Bayesianska numeriska metoder II

T. Olofsson

Gibb's sampling

Vi har sett att en viktig teknik vid Bayesiansk inferens är s.k marginalisering vilket, för kontinuerliga variabler, innebär att vi "integrerar bort" variabler som vi inte är intresserade av, s.k. nuissance-parametrar. I ett av våra tidigare exempel var vi tex endast intresserade av att uttala oss om parametern w_0 men modellen innehöll även den okända parametern w_1. Vi kunde erhålla $p(w_0|\mathcal{X},\mathcal{Y})$ genom att beräkna integralen $\int p(w_0,w_1|\mathcal{X},\mathcal{Y})dw_1$. Hade vi haft fler nuissance-parametrar, $w_2,...,w_K$, så hade vi behövt beräkna en multipelintegral

$$\int_{w_1} ... \int_{w_K} p(w_0,w_1,,...,w_K|\mathcal{X},\mathcal{Y})dw_1...dw_K. \tag{1}$$

Ett viktigt praktiskt problem är att multipla integraler är svåra att beräkna av flera skäl och vårt motmedel var här att använda oss av Monte–Carlo simuleringar. Integralen ovan kan approximeras numeriskt genom att

1. "Dra" ett stort antal vektorvärda sampel, $\mathbf{w}^1,\mathbf{w}^2,...,\mathbf{w}^M$, från fördelningen med PDF $p(w_0,w_1,,...,dw_K|\mathcal{X},\mathcal{Y})$. (Varje $\mathbf{w}^i$ består av en w_0-komponent, en w_1-komponent osv.)
2. Spara endast w_0-komponenterna i samplen. Bilda mängden $\{w_0^1,...,w_0^M\}$. Elementen i mängden $\{w_0^1,...,w_0^M\}$ har nu en fördelning med PDF $p(w_0|\mathcal{X},\mathcal{Y})$ och det står oss fritt att använda dessa som vi vill för att numeriskt ta fram en approximation av denna PDF.[1]

Nästa hinder som vi stötte på var att det brukar generellt vara svårt att dra sampel ur en godtycklig vektorvärd fördelning. (Ett viktigt undantag är de fördelningar där komponenterna i vektorn är statistiskt oberoende eftersom vi då kan dra en komponent i taget.) Det är däremot mycket enklare att dra sampel från någon godtycklig endimensionell fördelning, $p(w)$. Vi kan till exempel använda acceptance/rejection-metoden. Denna metod hade bland annat fördelen att det räcker med att vi känner till en funktion, $f(w)$, som PDF:en är *proportionell mot*, dvs vi behöver endast veta kurvformen, $f(w)$.

En metod som utnyttjar sampling från endimensionella fördelningar för att effektivt dra sampel från vektorvärda fördelningar är *Gibb's sampling*. Denna metod ingår i en större familj av metoder som går under samlingsnamnet *Markov Chain Monte–Carlo* (MCMC eller $(MC)^2$). Monte–Carlo-delen i metoderna har vi redan behandlat ovan. Namnet Markov Chain kommer sig av att samplen dras sekventiellt från fördelningen och det sampel som dras beror endast på det tidigare dragna samplet (jmfr Markovegenskapen). Förloppet kan beskrivas med hjälp av Markovkedjor och beviset för att metoderna verkligen ger oss sampel från rätt fördelning baserar sig på teori för stationära fördelningar för Markovkedjor. I ett delavsnitt nedan kommer vi att kortfattat gå igenom ett exempel som illustrerar vad det hela går ut på. Vi hoppar dock över den tyngre teorin.

Låt oss återgå till problemet att dra sampel från en fördelning med PDF $p(w_0,w_1,,...,w_K|\mathcal{X},\mathcal{Y})$ med hjälp av Gibb's sampling. Det går det till som följer:

- Ansätt startvärden $(w_0^0,...,w_K^0)$.
- Dra ett sampel w_0^1 ur den betingade fördelningen $p(w_0|w_1^0,,...,w_K^0,\mathcal{X},\mathcal{Y})$. Man brukar skriva $w_0^1 \leftarrow p(w_0|w_1^0,,...,w_K^0,\mathcal{X},\mathcal{Y})$ för att beskriva detta. Pilen representerar en slumpmässig tilldelning.

[1] Det kan kanske vara lite svårt att se vart marginaliseringsintegralen i Monte–Carlo simuleringen tog vägen. Faktum är att när vi kastar alla w_1- till w_K-komponenter och beräknar tex ett histogram över de kvarvarande w_0-värdena så är detta samma sak som att säga att vi *summerar* $\approx$ integrerar över alla vektorvärda sampel för vilka w_0-komponenterna ligger i något visst intervall.

- Dra ett sampel $w_1^1 \leftarrow p(w_1|w_0^1, w_2^0, ..., w_K^0, \mathcal{X}, \mathcal{Y})$. Notera att vi har stoppat tillbaks det nyss dragna samplet w_0^1 bakom "betingat-strecket".
- Allmänt, dra ett sampel $w_i^1 \leftarrow p(w_i|w_0^1, w_1^1, ..., w_{i-1}^1, w_{i+1}^0, ..., w_K^0, \mathcal{X}, \mathcal{Y})$.
- När vi dragit den sista skalären w_K^1, bilda $\mathbf{w}^1 = ({w_0}^1, ..., {w_K}^1)^T$ som får utgöra det första vektorvärda samplet.
- Upprepa förloppet cykliskt för komponenter $w_0, w_1, , ..., w_K$. Den allmänna formeln under iteration nr k blir $w_i^k \leftarrow p(w_i|w_0^k, w_2^k, ..., w_{i-1}^k, w_{i+1}^{k-1}, ..., w_K^{k-1}, \mathcal{X}, \mathcal{Y})$. Varje gång $i = K$ är uppfyllt, dvs då vi gått igenom en cykel, bilda samplet $\mathbf{w}^k = (w_0^k, ..., w_K^k)^T$.

För att de vektorvärda sampel vi får ur denna algoritm skall gälla som dragna ur den eftersökta fördelningen krävs bland annat att algoritmen har kommit in i sin *stationära fas*. Denna fas inträder först efter en initialfas som brukar kallas "burn in". Man brukar därför bortse från de första samplen (det är dock svårt att säga exakt hur många detta ska vara) när man utnyttjar data från denna algoritm.

Det brukar vara relativt lätt att tro på att ovanstående schema faktiskt ger oss sampel från rätt fördelning om man ritar upp en tvådimensionell fördelning och tänker sig hur algoritmen skulle fungera i detta enkla fall. Se figurerna 1 till 3 nedan.

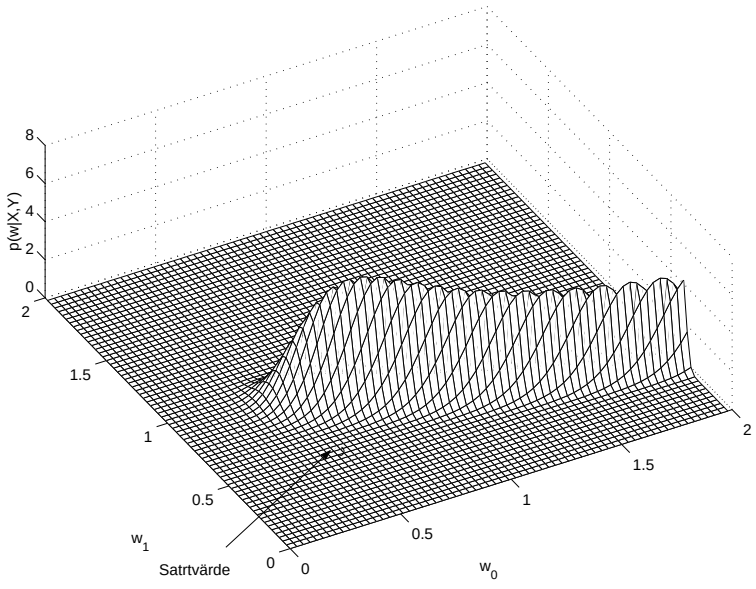

Figure 1: Vi ska med hjälp av Gibb's sampling dra sampel från denna 2-d fördelning. Startvektorn är markerad med en pil.

Vi ser ur schemat att vi endast behöver dra sampel från endimensionella fördelningar. *Kurvformen* för dessa fördelningar kommer vi lätt åt via Bayes sats:

$$\begin{aligned} &p(w_i|w_0^k, w_2^k, ..., w_{i-1}^k, w_{i+1}^{k-1}, ..., w_K^{k-1}, \mathcal{X}, \mathcal{Y}) = \\ =&\frac{p(\mathcal{Y}|w_0^k, w_1^k, ..., w_{i-1}^k, w_i, w_{i+1}^{k-1}, ..., w_K^{k-1}, \mathcal{X}) p(w_i|w_0^k, w_1^k, ..., w_{i-1}^k, w_{i+1}^{k-1}, ..., w_K^{k-1}, \mathcal{X})}{p(\mathcal{Y}|w_0^k, w_1^k, ..., w_{i-1}^k, w_{i+1}^{k-1}, ..., w_K^{k-1}, \mathcal{X})} \end{aligned} \quad (2)$$

Observera att vi bör se uttrycket i ekv. (2) *som en funktion av* w_i. (Notera att alla parametrar utom w_i antar fixa värden som erhållits under tidigare iterationer.) Nämnaren är som vanligt en normeringskonstant och innehåller inte w_i, dvs den påverkar inte kurvformen. Med hjälp av tex acceptance/rejection-metoden kan vi nu dra ett ett sampel från $p(w_i|w_0^k, w_1^k, ..., w_{i-1}^k, w_{i+1}^{k-1}, ..., w_K^{k-1}, \mathcal{X}, \mathcal{Y})$.

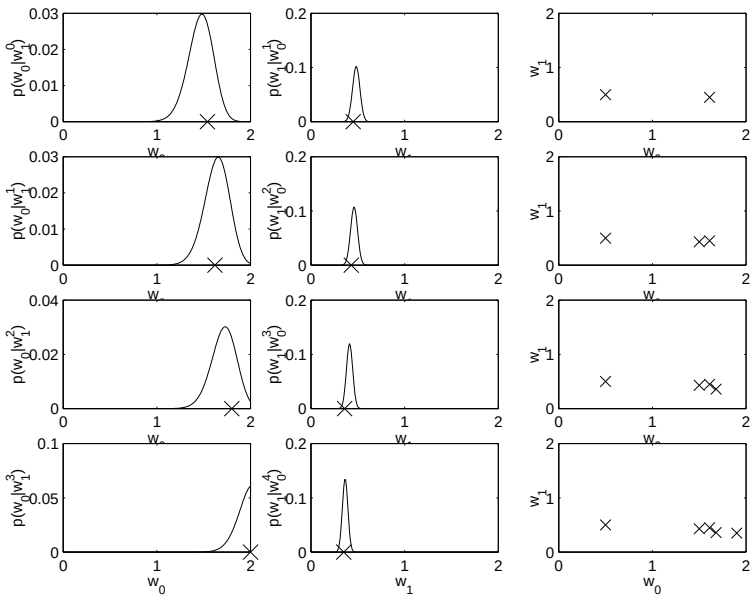

Figure 2: Fyra iterationer av algoritmen. I varje iteration dras först ett w_0-värde från den betingade fördelningen $p(w_0|w_1^{k-1})$ (vänstra kolumnen). Det dragna värdet markeras här med ett kryss. Notera att den betingade fördelningen erhålls "grafiskt" genom att skära ett snitt genom den tvådimensionella fördelningen i fig. 1 för ett fixerat $w_1 = w_1^{k-1}$. Givet det nya värdet, w_0^k, dras ett nytt w_1 värde från den betingade fördelningen $p(w_1|w_0^k)$. Det dragna värdet markeras återigen med ett kryss. Den nya vektorn (w_0^k, w_1^k) plottas till sist i ett spridningsdiagram i den högra kolumnen (tillsammans med tidigare dragna vektorer).

En beskrivning av Gibb's sampling med hjälp av Markovkedor. Diskreta fördelningar.

Även om det rent intuitivt verkar troligt att Gibbs sampling fungerar så är det ju för den skull inte självklart att metoden verkligen garanterar vektorvärda sampel dragna från exakt rätt fördelning. I detta delavsnitt kommer vi visserligen inte att bevisa att metoden har denna egenskap men vi kommer med hjälp av ett enkelt exempel att dels illustrera hur metoden kan beskrivas av en Markovkedja och visa att Markovkedjan som motsvarar det valda exemplet genererar sampel från exakt rätt fördelning. För att kunna göra detta måste vi dock först förbereda oss med lite extra teori för Markovkedjor.

Låt oss rekapitulera Markovmodellen. Vi har ett antal tillstånd, $q_1, ..., q_M$, och en $M \times M$ matris, $\mathbf{T}$, med övergångssannolikheter $t_{ij} = P(S^{t+1} = q_i | S^t = q_j)$ som element. Anta att vi låter en Markovmodell "snurra" under lång tid och är intresserade av hur ofta ett visst tillstånd besöks i allmänhet under stationäritet.

Tänk er först att vi startar modellen, vid tiden $t = 0$, genom att slumpmässigt välja starttillstånd. Låt oss definiera vektorn ϕ^0 enligt

$$\phi^0 = (P_1, ..., P_M)^T \qquad \text{(kolumnvektor)} \tag{3}$$

där initialsannolikheten P_i står för

$$P_i = P(S^0 = q_i). \tag{4}$$

Vilken är sannolikheten att befinna sig i tillstånd q_j vid tiden $t = 1$, dvs efter en tillståndsövergång? Svaret får vi med hjälp av summaregeln:

$$P(S^1 = q_j) = \sum_i P(S^1 = q_j, S^\circ = q_i) = \sum_i P(S^1 = q_j | S^\circ = q_i) P(S^\circ = q_i) = \sum_i t_{ji} P_i. \tag{5}$$

Om vi placerar dessa sannolikheter i en kolumnvektor $\phi^1 = (P(S^1 = q_1), P(S^1 = q_2), ..., P(S^1 = q_M))^T$ så kan vi skriva ekv. (5) på matrisform som

$$\phi^1 = \mathbf{T}\phi^0. \tag{6}$$

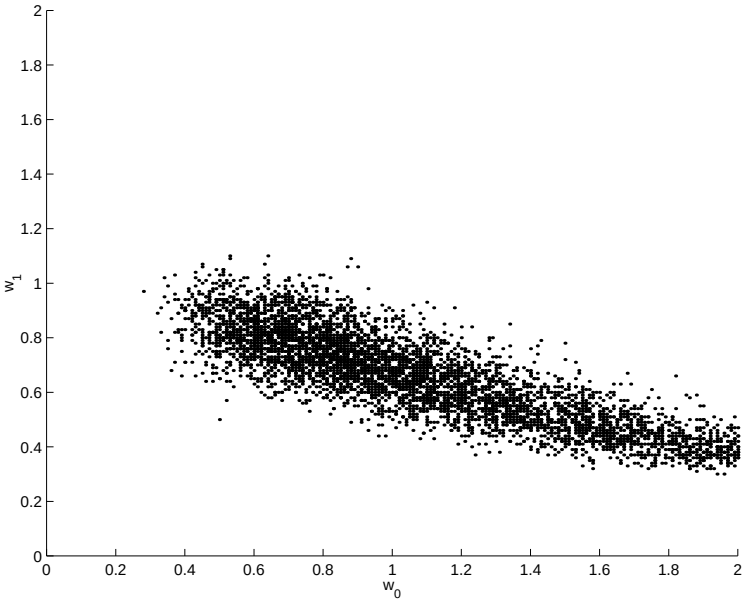

Figure 3: Spridningsdiagram som visar 5000 vektorsampel dragna med hjälp av Gibb's sampling från fördelningen i figur 1.

Det är sen lätt att inse att denna formel kan appliceras rekursivt för att beräkna sannolikhetsfördelningar för godtyckliga t. Vi kommer fram till dels

$$\phi^{t+1} = \mathbf{T}\phi^t \qquad \text{och} \qquad \phi^t = \mathbf{T}^t\phi^0 \tag{7}$$

En Markovkedja sägs vara i sin stationära fas då fördelningen inte ändras från iteration till iteration, dvs då $\phi^{t+1} = \phi^t$. Fördelningen ϕ är då stationär (dvs beror ej av t) och om vi betecknar denna stationära fördelning[2] med ϕ^* så måste följande ekvation gälla för ϕ^*:

$$\phi^* = \mathbf{T}\phi^*. \tag{8}$$

Observera att denna ekvation är en egenvektorsekvation där egenvärdet är 1. Slutsatsen är alltså att vi kan ta fram den sannolikhetsfördelning för tillstånden som vi skulle få genom att låta modellen iterera under lång tid genom att helt enkelt beräkna den egenvektor till matrisen $\mathbf{T}$ som svarar mot egenvärdet 1.

Exempel: Stationär fördelning för Markovkedja med fyra tillstånd

Betrakta en Markovmodell med övergångsmatrisen

$$\mathbf{T} = \begin{pmatrix} 0.1 & 0.4 & 0.2 & 0.8 \\ 0.1 & 0.2 & 0.1 & 0.05 \\ 0.3 & 0.2 & 0.4 & 0.1 \\ 0.5 & 0.2 & 0.3 & 0.05 \end{pmatrix} \tag{9}$$

Ett exempel på sekvens som kan genereras av denna modell är

$$S = \{1, 4, 4, 1, 3, 1, 4, 1, 4, 1, ...\} \tag{10}$$

I figur 4 visas ett antal histogramskattningar av den stationära sannolikhetsfördelningen. De baseras på samma sekvens men olika antal tillstånd har använts vid beräkningen av histogrammen. Den sista grafen

[2] Kallas ibland även jämviktsfördelning

visar den teoretiskt beräknade stationära fördelningen som baseras på egenvektorsberäkningen enligt ekv. (8).[3]

I figuren ser vi tydligt att den teoretiskt beräknade stationära fördelningen sammanfaller i princip exakt med det histogram vi erhåller efter $t = 100000$, vilket i sammanhanget bör gälla som en mycket god approximation av den faktiska stationära födelningen. □

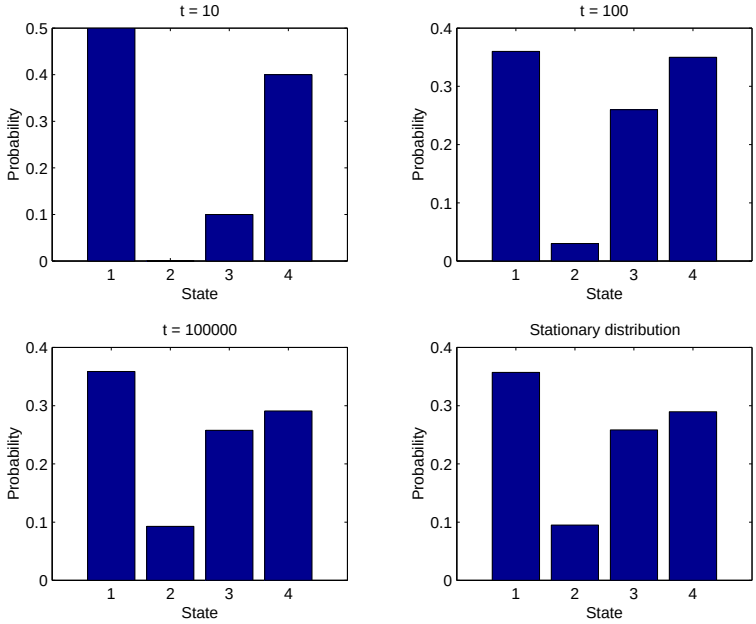

Figure 4: Jämförelse mellan histogramskattningar av sannolikhetsfördelningen för tillstånden i Markovmodellen med övergångsmatrisen $\mathbf{T}$ och den teoretiskt beräknade stationära fördelningen.

Nu har vi sett att en Markovkedja med given övergångsmatris har en stationär fördelning och dessutom något om hur vi kan beräkna denna fördelning. Låt oss nu vända på steken och ställa oss frågan hur vi kan skapa en Markovkedja så att den i sin stationära fas genererar sampel från någon önskad fördelning. Detta scenario är precis det vi hade tidigare. Vi kände till utseendet på någon PDF och ville dra sampel från denna men visste inte riktigt hur vi skulle göra. Alla metoder som ingår i MCMC-familjen baseras på idén att skapa en lämplig Markovkedja som ger oss den stationära fördelning vi vill ha. Det som skiljer metoderna åt är helt enkelt hur man sätter ihop en lämplig Markovkedja, och därmed implicit den associerade övergångsmatrisen, $\mathbf{T}$. Det är lätt att inse att flera Markovkedjor kan ha samma stationära fördelning (på samma sätt som flera matriser kan ha samma egenvektorer, jmfr ekv. (8)). Den kedja som vi får med Gibb's sampling är alltså bara en i mängden av många möjliga.

Exempel: Gibb's sampling av 2-d binär vektor

Tidigare beskrev vi hur Gibb's sampling gick till för kontinuerliga variabler, $w_1, ..., w_K$. För att se en tydlig koppling mellan Gibb's sampling och en specifik Markovkedja så är det dock betydligt enklare att betrakta ett enkelt fall för diskreta variabler. Vi väljer det enklast möjliga fallet med en tvådimensionell vektor, $\mathbf{x}$, bestående av två binära variabler, x_1 och x_2. Vi har alltså fyra tänkbara realiseringar av $\mathbf{x}$ så den totala sannolikhetsfördelningen beskrivs fullständigt av de fyra sannolikheterna

$$P_{00} = P(x_1 = 0, x_2 = 0), \;\; P_{01} = P(x_1 = 0, x_2 = 1), \;\; P_{10} = P(x_1 = 1, x_2 = 0), \text{ och } P_{11} = P(x_1 = 1, x_2 = 1) \tag{11}$$

[3]Man måste här tänka på att egenvektorer inte är entydigt bestämda och de som beräknas via numeriska metoder kan peka i den negativa riktningen. Dessutom är det vanligt i numeriska paket att normera vektorerna med avseende på kvadratnormen. Vi vill däremot normera ϕ^* så att alla komponenter är positiva (eller noll) samt att *summan* av elementen är 1 (sannolikheter).

Gibb's sampling från denna fördelning går till så att vi låser x_2 och drar ett sampel från den betingade fördelningen $P(x_1|x_2)$. Det värde på x_1 vi får vid denna dragning håller vi sen fixt när vi drar ett sampel från den betingade fördelningen $P(x_1|x_2)$. Schemat itereras sedan.

De betingade fördelningarna, $P(x_2|x_1)$ och $P(x_2|x_1)$, fås enkelt genom en omskrivning av produktregeln

$$P(x_1|x_2) = \frac{P(x_1, x_2)}{P(x_2)} \quad \text{samt} \quad P(x_2|x_1) = \frac{P(x_1, x_2)}{P(x_1)} \tag{12}$$

Nämnarna $P(x_1)$ och $P(x_2)$ får vi i sin tur via summaregeln (marginalisering). Till exempel får vi att

$$P(x_2 = 1) = P(x_2 = 1, x_1 = 0) + P(x_2 = 1, x_1 = 1) \tag{13}$$

Låt oss exempelvis betrakta fallet då vi har följande simultana sannolikheter:

$$P_{00} = 0.5, \quad P_{01} = 0.2, \quad P_{10} = 0.2, \quad P_{11} = 0.1 \tag{14}$$

Låt tillstånden $q_1, ..., q_4$ motsvara

$$q_1 ='' \, x_1 = 0, x_2 = 0'', \quad q_2 ='' \, x_1 = 0, x_2 = 1'', \quad q_3 ='' \, x_1 = 1, x_2 = 0'', \quad q_4 ='' \, x_1 = 1, x_2 = 1''. \tag{15}$$

Beroende på i vilken fas under samplingsalgoritmen vi befinner oss (sampling av x_1 eller x_2) kommer vi att hoppa mellan dessa olika tillstånd med olika sannolikheter. Vi är dock intresserade av sannolikheten att hoppa mellan tillstånden via kombinationen av båda dessa steg.

Ett naturligt och förhållandevis enkelt sätt att komma fram till vilken övergångsmatrisen $\mathbf{T}$ är, är att dela upp hela steget från $\mathbf{x}^k$ till $\mathbf{x}^{k+1}$ i två steg, nämligen $x_1^{k+1} \leftarrow P(x_1|x_2^k)$ och $x_2^{k+1} \leftarrow P(x_2|x_1^{k+1})$. De två separata stegen kan enkelt beskrivas med varsin övergångsmatris, $\mathbf{T}_1$ och $\mathbf{T}_2$, med de respektive utseendena

$$\mathbf{T}_1 = \begin{pmatrix} P(x_1 = 0|x_2 = 0) & 0 & P(x_1 = 0|x_2 = 0) & 0 \\ 0 & P(x_1 = 0|x_2 = 1) & 0 & P(x_1 = 0|x_2 = 1) \\ P(x_1 = 1|x_2 = 0) & 0 & P(x_1 = 1|x_2 = 0) & 0 \\ 0 & P(x_1 = 1|x_2 = 1) & 0 & P(x_1 = 1|x_2 = 1) \end{pmatrix} \tag{16}$$

och

$$\mathbf{T}_2 = \begin{pmatrix} P(x_2 = 0|x_1 = 0) & P(x_2 = 0|x_1 = 0) & 0 & 0 \\ P(x_2 = 1|x_1 = 0) & P(x_2 = 1|x_1 = 0) & 0 & 0 \\ 0 & 0 & P(x_2 = 0|x_1 = 1) & P(x_2 = 0|x_1 = 1) \\ 0 & 0 & P(x_2 = 1|x_1 = 1) & P(x_2 = 1|x_1 = 1) \end{pmatrix} \tag{17}$$

Nollorna i de båda matriserna kommer sig helt enkelt av att vi inte tillåts byta tillstånd på den variabel som vi för tillfället håller låst.

Den totala övergångsmatrisen, $\mathbf{T}$, får vi sen genom produkten (jämför tex ekv. (5))

$$\mathbf{T} = \mathbf{T}_2\mathbf{T}_1. \tag{18}$$

I vårt exempel får vi

$$\mathbf{T} = \begin{pmatrix} 0.5102 & 0.5102 & 0.4762 & 0.4762 \\ 0.1905 & 0.1905 & 0.2222 & 0.2222 \\ 0.2041 & 0.2041 & 0.1905 & 0.1905 \\ 0.0952 & 0.0952 & 0.1111 & 0.1111 \end{pmatrix} \tag{19}$$

Beräknar vi nu den egenvektor som svarar mot egenvärdet 1 så får vi den stationära fördelningen

$$\phi = (0.5, 0.2, 0.2, 0.1)^T \tag{20}$$

Vilket är precis den fördelning vi eftersökte. Vi kan enkelt kontrollera att det stämmer genom att ta produkten $\mathbf{T}\phi$ vilket visar sig bli ϕ.

Simulated annealing

En tillämpning av MCMC, eller den variant vi fokuserat på nämligen Gibb's sampling, är *global optimering*. Det vi kan åstadkomma med Markovkedjan är endast att kunna dra sampel från någon godtycklig fördelningsfunktion så vad har detta med optimering att göra?

Idén med optimerinsgmetoden simulated annealing (SA) är följande:

1. För en given funktion, $f(\mathbf{w})$, som vi vill minimera,[4] skapa en associerad PDF, $p(\mathbf{w})$, som har egenskapen att den har sitt globala maximum, $\mathbf{w}*$, på samma ställe som $f(\mathbf{w})$ har sitt minimum, dvs

$$\arg\max_{\mathbf{w}} p(\mathbf{w}) = \arg\min_{\mathbf{w}} f(\mathbf{w}). \tag{21}$$

2. Modifiera långsamt $p(\mathbf{w})$ till att en serie av PDF:er som succesivt konvergerar mot en PDF som är noll överallt utom i $\mathbf{w}^*$ där den har en "spik". Dra hela tiden sampel från denna långsamt förändrade PDF. Eftersom PDF:en succesivt går mot att vara i princip en enda skarp topp kring ett enda värde, $\mathbf{w}^*$, så måste i så fall det sampel vi drar från denna skarpa fördelning vara just $\mathbf{w}^*$.

Den associerade PDF vi använder i simulated annealing är

$$p_T(\mathbf{w}) = \frac{1}{Z} e^{-\frac{f(\mathbf{w})}{T}} \tag{22}$$

där $T > 0$ står för "temperatur"[5] och Z är en normeringskonstant som garanterar att vår PDF integrerar till 1.

Vi noterar först att $p(\mathbf{w})$ har sitt maximum i samma punkt som $f(\mathbf{w})$ har sitt minimum. Detta gäller oavsett vilket värde T har. Vi noterar för övrigt att $p_T(\mathbf{w})$ kan relateras till $p_1(\mathbf{w})$ (fördelningen då $T = 1$) genom proportionaliteten

$$p_T(\mathbf{w}) \propto p_1(\mathbf{w})^{\frac{1}{T}}. \tag{23}$$

Alltså, funktionsutseendet (normeringskonstanten borträknad) ges av $p_1(\mathbf{w})^{1/T}$. Om $p_T(\mathbf{w})$ för ett stort T har en utsträckt topp så kommer $p_T(\mathbf{w})$ för ett litet T att vara mer isolerad (mer utkristalliserad).

Simulated annealing går till enligt schemat

1. Sätt T till ett stort värde.

2. Dra sampel från $p(\mathbf{w})$ under det att T sänks (långsamt).

Man kan visa att detta schema, under relativt allmänna villkor, konvergerar "med sannolikhet gränsande till 1" till det *globala* minimat till funktionen $f(\mathbf{w})$. Tyvärr baserar sig beviset på ett "avkylningsschema" som i de flesta praktiska situationer är outhärdligt långsamt, nämligen

$$T^k = \frac{K}{\log k}, \tag{24}$$

där K är en konstant och k är iterationsnumret. I praktiska situationer används oftare varianten

$$T^k = \mu T^{k-1} \tag{25}$$

vilket ger en temperatur som avtar exponentiellt med iterationsnumret. Sen håller man tummarna och hoppas att det ska fungera ändå.

[4] Som vanligt skulle vi lika gärna kunna formulera optimeringsproblemet som ett *maximeringsproblem*. Vi får då modifiera resonemanget därefter.

[5] SA kommer historiskt sett från statistisk mekanik. I denna litteratur brukar man i nämnaren i exponenten även ha en faktor k som står för "Boltzmanns konstant". Vi kan här välja att sätta k till 1.