slumpmässighet gav upphov till namnet.

Exempel 12.1. Låt oss återgå till Exempel 8.5 (skattning av vindhastighet för småskalig vindkraft) där vi tog fram aposteriorifördelningen

$$V \mid \boldsymbol{X} = \boldsymbol{x} \sim \mathrm{t}(6, 4.68, 0.31).$$

Aposterioriväntevärdet $\mathrm{E}[V \mid \boldsymbol{X} = \boldsymbol{x}]$ är här enkelt att få ut och är lika med 4.68, men det som intresserade oss var $\mathrm{E}[V^3 \mid \boldsymbol{X} = \boldsymbol{x}]$ (eftersom detta var en term i vår aposterioririsk).

Om vi approximerar $\mathrm{E}[V^3 \mid \boldsymbol{X} = \boldsymbol{x}]$ med Monte Carlo-metoden och varierar antalet utfall k får vi följande tabell:

k	approximation	approximationsfel
1000	109.05	1.55
10000	109.42	0.478
100000	109.07	0.158
1000000	108.97	0.0490

Beroende på användningsområdet kan det alltså räcka med 10000 eller 100000 utfall för att få en bra approximation.

På samma sätt kan vi approximera sannolikheten $\mathrm{P}[V^3 > 100 \mid \boldsymbol{X} = \boldsymbol{x}]$ genom att räkna antalet utfall som uppfyller $v^3 > 100$. Vi får då

k	approximation	approximationsfel
1000	0.506	0.0158
10000	0.523	0.00499
100000	0.528	0.00158
1000000	0.527	0.000499

Sannolikheten verkar alltså ligga runt 0.53 och det räcker återigen med 10000 eller 100000 utfall.

Till slut kan vi skapa symmetriska kredibilitetsintervall genom att sortera utfallen och välja värdena på plats $(\alpha/2)k$ och $(1-\alpha/2)k$ för $\alpha = 0.05$. Här är det svårare att approximera felet, men vi kan ändå se en viss konvergens när vi ökar antalet utfall:

k	approximation
1000	[31.4, 224]
10000	[35.6, 223]
100000	[36.4, 221]
1000000	[36.5, 221]

(I detta fall kan vi beräkna kredibilitetsintervallen explicit eftersom $v \mapsto v^3$ är en monoton funktion och det därför räcker att transformera kredibilitetsintervallet för V.)

12.2 Förkastelsemetoden

Hjärtat i alla Monte Carlo-approximationer är en metod för att generera utfall från en viss fördelning. I början av kursen såg vi en grundläggande samplingalgoritm, inversionsmetoden, som gav oss ett sätt att generera utfall från fördelningar vars fördelningsfunktioner kunde inverteras enkelt. Vi såg också Box–Muller-metoden som används för att generera utfall från den standardiserade normalfördelningen. Dessa metoder går ibland under benämningen *direkta samplingmetoder* eftersom de genererar utfall för en fördelning utifrån utfall från en enklare fördelning (som den likformiga fördelningen $\mathrm{U}(0,1)$) genom variabeltransformation. Samplingmetoder som inte är på denna form kallas för *indirekta samplingmetoder* och kan användas för mer allmänna fördelningar.

En av de mer kraftfulla metoderna av detta slag är *förkastelsemetoden* (alternativt acceptans–förkastelsemetoden eller acceptans–förkastelsesampling). Vi illustrerar denna algoritm med ett exempel.

Låt oss först anta att vi har en fördelning med ändligt stöd $[a,b]$ och täthetsfunktion $f(x)$. Detta kan, till exempel, vara en $\mathrm{Beta}(\alpha,\beta)$-fördelning när α och β inte är heltal (till exempel $\alpha = 2.7$, $\beta = 6.3$). I samband med inversionsmetoden såg vi att betafördelningen för heltalsparametrar kan simuleras med utfall från gammafördelningen, vilka i sin tur kan simuleras med exponentialfördelade utfall och exponentialfördelningen kan simuleras med hjälp av inversionsmetoden. Med godtyckliga positiva parametrar fungerar tyvärr inte denna metod (samma problem uppstår om vi vill simulera en gammafördelning med en formparameter som inte är ett heltal).

Istället betraktar vi mängden

$$B = \{(x,u) : x \in [a,b], 0 \leq 0 \leq f(x)\}.$$

Vi påstår nu att om vi tar en punkt (X,U) på måfå inom B, dvs. vi genererar ett utfall från den likformiga fördelningen över B, så har X täthetsfunktionen $f(x)$ (dvs. när vi marginaliserar bort U).

Vi har alltså den simultana täthetsfunktionen

$$f_{X,U}(x,u) = \mathbf{1}_B(x,u) = \mathbf{1}_{[0,1]}(x)\mathbf{1}_{[0,f(x)]}(u).$$

Detta ger då för $x \in [a,b]$ att

$$f_X(x) = \int_{\mathbb{R}} f_{X,U}(x,u)du = \int_{\mathbb{R}} \mathbf{1}_B(x,u)du = \int_{\mathbb{R}} \mathbf{1}_{[0,f(x)]}(u)du = \int_0^{f(x)} 1du = f(x).$$

Vi kan då säga att sampling från täthetsfunktionen $f(x)$ är detsamma som sampling av (X,U) likformigt från $B = \{(x,u) : x \in [a,b], 0 \leq u \leq f(x)\}$. Detta faktum kallas ibland för *simuleringens fundamentalsats*.

Men hur hjälper detta oss? Jo, att sampla från en likformig fördelning över B är (relativt) enkelt. Nyckeln är att välja en större mängd R som innehåller B men där sampling från den likformiga fördelningen över R kan göras med standardmetoder. Sedan behåller vi endast de utfall som ligger i B (och *förkastar* de värden som hamnar i $R \setminus B$, därav namnet).

Till exempel kan vi välja

$$R = [a,b] \times [0,m],$$

där $m \geq f(x)$ för alla $x \in [a,b]$. Om vi då tar $V \sim \mathrm{U}(a,b)$ och $U \sim \mathrm{U}(0,1)$ så har vi att (V,mU) är likformigt fördelad över R.

Låt oss nu anta att vi har ett en punkt (V,mU) från den likformiga fördelningen över R. Om $(V,mU) \in B$ behåller vi alltså utfallet, annars genererar vi ett nytt utfall och börjar om. Vi fortsätter tills vi erhåller ett utfall inom B. När vi väl har detta så sparar vi V som ett utfall av X.

Algoritmen är alltså följande:

(1) Generera $V \sim \mathrm{U}(a,b)$ och $U \sim \mathrm{U}(0,1)$.

(2) Om $U < f_X(V)/m$, sätt $X = V$ och terminera, annars gå tillbaka till (1).

Vi terminerar alltså bara om $U < f_X(V)/m$, vilket betyder att sannolikheten att X antar ett visst värde är betingat på händelsen $U < f(V)/m$. Detta ger

$$\begin{aligned}
\mathrm{P}(X \le x) &= \mathrm{P}(V \le x \mid U < f(V)/m) \\
&= \frac{\mathrm{P}(V \le x, U < f(V)/m)}{\mathrm{P}(U < f(V)/m)} \\
&= \frac{\int_a^x \int_0^{f(v)/m} du dv}{\int_a^b \int_0^{f(v)/m} du dv} = \frac{\int_a^x \frac{f(v)}{m} dv}{\int_a^b \frac{f(v)}{m} dv} \\
&= \frac{\int_a^x \frac{f(v)}{m} dv}{\frac{1}{m}} = \int_a^x f(v) dv.
\end{aligned}$$

Med andra ord har den genererade X den sökta fördelningsfunktionen.

Risken med denna metod är att den kan kräva ett stort antal försök (generering av punkter från den likformiga fördelningen på R) innan ett värde accepteras. Notera att sannolikheten att acceptera paret (V,U) ges av

$$\mathrm{P}(U \le f(V)/m) = \int_a^b \int_0^{f(v)/m} du dv = \int_a^b \frac{f(v)}{m} dv = \frac{1}{m}.$$

Antalet försök N innan vi accepterar ett par har alltså en $\mathrm{Geom}(1/m)$-fördelning. Dess väntevärde är

$$\mathrm{E}[N] = (1/m)^{-1} = m.$$

Vi måste i genomsnitt alltså generera m utfall av (V,U) för att generera ett enda utfall av X enligt förkastelsemetoden. Därför är det viktigt att vi väljer m så litet som möjligt. Eftersom vi måste ha $m \ge f(x)$ för alla $x \in [a,b]$ får vi då

$$m = \max_{x \in [a,b]} f(x).$$

Ofta kan detta värde beräknas analytiskt. Om inte kan det approximeras genom att beräkna $f(x)$ på ett finmaskigt rutnät och ta det största värdet.

För en given täthetsfunktion $f(x)$ kan vi dock inte komma ifrån faktumet att fördelningar som är mer koncentrerade i en viss punkt (och därför har ett stort maxvärde) leder till mindre effektiva implementeringar av förkastelsemetoden. Detta är ganska naturligt då komplementet $R \setminus B$ är större och vi oftare hamnar utanför B när vi genererar punkter på R.

Ett sätt att effektivisera metoden att välja R så att följande kriterier uppfylls:

(i) Vi kan (enkelt) sampla från den likformiga fördelningen på R,

(ii) $B \subset R$ och

(iii) $R \setminus B$ är liten.

Antag att vi har en annan fördelning med täthetsfunktion $g(x)$ så att

$$f(x) \le mg(x)$$

för alla $x \in \mathbb{R}$ och något $m \ge 1$. Vi kan då definiera

$$R = \{(x,u) : 0 \le u \le mg(x)\}.$$

Eftersom $f(x) \le mg(x)$ har vi att $B \subset R$ och om m är litet så är komplementet $R \setminus B$ litet. Vi uppfyller alltså (ii) och (iii).

Om vi enkelt kan sampla från $g(x)$ uppfyller vi även (i). Vi tar då V från $g(x)$ och $U \sim \mathrm{U}(0,1)$ och bildar

$$(V, mg(V)U).$$

Enligt samma argument som för simuleringens fundamentalsats är detta par likformigt fördelad över R.

Om vi sätter ihop detta får vi följande algoritm:

(1) Generera V från $g(x)$ och $U \sim \mathrm{U}(0,1)$ oberoende.

(2) Om $mg(V)U \leq f(V)$, dvs. om $U \leq f(V)/mg(V)$, sätt $X = V$ och terminera, annars gå tillbaka till steg (1).

Detta kallas alltså *förkastelsemetoden* med *kandidatfördelning* $g(x)$ och *målfördelning* $f(x)$.

Återigen får vi att fördelningsfunktionen för X ges av fördelningsfunktionen för V betingad på händelsen $U \leq f(V)/mg(V)$, så vi har

$$\begin{aligned}
\mathrm{P}(X \leq x) &= \mathrm{P}(V \leq x \mid U \leq f(V)/mg(V)) \\
&= \frac{\mathrm{P}(V \leq x \mid U \leq f(V)/mg(V))}{\mathrm{P}(U \leq f(V)/mg(V))} \\
&= \frac{\int_{-\infty}^{x} \int_{0}^{f(v)/mg(v)} du g(v) dv}{\int_{-\infty}^{+\infty} \int_{0}^{f(v)/mg(v)} du g(v) dv} \\
&= \frac{\int_{-\infty}^{x} \frac{f(v)}{m} dv}{\int_{-\infty}^{+\infty} \frac{f(v)}{m} dv} = \frac{\int_{-\infty}^{x} \frac{f(v)}{m} dv}{\frac{1}{m}} = \int_{-\infty}^{x} f(v) dv
\end{aligned}$$

Vi får alltså den sökta fördelningsfunktionen.

Ett par aspekter av denna metod är värda att diskutera. Först så har vi inte längre någon begränsning av fördelningens stöd. Så länge $g(x)$:s stöd har $f(x)$:s stöd som delmängd fungerar algoritmen. (Det tidigare kravet att $f(x)$ måste ha ändligt stöd berodde på valet av kandidatfördelning $\mathrm{U}(a,b)$.)

Sedan har vi att sannolikheten att acceptera paret (V,U) nu ges av

$$\mathrm{P}(U \leq f(V)/mg(V)) = \int_{-\infty}^{+\infty} \int_{0}^{f(v)/mg(v)} du g(v) dv = \int_{-\infty}^{+\infty} \frac{f(v)}{m} dv = \frac{1}{m}.$$

Så antalet förväntade försök är $(1/m)^{-1} = m$. Vi vill alltså återigen välja m så litet som möjligt men så att $f(x) \leq mg(x)$ för alla $x \in \mathbb{R}$. Det naturliga valet är då

$$m = \sup_{x \in \mathbb{R}} \frac{f(x)}{g(x)}.$$

Som tidigare kan detta ofta beräknas analytiskt men vi kan också approximera m genom att utvärdera kvoten $f(x)/g(x)$ på ett finmaskigt rutnät och välja det största värdet.

En av de viktigaste egenskaperna hos förkastelsemetoden är att täthetsfunktionen $f(x)$ *behöver inte vara normaliserad* (men den måste vara integrerbar). Låt

$$f(x) = cq(x)$$

för någon $c > 0$ där $q(x)$ är en (normaliserad) täthetsfunktion. Då har vi

$$f(x) \leq mg(x) \Leftrightarrow q(x) \leq \frac{m}{x} g(x),$$

och acceptanskriteriet blir

$$U \leq \frac{f(V)}{mg(V)} \Leftrightarrow U \leq \frac{q(V)}{\frac{m}{c} g(V)}.$$

Med andra ord är det enda som spelar roll för algoritmen kvoten

$$\frac{f(V)}{m} = \frac{q(V)}{m/c},$$

vilken är densamma i båda fall. Med andra ord kan vi multiplicera $f(x)$ med en godtycklig konstant, men detta betyder bara att vi multiplicerar m med samma konstant. (Notera dock att resultatet ovan angående det genomsnittliga antalet försök m innan acceptans bara håller när $f(x)$ är en normaliserad täthetsfunktion.)

Denna sista punkt blir speciellt viktig när vi genererar utfall från aposteriorifördelningar. I dessa fall så vill vi inte alltid beräkna normaliseringsfaktorn för täthetsfunktionen utan uttrycker ofta denna genom

$$f_{\Theta \mid X}(\theta \mid x) \propto f_{X \mid \Theta}(x \mid \theta) f_{\Theta}(\theta).$$

Med förkastelsemetoden kan vi helt enkelt använda oss av högerledet som det är utan att behöva ta fram normaliseringsfaktorn.

12.3 Markovkedje-Monte Carlo

Även om förkastelsemetoden som presenterades ovan är mycket kraftfull har den ett par potentiella nackdelar. Den första är att valet av kandidatfördelning inte alltid är enkelt och utöver behöver vi dessutom beräkna $m = \sup_{x\in\mathbb{R}} f(x)/g(x)$. För vissa fördelningar med hög sannolikhet att ta stora värden är det inte heller alltid möjligt att hitta möjliga kandidatfördelningar så att $m < +\infty$. Det kan också vara så att m är ändlig men så pass stor att algoritmen inte är effektiv. Detta är vanligt när vi arbetar med fler dimensioner.

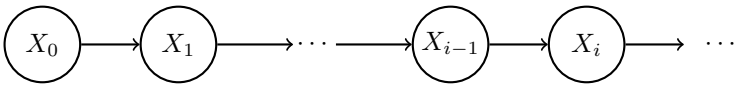
Ett alternativ till förkastelsemetoden är en familj av algoritmer som kallas *markovkedje-Monte Carlo* (på engelska *Markov chain Monte Carlo* eller MCMC). Dessa är mer flexibla och ställer inte samma typ av krav som förkastelsemetoden utan kan köras utan större förberedelser. Priset är att vi inte längre kan garantera att vi exakt generar utfall från den sökta täthetsfunktionen. MCMC skiljer sig alltså från inversions- och förkastelsemetoden som är *exakta* simuleringsmetoder utan ger en *approximativ* simulering. Oftast innebär inte detta några större problem, utan löses oftast genom mindre kalibreringssteg.

12.3.1 Markovkedjor

För att beskriva MCMC-metoder behöver vi introducera ett par begrepp. Först har vi en sekvens $X_0, X_1, \ldots$ av stokastiska variabler så att

$$X_i \perp\!\!\!\perp X_0, X_1, \ldots, X_{i-2} \mid X_{i-1}$$

för $i = 1, 2, \ldots$. Med andra ord är de stokastiska variablerna markovska med avseende på grafen



och uppfyller

$$\begin{aligned}&\mathrm{P}(X_i \le x_i \mid X_0 = x_0, X_1 = x_1, \ldots, X_{i-1} = x_{i-1})\\ &= \mathrm{P}(X_i \le x_i \mid X_{i-1} = x_{i-1}) = K(x_i, x_{i-1})\end{aligned}$$

för $i = 1, 2, \ldots$ där $K : \mathbb{R}^2 \to \mathbb{R}$. Med andra ord så beror X_i på tidigare tillstånd endast genom X_{i-1} och detta beroende är detsamma för alla i. Denna sekvens är alltså entydigt definierad av X_0:s fördelning samt $K(x, y)$ och kallas för en *markovkedja*.

Ett exempel på en markovkedja är

$$X_{i+1} = X_i + Z_i$$

där $Z_i \perp\!\!\!\perp X_0, X_1, \ldots, X_i$ och har samma fördelning för alla i. Detta kallas för en *slumpvandring*. Om Z_i har en symmetrisk fördelning kallas kedjan för en *symmetrisk slumpvandring*.

En vanlig egenskap hos markovkedjor är att de ofta har en *stationär fördelning* $p(x)$ så att

$$X_i \sim p(x) \Rightarrow X_{i+1} \sim p(x).$$

Om vi alltså börjar med $X_0 \sim p(x)$ så kommer alla element i kedjan ha samma fördelning $p(x)$.

Även om X_0 inte har den stationära fördelningen så gäller det ofta att X_i konvergerar till den stationära fördelningen när $i \to +\infty$. Många markovkedjor är också *ergodiska*, vilket betyder att

$$\mathrm{P}\left(\left|\frac{1}{k}\sum_{i=0}^{k} h(X_i) - \mathrm{E}[h(X)]\right| < \epsilon\right) \to 1$$

när $k \to +\infty$ för varje $\epsilon > 0$ där $X \sim p(x)$.

Denna sista egenskap är lik de stora talens lag, men skiljer sig väsentligt från denna. För det första så kräver stora talens lag att de stokastiska variablerna $X_1, X_2, \ldots$ är oberoende och likafördelade, men detta gäller inte här. Resultatet är alltså mer generellt än stora talens lag. Anledningen till att detta fungerar beror på att X_i konvergerar till den stationära fördelningen $p(x)$ när $i \to +\infty$.

Idén med MCMC är nu att konstruera en markovkedja så att den stationära fördelningen har den täthetsfunktion vi vill simulera. Om vi lyckas med detta kan vi approximera olika väntevärden med hjälp av ergodicitetsresultatet ovan.

Innan vi går in på detta ska vi bara notera att markovkedjor används också studeras för sin egen skull och har tillämpningar inom flera andra områden, som taligenkänning, sökmotorer, beräkningslingvistik, kompression, och optimering. Mer information kan fås i kursen SF1904.

12.3.2 Metropolisalgoritmen

Som innan, låt nu $f(x)$ vara täthetsfunktionen för någon målfördelning och låt $q(y \mid x)$ vara en betingad täthetsfunktion så att

$$q(y \mid x) = q(x \mid y)$$

(dvs. $q(y \mid x)$ är symmetrisk). Vi kallar $q(y \mid x)$ för en *hoppfördelning* (eller *förslagsfördelning*).

Vi skulle nu kunna definiera en markovkedja genom

$$X_i \mid X_{i-1} \sim q(x \mid X_{i-1}).$$

Denna kommer då konvergera till någon stationär fördelning som bara beror på $q(y \mid x)$. Eftersom vi vill att denna stationära fördelning ska ha en viss täthetsfunktion måste vi modifiera denna hoppfördelning på något sätt innan vi använder den i våran markovkedja.

Detta uppnås av *metropolisalgoritmen* som ges av

(1) Generera ett initalvärde X_0 (enligt någon fördelning)

(2) För $i = 1, 2, \ldots$:

  (2.1) Generera $Y \sim q(y \mid X_{i-1}$ (ett *förslag*)

  (2.2) Sätt $R = f(Y)/f(X_{i-1})$

  (2.3) Generera $U \sim \mathrm{U}(0, 1)$

  (2.4) Om $U \leq \min(1, R)$, sätt $X_i = Y$

  (2.5) Annars sätt $X_i = X_{i-1}$

Det finns ett par ytliga likheter med förkastelsemetoden här, men algoritmen skiljer sig på ett flertal viktiga sätt.

Först så beror förslaget Y på tidigare genererade värden. I förkastelsemetoden så var varje förslag oberoende av de andra. Denna länk med tidigare värden låter metropolisalgoritmen "utforska" olika delar av utfallsrummet en bit i taget. Eftersom förslaget beror på tidigare värden har vi också ett beroende mellan successiva utfall.

Sedan så resulterar varje iteration i ett genererat utfall. Med andra ord så har vi ett nytt värde X_i även om förslaget förkastas. Här är detta värde lika med det föregående, men det räknas fortfarande som ett "nytt" element i kedjan.

Vi kräver inte att hoppfördelningen $q(y \mid x)$ har någon speciell form med avseende på målfördelningen $f(x)$. På grund av detta behöver vi inte heller räkna ut någon konstant m för att algoritmen ska fungera. Med detta sagt så spelar formen på $q(y \mid x)$ roll för hur snabbt kedjan konvergerar till den stationära fördelningen.

En viktig likhet med förkastelsemetodne är däremot att $f(x)$ inte behöver vara normaliserad (men måste såklart vara integrerbar). Anledningen är att $f(x)$ endast återfinns på formen $f(Y)/f(X_{i-1})$, så någon möjlig normaliseringsfaktor tar ut sig själv. Som sagt är detta mycket användbart när vi vill generera utfall från en aposteriorifördelning utan att normalisera denna.

Så vad gör denna algoritm? I varje iteration genererar den ett förslag Y enligt $q(y \mid X_{i-1}$. Vi kan se detta som att den "hoppar" från ett tidigare värde X_{i-1} till något intilligande värde Y i utfallsrummet. Sedan bildar den *acceptanskvoten*

$$R = \frac{f(Y)}{f(X_{i-1})}$$

för att bestämma om förslaget ändrar sannolikheten jämfört med tidigare värde. Om $R \geq 1$ så har Y högre sannolikhet och vi accepterar alltid förslaget (eftersom $U \leq 1 = \min(1, R)$).

Å andra sidan om $R < 1$ så accepterar vi bara förslaget med sannolikhet R. Om sannolikheten går ner mycket med avseende på X_{i-1} är det alltså osannolikt att vi accepterar. En mindre minskning har däremot större chans att accepteras.

Om det inte var för denna andra möjlighet (dvs. att vi förflyttar oss till ett område med lägre sannolikhet) så skulle markovkedjan konvergera till ett lokalt maximum av $f(x)$ och stanna där. Möjligheten att "gå ner" i sannolikhet låter markovkedjan utforska en större del av utfallsrummet.

Nu påstår vi att den stationära fördelningen för markovkedjan som metropolisalgoritmen producerar har täthetsfunktion $f(x)$. Är detta sant?

För att visa detta krävs två egenskaper: existens och entydighet. Existens bevisas genom att visa att $f(x)$ är en stationär fördelning för kedjan. Med andra ord, givet $X_i \sim f(x)$ måste vi visa att $X_{i+1} \sim f(x)$. Detta reducerar till faktumet att $q(y \mid x) = q(x \mid y)$ och att sannolikheten att "gå ner" exakt kompenserar hoppfördelningens bidrag.

Entydighet kräver ett mer tekniskt argument och beror på olika egenskaper hos hoppfördelningen $q(y \mid x)$ som att denna ska vara tillräckligt ”bred” för att täcka stora delar av utfallsrummet.

I vårt fall kommer egenskaperna ovan att gälla för de fördelningar som vi betraktar i denna kurs. Vi har då

$$\frac{1}{k}\sum_{i=0}^{k} h(X_i) \to \mathrm{E}[h(X)]$$

och att X_i är approximativt fördelad enligt $f(x)$ när $k \to +\infty$.

12.3.3 Praktiska tips

De goda nyheterna om metropolisalgoritmen är att den konvergerar till målfördelningen för en bred uppsättning initialvärden och hoppfördelningar. De dåliga nyheterna är att denna konvergens kan ta mycket lång tid om man har otur. Som konsekvens av detta är det viktigt att se över konvergensen av algoritmen och se till att den ger vettiga värden innan vi börjar använda utfallen för inferens.

Det enklaste sättet är att plotta utfallen som funktion av iterationen. Om värdena rör sig mycket långsamt är det ett tecken på att hoppfördelningen är för begränsad, dvs. att dess skala är för liten. Om värdena är konstanta under en lång tid kan detta bero på att hoppfördelningen har för stor skala, eftersom alla förslag då hamnar i områden med låg sannolikhet.

En annan metod är att beräkna löpande medelvärden. Detta kommer att variera mycket i början av algoritmen och sedan stabilisera runt någon värde när den börjar konvergera.

Generellt sett kommer de första värdena i kedjan bero starkt på initialvärdet X_0 som använts. Därför approximerar dessa värden ganska dåligt målfördelningen och denna fas kallas ofta för ”uppvärmningsfasen”. En bra tumregel är att så fort kedjan verkar ha konvergerat så delar men den i två delar, kastar bort den första delen, och behåller den andra delen för att göra sina beräkningar.

Som vi poängterade ovan så är metropoliskedjan också starkt beroende elementen sinsemellan så om (approximativt) oberoende värden söks krävs en del efterbehandling. Det vanligaste sättet att åstadkomma detta är att behålla var m:te värde för något tillräckligt stort m (detta kan väljas genom att skatta korrelationen mellan element i kedjan). Observera att för många tillämpningar (som för att beräkna väntevärden) är detta inte nödvändigt eftersom summan $\frac{1}{k}\sum_{i=0}^{k} h(X_i)$ konvergerar ändå. Samma gäller för kredibilitetsintervall, histogram, osv. där vi kan använda värdena som de är.

En viktig metod för att studera konvergens hos metropolisalgoritmen är att köra flera kedjor parallelt. Varje kedja börjar då i olika startpunkter och vi kan då se hur mycket kedjorna ”blandas” (dvs. om de täcker samma område av utfallsrummet). När vi ser detta är det ett tecken på att kedjorna konvergerat.