

UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

MASTER RAD

---

# Monte Karlo metode zasnovane na Markovljevima lancima i neke primene u statističkom učenju

---

Autor:

Katarina Svorcan

Mentor:

doc. dr Bojana Milošević



# Sadržaj

<b>1</b>	<b>Uvod</b>	<b>2</b>
<b>2</b>	<b>Osnovni koncepti</b>	<b>3</b>
2.1	Monte Karlo metode . . . . .	3
2.1.1	Uzorkovanje odbacivanjem . . . . .	3
2.1.2	Uzorkovanje značajnih regiona . . . . .	4
2.2	Markovljevi lanci . . . . .	5
2.3	Bajesovsko odlučivanje . . . . .	6
<b>3</b>	<b>MCMC algoritmi uzorkovanja</b>	<b>7</b>
3.1	Gibsov algoritam . . . . .	7
3.1.1	Hamersli Kliford teorema . . . . .	8
3.1.2	Konvergencija . . . . .	9
3.2	Metropolis-Hejstings algoritam . . . . .	11
3.2.1	Konvergencija . . . . .	12
3.2.2	Jednostavan primer MH algoritma . . . . .	13
3.3	Hamiltonovski Monte Karlo algoritam . . . . .	14
<b>4</b>	<b>Nedostaci MCMC algoritama</b>	<b>16</b>
4.1	Izbor predložene raspodele i inicijalnih parametara . . . . .	16
4.2	Odbacivanje početnih uzoraka . . . . .	17
4.3	Korelacija uzoraka . . . . .	18
4.4	Konvergencija ka lokalnim optimumima . . . . .	20
4.5	Adaptivni MCMC algoritmi . . . . .	20
<b>5</b>	<b>Primena na ocenjivanje parametara statističkih modela</b>	<b>21</b>
5.1	Ocenjivanje parametara linearne regresije . . . . .	21
5.2	Ocenjivanje parametara mešavine normalnih raspodela . . . . .	30
<b>6</b>	<b>Zaključak</b>	<b>32</b>
<b>A</b>	<b>Dodatak A: Pregled biblioteke TensorFlow probabilty i primeri</b>	<b>33</b>
A.1	Primena na ocenjivanje parametara linearne regresije . . . . .	33
A.2	Primena na ocenjivanje parametara mešavine Gausovih raspodela . . . . .	36
A.3	Implementacija HMC u Pajtonu . . . . .	38

# 1 Uvod

Problem analitičkog računanja aposteriorne raspodele je čest problem u bajesovskoj statistici. Monte Karlo metode zasnovane na Markovljevim lancima (eng. *Markov Chain Monte Carlo*, skraćeno *MCMC*) su jedna klasa algoritama koji rešavaju navedeni problem. Kreiranjem uzoraka koji formiraju Markovljev lanac sa stacionarnom raspodelom koja je jednaka traženoj raspodeli, mogu se dobiti uzorci tražene raspodele beležeći stanja Markovljevog lanca.

Iako su nastali sredinom prošlog veka, razvijaju se ubrzano tek 90ih godina sa povećanom potrebom za ovakvim algoritmima, kao i povećanim kompjuterskim resursima za izvršavanje zahtevnih simulacija. Taj razvoj i dalje traje, sa konstantnim unapređenjima osnovnih algoritama.

Rad predstavlja pregled osnovnih MCMC algoritama sa akcentom na primeni na ocenjivanje parametara statističkih modela, kao i praktičnim savetima radi dobijanja boljih performansi modela.

Najpre ćemo u poglavlju 2 napraviti pregled znanja potrebnih za uvođenje MCMC. Izdvojićemo relevantne metode uzorkovanja, koje su preteča MCMC algoritama, radi boljeg razumevanja koncepta uzorkovanja. Zatim ćemo nabrojati osnovna svojstva Markovljevih lanaca, koja će nam biti potrebna u narednim poglavljima. Na kraju napravićemo uvod u bajesovsko odlučivanje i statističko učenje.

Poglavlje 3 obrađuje 2 osnovna MCMC algoritma - Gibsovo uzorkovanje i Metropolis-Hejstings algoritam. Dokazaćemo konvergenciju za oba algoritma. Biće navedeni osnovni primeri radi bolje ilustracije samog procesa uzorkovanja. Pored toga obradićemo i malo napredniji algoritam Hamiltonovski Monte Karlo i prikazati koja unapređenja donosi u odnosu na prethodno navedene.

Poglavlje 4 bavi se ograničenjima i nedostacima MCMC algoritama, kao i potencijalnim rešenjima navedenih izazova. Spomenućemo različita unapređenja MCMC algoritama koristeći savete iz prakse.

Poglavlje 5 bliže opisuje algoritme statističkog učenja koji će se koristiti u radu, kao i detaljan prikaz primene MCMC na ocenu parametara navedenih algoritama.

U dodatku A nalazi se prateći kod za sve primere u radu u programskom jeziku Pajton (eng. Python), kao i kratak prikaz biblioteke TensorFlow Probability u jeziku Pajton.

## 2 Osnovni koncepti

### 2.1 Monte Karlo metode

Monte Karlo metode čini skup algoritama čiji je cilj da iterativnim ponavljanjima slučajnih eksperimenata ocene neku numeričku vrednost koju nije moguće odrediti analitičkim putem (npr. vrednost integrala). Osnovne primene su:

- Uzorkovanje raspodele,  $p(x)$
- Ocenjivanje integrala  $\int f(x)p(x)dx$ , odnosno  $E(f(x))$  u odnosu na raspodelu  $p$

Ova dva slučaja su usko povezana, jer rešavajući prvi slučaj, možemo da rešimo i drugi.

Ideja Monte Karlo simulacija je da izvučemo skup nezavisnih jednako raspodeljenih uzoraka  $x^{(i)}, i = 1, \dots, N$  iz željene raspodele  $p(x)$  definisane na  $n$ -dimenzionom prostoru  $\mathbb{X}$ . Navedenih  $N$  uzoraka mogu se iskoristiti za aproksimaciju gustine na sledeći način

$$(1) \quad p_N(x) = \frac{1}{N} \sum_{i=1}^N \delta_{x^{(i)}}(x),$$

gde je  $\delta_{x^{(i)}}(x)$  Dirakova delta funkcija u  $x^{(i)}$ <sup>1</sup> Nadalje, moguće je aproksimirati integrale  $I(f)$ :

$$(2) \quad I_N(f) = \frac{1}{N} \sum_{i=1}^N f(x^{(i)}) \xrightarrow[N \rightarrow \infty]{s.s.} I(f) = \int f(x)p(x)dx,$$

tj.  $I_N(f)$  je nepristrasna i po zakonu velikih brojeva važi skoro sigurna konvergencija ka  $I(f)$ .

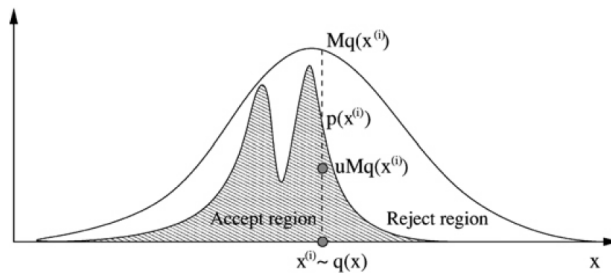
Prednost Monte Karlo metoda u odnosu na determinističke metode leži u tome što se Monte Karlo fokusira na regione visoke verovatnoće.

#### 2.1.1 Uzorkovanje odbacivanjem

Kada raspodela  $p$  nema standardan, poznat oblik, potrebno je definisati sofisticiranije tehnike uzorkovanja. Jedna od njih je uzorkovanje odbacivanjem (eng. *rejection sampling*). Neka je  $p(x)$  željena raspodela, koja nam je poznata do na konstantu. Definišemo predloženu raspodelu  $q(x)$  iz koje možemo direktno da uzorkujemo i čiji nam je oblik poznat, takvu da važi  $p(x) \leq Mq(x), M < \infty$ .

Neka je početna vrednost  $i = 1$ . Iterativno ponavljamo navedene korake dok i ne bude jednako  $N$ .

1. Uzorkujemo  $x^{(i)} \sim q(x)$  i  $u \sim U_{(0,1)}$
2. Ako je  $u < \frac{p(x^{(i)})}{Mq(x^{(i)})}$  tada prihvatamo  $x^{(i)}$  i uvećavamo  $i$  za 1. U suprotnom, uzorak se odbija.



Slika 1: (Preuzeto iz rada [1]): Ilustracija uzorkovanja odbacivanjem: Uzorkuje se kandidat  $x^{(i)}$  i vrednost iz uniformne raspodele  $u$ . Kandidat se prihvata samo u slučaju  $uMq(x^{(i)}) < p(x^{(i)})$

---

<sup>1</sup> $\delta_{x^{(i)}}(x) = \begin{cases} 1, & \text{za } x = x^{(i)} \\ 0, & \text{za } x \neq x^{(i)} \end{cases}$

Ovaj metod ima par ograničenja. Nije uvek moguće ograničiti  $p(x)/q(x)$  sa „razumnom” konstantom  $M$  na celom prostoru  $X$ . Ukoliko je  $M$  mnogo veliko, tada je verovatnoća prihvatanja uzorka

$$P\left(u < \frac{p(x)}{Mq(x)}\right) = \frac{1}{M}$$

previše mala, čime metod postaje neupotrebljiv u višedimenzionim prostorima.

### 2.1.2 Uzorkovanje značajnih regiona

Alternativno rešenje je uzorkovanje značajnih tačaka (eng. *importance sampling*). Ponovo uvodimo predloženu raspodelu  $q(x)$ , takvu da njen nosač sadrži i nosač  $p(x)$ . Tada jednačina 2 postaje:

$$(3) \quad I(f) = \int f(x)w(x)q(x)dx,$$

gde je  $w(x) := \frac{p(x)}{q(x)}$  težina značajnosti (eng. *significance weight*) Ukoliko je moguće izvući  $N$  nezavisnih jednako raspodeljenih uzoraka iz  $q(x)$  i izračunati vrednosti  $w(x)$ , Monte Karlo ocena  $I(f)$  bila bi

$$(4) \quad \hat{I}_N(f) = \sum_{i=1}^N f(x^{(i)})w(x^{(i)})$$

Ovo je nepristrasna ocena i na osnovu zakona velikih brojeva važi  $\hat{I}_N(f) \xrightarrow[N \rightarrow \infty]{s.s.} I(f)$ .

Dakle, prvo uzimamo predloženu raspodelu i generišemo skup nezavisnih jednako raspodeljenih uzoraka. Zatim svakom uzorku dodeljujemo težinu značajnosti i te težine se koriste za ocenu raspodele.

Neke raspodele  $q(x)$  biće pogodnije od drugih. Bitan kriterijum za izbor optimalne predložene raspodele je da izabrana raspodela minimizuje disperziju ocene  $\hat{I}_N(f)$ . Disperzija  $f(x)w(x)$  u odnosu na  $q(x)$  glasi

$$(5) \quad D_{q(x)}(f(x)w(x)) = E_{q(x)}(f^2(x)w^2(x)) - I^2(f)$$

Drugi deo ne zavisi od  $q(x)$ , a prvi deo se može ograničiti na osnovu Jensenove nejednakosti na sledeći način

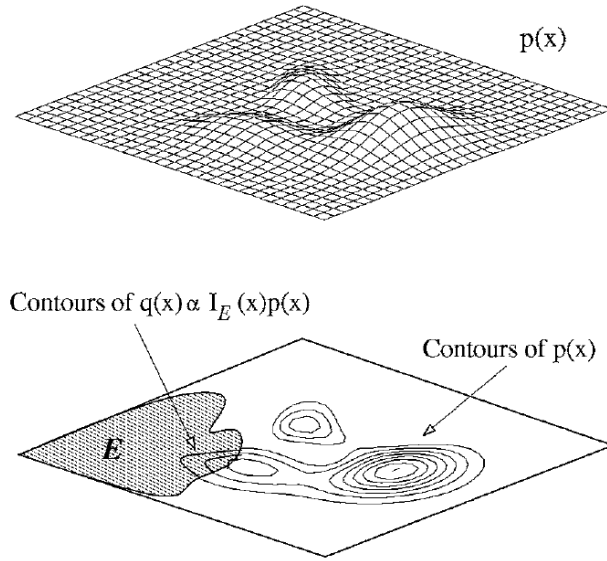
$$(6) \quad E_{q(x)}(f^2(x)w^2(x)) \geq (E_{q(x)}(|f(x)|w(x)))^2 = \left( \int |f(x)|p(x)dx \right)^2$$

Jednakost se dostiže za raspodelu optimalne značajnosti

$$(7) \quad q^*(x) = \frac{|f(x)|p(x)}{\int |f(x)|p(x)dx}$$

**Primer 2.1.** (Smith, Shaffi, Gao 1997) Potrebno je izračunati verovatnoću retkih događaja  $E$  u komunikacionim mrežama. Vrednost od interesa ima verovatnoću na repovima raspodele, pa je  $f(x) = I_E(x)$ , gde je  $I_E(x) = 1$  kada  $x \in E$ , u suprotnom  $I_E(x) = 0$ . Ukoliko  $q(x) \propto I_E(x)p(x)$ , uzorkovanje će biti efikasnije, nego  $q(x) = p(x)$ , jer nema potrebe za predlozima uzoraka u regionima koji nisu od interesa.

Kako se dimenzija prostora  $X$  povećava, sve je teže pronaći adekvatnu predloženu raspodelu  $q(x)$ . Jedno od rešenja je parametrizovana predložena raspodela  $q(x, \theta)$ , gde je  $\theta$  parametar raspodele koji se prilagođava tokom uzorkovanja. Međutim, ponekad ni ovo nije dovoljno da pronađemo dovoljno jednostavnu raspodelu  $q$ , a opet da dobro aproksimira  $p$ . Zbog toga se uvode algoritmi uzorkovanja zasnovani na Markovljevima lancima.



Slika 2: (Preuzeto iz rada [1]) Ilustracija primera 2.1 uzorkovanja u značajnim regionima

## 2.2 Markovljevi lanci

Neka je  $\mathbb{X} = \{X_n, n \geq 0\}$  slučajan proces sa diskretnim vremenom  $n \in \mathbb{N}_0$ . Skup stanja ovog procesa označavamo sa  $\mathcal{S}$ .

**Definicija 2.1.** Diskretni slučajni proces  $X_n, n \geq 0$  je lanac Markova ako za svaki trenutak  $n \geq 0$  i za sva stanja  $i_0, i_1, \dots \in \mathcal{S}, i, j \in \mathcal{S}$ , važi

$$(8) \quad P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \dots, X_0 = i_0) = P(X_{n+1} = j | X_n = i).$$

Uvedimo oznaku za verovatnoću prelaska iz stanja  $i$  u stanje  $j$  u trenutku  $n$

$$(9) \quad p_{ij}(n) = P(X_{n+1} = j | X_n = i).$$

Za Markovljev lanac kažemo da je homogen ili da ima stacionarne verovatnoće prelaska ako je  $p_{ij}(n) = p_{ij}$ , za sve  $i, j \in \mathcal{S}, n \geq 0$ .

Verovatnoće prelaska u jednom koraku  $p_{ij}$  čine kvadratnu matricu  $\mathcal{P} = p_{ij}$ , dimenzija  $m \times m$ , gde je  $m$  broj elemenata prostora stanja  $\mathcal{S}$ . Ovu matricu nazivamo matricom verovatnoća prelaska (eng *kernel*). Ova matrica je stohastička<sup>2</sup>.

Homogen lanac Markova je potpuno određen verovatnoćama  $a_i = P(X_0 = i), i \geq 0$  i matricom verovatnoća prelaska  $\mathcal{P} = p_{ij}$ .

Pored verovatnoća prelaska u jednom koraku, odnosno prvog reda, možemo definisati i verovatnoće prelaska višeg reda

$$(10) \quad p_{ij}^{(n+1)} = \sum_{k \in \mathcal{S}} p_{ik}^{(n)} p_{kj} = \sum_{k \in \mathcal{S}} p_{ik} p_{kj}^{(n)}.$$

Navešćemo neka svojstva lanaca Markova, potrebna u narednim poglavljima.

<sup>2</sup>Stohastička matrica je kvadratna matrica  $n \times n$  elemenata  $p_{ij}$  takvih da je  $p_{ij} \in [0, 1]$  i važi  $\sum_{i=1}^n p_{ij} = 1$

**Definicija 2.2.** Lanac Markova je nesvodljiv (eng. *irreducible*), kada je svako stanje dostižno iz svakog drugog stanja, odnosno kada  $\forall i, j \in \mathcal{S} \exists n \geq 0 p_{ij}^{(n)} > 0$ .

**Definicija 2.3.** Stanje  $i$  je povratno ako se lanac Markova sa verovatnoćom 1 za konačno mnogo koraka vraća u stanje  $i$ . U suprotnom, stanje se naziva prolazno.

**Definicija 2.4.** Stanje  $i$  se naziva pozitivno povratno ako važi da je očekivano vreme do prvog povratka u stanje  $i$  konačno.

**Definicija 2.5.** Stanje  $j$  ima period  $m > 1$ , ako je  $p_{jj}^{(n)} = 0$ , osim u slučajevima kada je  $n$  oblika  $m \times k, k = 1, 2, \dots$ . Stanje  $j$  je neperiodično ako ne postoji broj  $m$  sa ovakvom osobinom.

**Definicija 2.6.** Neka je  $\pi = \{\pi_j, j \in \mathcal{S}\}$  raspodela verovatnoća. Ona se naziva stacionarnom raspodelom lanca Markova sa matricom verovatnoća prelaska  $\mathcal{P}$  ako važi  $\pi = \pi \mathcal{P}$  tj

$$(11) \quad \pi_j = \sum_{k \in \mathcal{S}} \pi_k p_{kj}, j \in \mathcal{S}.$$

Stacionarna raspodela ne mora uvek da postoji, međutim postoje dovoljni uslovi za postojanje ovakve raspodele. Naredne dve teoreme navodimo bez dokaza (v.[8]).

**Teorema 2.1.** *Ako je lanac Markova pozitivno povratan i nesvodljiv, onda postoji jedinstvena stacionarna raspodela  $\pi$ .*

**Teorema 2.2.** *Ukoliko je lanac Markova nesvodljiv, neperiodičan i postoji stacionarna raspodela  $\pi$ , tada je stacionarna raspodela  $\pi$  jedinstvena i važi*

$$(12) \quad \lim_{n \rightarrow \infty} p_{ij}^{(n)} = \pi_j, \forall i, j \in \mathcal{S}.$$

Lanac koji zadovoljava drugu teoremu naziva se i ergodičan lanac.

**Definicija 2.7.** Matrica verovatnoća prelaska  $\mathcal{P}$  zadovoljava uslov balansiranosti (eng *detailed balance*) za raspodelu  $\pi$  ako

$$(13) \quad \forall i, j \in \mathcal{S}, \pi(i)p_{ij} = \pi(j)p_{ji}$$

**Lema 2.3.** *Ako matrica verovatnoća prelaska  $\mathcal{P}$  zadovoljava uslov balansiranosti uslov za  $\pi$ , tada je  $\pi$  stacionarna raspodela za  $\mathcal{P}$ .*

Za dokaz pogledati [3].

## 2.3 Bajesovsko odlučivanje

U mnogim primenama, dostupni su nam podaci  $\mathcal{D}$  na osnovu kojih možemo da donosimo zaključke o određenim pojavama. Najčešće želimo da kreiramo neki model  $\mathcal{M}$  koji će opisivati zakonitosti u našim podacima, gde ćemo model posmatrati kao funkciju slučajnog parametra  $\Theta_M$  i podataka  $\mathcal{D}$ .

Parametar  $\Theta_M$  bi trebalo da bude takav tako da model  $M$  što bolje opisuje podatke  $\mathcal{D}$ , odnosno da je verovatnoća podataka  $P(\mathcal{D}|\Theta_M, \mathcal{M})$  u zavisnosti od slučajnog parametra što veća. Ovu verovatnoću još nazivamo i verodostojnost (eng. *likelihood*) parametara.

Osnovni cilj bajesovskog učenja jeste da pronademo vrednosti parametra  $\Theta_M$  u zavisnosti od definicije modela  $M$  i datih podataka -  $P(\Theta_M|\mathcal{D}, \mathcal{M})$ . Vezu između navedenih vrednosti možemo prikazati koristeći Bajesovu teoremu:

$$(14) \quad P(\Theta_M|\mathcal{D}, \mathcal{M}) = \frac{P(\mathcal{D}|\Theta_M, \mathcal{M})P(\Theta_M|\mathcal{M})}{P(\mathcal{D}|\mathcal{M})}$$

$P(\Theta_M|\mathcal{M})$  nazivamo apriornom raspodelom. Apriorna raspodela predstavlja neko uverenje koje imamo o parametru  $\Theta_M$  (npr. znamo da je verovatnoća da je  $\Theta_M$  pozitivan 0.95). Dodatno,  $P(\Theta_M|\mathcal{D}, \mathcal{M})$

nazivaćemo aposteriornom raspodelom, koja će predstavljati naše novo uverenje o parametru  $\Theta_M$  nakon saznanja  $D$ .

Vrednost  $P(D|\mathcal{M})$  nazivaćemo marginalna verodostojnost, jer  $P(D|\mathcal{M})$  možemo prikazati i kao marginalnu verodostojnost našeg modela  $M$ , marginalizovanu po svim mogućim vrednostima za parametar  $\Theta_M$ :

$$(15) \quad P(D|\mathcal{M}) = \int P(\mathcal{D}|\Theta_M, \mathcal{M})P(\Theta_M|\mathcal{M})d\Theta_M.$$

Jednom kada smo definisali aposteriornu raspodelu zanima nas kako možemo da je iskoristimo za donošenje zaključaka.

Osnovna upotreba aposteriorne raspodele jeste ocena parametara modela. Iako za ocene parametara modela postoje različite frekventističke statistike, u bajesovskom ocenjivanju, pored tačkaste ocene možemo definisati i intervale prekrivanja za naše parametre. Nadalje možemo koristiti aposteriornu raspodelu za kreiranje predikcija na neviđenim podacima, kao i za izbor modela u slučaju većeg broja dostupnih modela.

Uzimajući primene u obzir, češće će nas zanimati integrali nad aposteriornom raspodelom, nego sama aposteriorna raspodela. Vrlo često ti integrali ne mogu da se izračunaju analitički, gde primenu, pored različitih numeričkih estimacija, nalaze i MCMC tehnike uzorkovanja.

### 3 MCMC algoritmi uzorkovanja

Kao što je navedeno, koristićemo MCMC algoritme za ocenjivanje vrednosti koje nije moguće izračunati analitički, koristeći uzorke dobijene navedenim algoritmom.

Svi MCMC algoritmi zasnivaju se na kreiranju niza uzoraka  $\theta_1, \dots, \theta_n$  takvih da ovaj niz poseduje stacionarnu raspodelu  $\pi$ , koja je ujedno i njegova granična raspodela. Kako ćemo dole pokazati, izbor narednog uzorka zavisi samo od trenutnog uzorka, čime je ispunjeno Markovljevo svojstvo, čime naš lanac postaje Markovljev, a raspodela  $\pi$  postaje granična raspodela Markovljevog lanca.

Osnovni MCMC algoritmi su Gibsov (eng. *Gibbs*) algoritam i Metropolis-Hejstingsov algoritam (eng. *Metropolis Hastings*).

#### 3.1 Gibsov algoritam

Gibsovo uzorkovanje je jedan od najpopularnijih Monte Karlo simulacija za uzorkovanje visokodimenzionih raspodela, posebno u slučajevima gde su pojedinačne uslovne raspodele jednostavne. Ideja je da umesto željene d-dimenzionih raspodele  $\pi(\Theta) = \pi(\Theta_1, \Theta_2, \dots, \Theta_d)$  uzorkujemo d jednodimenzionih uslovnih raspodela. Neka je  $\Theta_{-i} = (\Theta_1, \dots, \Theta_{i-1}, \Theta_{i+1}, \dots, \Theta_d)$ .

**Sekvencijalno Gibsovo uzorkovanje:**

Uzima se početna vrednost  $(\Theta_1^{(0)}, \dots, \Theta_d^{(0)})$ , a zatim se iterativno uzorkuju vrednosti za svako  $t = 1, 2, 3, \dots$  iz jednodimenzionih uslovnih raspodela na sledeći način:

1. Uzorkujemo  $\Theta_1^{(t)} \sim \pi_{\Theta_1|\Theta_{-1}}(\cdot|\Theta_2^{(t-1)}, \dots, \Theta_d^{(t-1)})$   
...
- i. Uzorkujemo  $\Theta_j^{(t)} \sim \pi_{\Theta_j|\Theta_{-j}}(\cdot|\Theta_1^{(t-1)}, \dots, \Theta_{j-1}^{(t-1)}, \Theta_{j+1}^{(t-1)}, \dots, \Theta_d^{(t-1)})$   
...
- d. Uzorkujemo  $\Theta_d^{(t)} \sim \pi_{\Theta_d|\Theta_{-d}}(\cdot|\Theta_1^{(t-1)}, \dots, \Theta_{d-1}^{(t-1)})$

Uslovne raspodele koje koristimo u ovom algoritmu nazivaju se još i potpune uslovne raspodele (eng. *full conditionals*). Druga popularna varijanta Gibsovog uzorkovanja jeste nasumični Gibsov algoritam (eng. *Random Gibbs*), gde se uzorkuju slučajno odabrane potpune uslovne raspodele.



### Nasumično Gibsovo uzorkovanje:

Uzima se početna vrednost  $(\Theta_1^{(0)}, \dots, \Theta_d^{(0)})$ , a zatim se iterativno uzorkuju vrednosti za svako  $t = 1, 2, 3, \dots$  iz jednodimenzionalnih uslovnih raspodela na sledeći način:

1. Uzorkuje se indeks  $j$  iz ravnomerne raspodele na  $\{1, \dots, d\}$
2. Uzorkujemo  $\Theta_j^{(t)} \sim \pi_{\Theta_j|\Theta_{-j}}(\cdot|\Theta_1^{(t-1)}, \dots, \Theta_{j-1}^{(t-1)}, \Theta_{j+1}^{(t-1)}, \dots, \Theta_d^{(t-1)})$ , dok se ostali elementi kopiraju iz prethodne iteracije  $\Theta_k^t := \Theta_k^{t-1}$  za  $k \neq j$

#### 3.1.1 Hamersli Kliford teorema

Važno svojstvo, koje se koristi u Gibsovom algoritmu, jeste da potpune uslovne raspodele u potpunosti karakterišu zajedničku raspodelu, ukoliko su ispunjeni određeni slabi uslovi (Hammersley i Clifford, 1970).

**Definicija 3.1.** Raspodela sa gustinom  $\pi(\theta_1, \theta_2, \dots, \theta_n)$  i marginalnim raspodelama  $\pi_{\Theta_i}(\theta_i)$  zadovoljava uslov pozitivnosti  $\pi(\theta_1, \theta_2, \dots, \theta_n) > 0$  ako za svaki  $\theta_1, \dots, \theta_n$  važi  $\pi_{\Theta_i}(\theta_i) > 0$ .

Ova definicija implicira da je nosač zajedničke raspodele zapravo Dekartov proizvod pojedinačnih nosača marginalnih raspodela.

**Teorema 3.1.** *Hamersli-Kliford teorema* Neka raspodela  $\pi(\theta_1, \theta_2, \dots, \theta_n)$  zadovoljava uslov pozitivnosti. Tada za svaki  $z_1, \dots, z_n \in \text{supp}(\pi)$  važi

$$(16) \quad \pi(\theta_1, \theta_2, \dots, \theta_n) \propto \prod_{j=1}^n \frac{\pi_{\Theta_j|\Theta_{-j}}(\theta_j|\theta_1, \dots, \theta_{j-1}, z_{j+1}, \dots, z_n)}{\pi_{\Theta_j|\Theta_{-j}}(z_j|\theta_1, \dots, \theta_{j-1}, z_{j+1}, \dots, z_n)}$$

*Dokaz.* Važi

$$\pi(\theta_1, \theta_2, \dots, \theta_n) = \pi_{\Theta_n|\Theta_{-n}}(\theta_n|\theta_1, \dots, \theta_{n-1})\pi(\theta_1, \theta_2, \dots, \theta_{n-1}).$$

Slično

$$\pi(\theta_1, \theta_2, \dots, z_n) = \pi_{\Theta_n|\Theta_{-n}}(z_n|\theta_1, \dots, \theta_{n-1})\pi(\theta_1, \theta_2, \dots, \theta_{n-1}).$$

Sledi

$$\begin{aligned} \pi(\theta_1, \theta_2, \dots, \theta_n) &= \pi(\theta_1, \theta_2, \dots, z_n) \frac{\pi_{\Theta_n|\Theta_{-n}}(\theta_n|\theta_1, \dots, \theta_{n-1})}{\pi_{\Theta_n|\Theta_{-n}}(z_n|\theta_1, \dots, \theta_{n-1})} \\ &= \dots \\ &= \pi(z_1, z_2, \dots, z_n) \frac{\pi_{\Theta_1|\Theta_{-1}}(\theta_1|z_2, \dots, z_n)}{\pi_{\Theta_1|\Theta_{-1}}(z_1|z_2, \dots, z_n)} \frac{\pi_{\Theta_n|\Theta_{-n}}(\theta_n|\theta_1, \dots, \theta_{n-1})}{\pi_{\Theta_n|\Theta_{-n}}(z_1|\theta_1, \dots, \theta_{n-1})}. \end{aligned}$$

■

Hamersli-Kliford (eng. *Hammersley-Clifford*) teorema tvrdi da je dovoljno poznavanje svih pojedinačnih uslovnih raspodela koje će u potpunosti karakterisati nepoznatu raspodelu  $\pi$ .

#### Primer 3.1. Definisanost marginalnih raspodela

Kao što smo naveli uslovne raspodele dovoljne su da definišu zajedničku raspodelu, ali nije nužno da skup dobro definisanih uslovnih raspodela definiše dobro definisane marginalne raspodele. Npr. neka su uslovne raspodele dve eksponencijalne raspodele koje su svakako dobro definisane.

$$(17) \quad p(x|y) = ye^{(-yx)}, 0 < x < \infty$$

$$(18) \quad p(y|x) = xe^{(-xy)}, 0 < y < \infty$$

Međutim, Gibsovim uzorkovanjem ne bismo mogli da izvučemo uzorak marginalne niti zajedničke raspodele, zato što raspodele 17 i 18 ne formiraju nijednu zajedničku raspodelu  $(x, y)$ .

### 3.1.2 Konvergencija

Definišemo verovatnoće prelaska standardnog Gibsovog algoritma uzorkovanja:

$$(19) \quad K(\theta^{(t-1)}, \theta^{(t)}) = \pi_{\Theta_1|\Theta_{-1}}(\theta_1^{(t)}|\theta_{2:n}^{(t-1)}) \times \pi_{\Theta_2|\Theta_{-2}}(\theta_2^{(t)}|\theta_1^{(t)}, \theta_{3:n}^{(t-1)}) \times \dots \times \pi_{\Theta_n|\Theta_{-n}}(\theta_n^{(t)}|\theta_{1:n-1}^{(t)})$$

Kao i verovatnoće prelaska nasumičnog Gibsovog algoritma uzorkovanja:

$$(20) \quad K(\theta^{(t-1)}, \theta^{(t)}) = \frac{1}{n} \sum_{j=1}^n \pi_{\Theta_j|\Theta_{-j}}(\theta_j^{(t)}|\theta_{-j}^{(t-1)}) \delta_{\theta_{-j}^{(t-1)}}(\theta_{-j}^{(t)}),$$

gde je  $\delta_{\theta_{-j}^{(t-1)}}$  Dirakova delta funkcija u tački  $\theta_{-j}^{(t-1)}$ .

**Lema 3.2.**  $\pi(\theta_1, \theta_2, \dots, \theta_n)$  je invarijantna raspodela za sekvencijalni Gibsov algoritam uzorkovanja.

*Dokaz.* Želimo da pokažemo da je

$$\int_X \pi(\theta^{(t-1)}) K(\theta^{(t-1)}, \theta^t) d\theta^{(t-1)} = \pi(\theta^{(t-1)}).$$

Radićemo slučaj za  $n=2$ . Analogno se radi i za  $n>2$ .

$$\begin{aligned} & \int_X \pi(\theta^{(t-1)}) K(\theta^{(t-1)}, \theta^t) d\theta^{(t-1)} \\ &= \int_X \pi(\theta^{(t-1)}) \pi_{\Theta_1|\Theta_{-1}}(\theta_1^{(t)}|\theta_2^{(t-1)}) \times \pi_{\Theta_2|\Theta_{-2}}(\theta_2^{(t)}|\theta_1^{(t)}, \theta_2^{(t-1)}) d_{\theta_1}^{(t-1)} d_{\theta_2}^{(t-1)} \\ &= \int \pi_{\Theta_2}(x_2^{(t-1)}) \pi_{\Theta_1|\Theta_{-1}}(\theta_1^{(t)}|\theta_2^{(t-1)}) \times \pi_{\Theta_2|\Theta_{-2}}(\theta_2^{(t)}|\theta_1^{(t)}, \theta_2^{(t-1)}) d_{\theta_2}^{(t-1)} \\ &= \int \pi(x_1^{(t)}, x_2^{(t-1)}) \times \pi_{\Theta_2|\Theta_{-2}}(\theta_2^{(t)}|\theta_1^{(t)}, \theta_2^{(t-1)}) d_{\theta_2}^{(t-1)} \\ &= \pi_{\Theta_1}(x_1^{(t)}) \times \pi_{\Theta_2|\Theta_{-2}}(\theta_2^{(t)}|\theta_1^{(t)}, \theta_2^{(t-1)}) \\ &= \pi(x_1^{(t)}, x_2^{(t)}) \end{aligned}$$

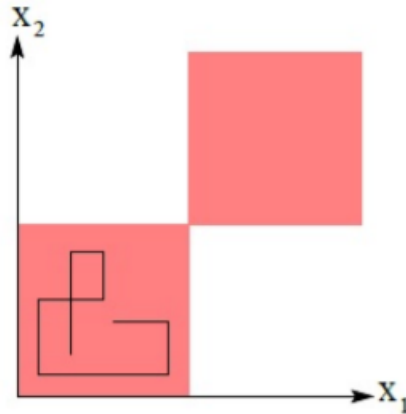
■

**Napomena:** Primetimo da matrica verovatnoća prelaska ne zadovoljava uslov balansiranosti (13).  
Npr. za  $n=2$

$$\pi(\theta^{(t-1)}) K(\theta^{(t-1)}, \theta^{(t)}) \neq \pi(\theta^{(t)}) K(\theta^{(t)}, \theta^{(t-1)})$$

Slično se može pokazati za nasumični Gibsov algoritam. Kao što znamo za konvergenciju lanca nije dovoljno da postoji stacionarna raspodela, već i da je lanac nesvodljiv.

**Primer 3.2.** Lanac koji nije nesvodljiv



Slika 3: Primer lanca koji nije nesvodljiv

Na slici 3 vidimo primer dvodimenzionalne raspodele  $p(X_1, X_2)$  za koju nije moguće ispravno izvršiti Gibsovo uzorkovanje. Raspodela ima vrednosti samo u obojenim kvadrantima. Neka je lanac u  $j$ -tom stanju  $(x_1^j, x_2^j)$ . Naredni korak je uzorkovanje iz  $p(x_2|x_1^j)$  vrednosti  $(x_1^{j+1}, x_2^{j+1})$  gde je  $x_1^{j+1} = x_1^j$ . Zatim uzorkujemo iz  $p(x_1|x_2 = x_2^{j+1})$ , itd. Ukoliko počnemo u donjem levom kvadrantu, gornji desni kvadrant nikad neće biti istražen.

**Lema 3.3.** *Ukoliko  $\pi(\theta_1, \dots, \theta_n)$  zadovoljava uslov pozitivnosti, tada uzorkovanje sekvencijalnim Gibsovim algoritmom generiše nesvodljiv i povratan Markovljev lanac.*

*Dokaz.* Za svaki skup  $A$  takav da  $\pi(A) := \int_A \pi(\theta_1, \dots, \theta_n) d\theta_1, \dots, d\theta_n > 0$  važi

$$\begin{aligned} P(\Theta^{(t)} \in A | \Theta^{(t-1)} = \theta^{(t-1)}) &= \int_A K(\theta^{(t-1)}, \theta^{(t)}) d\theta^{(t)} \\ &= \int_A \pi_{\Theta_1 | \Theta_{-1}}(\theta_1^{(t)} | \theta_2^{(t-1)}, \dots, \theta_n^{(t-1)}) \times \dots \times \pi_{\Theta_n | \Theta_{-n}}(\theta_n^{(t)} | \theta_1^{(t)}, \dots, \theta_{(n-1)}^{(t)}) d\theta^{(t)}, \end{aligned}$$

gde je  $\pi_{\Theta_1 | \Theta_{-1}}(\theta_1^{(t)} | \theta_2^{(t-1)}, \dots, \theta_n^{(t-1)}), \dots, \pi_{\Theta_n | \Theta_{-n}}(\theta_n^{(t)} | \theta_1^{(t)}, \dots, \theta_{(n-1)}^{(t)}) > 0$  na setu mere veće od nula. Tada  $P(\Theta^{(t)} \in A | \Theta^{(t-1)} = \theta^{(t-1)}) > 0$ , odnosno lanac je nesvodljiv. Kako smo već zaključili da lanac ima stacionarnu raspodelu i da je nesvodljiv, sledi i da je povratan. ■

Kako su zadovoljeni svi uslovi teoreme 2.2. možemo zaključiti da je lanac formiran Gibsovim algoritmom uzorkovanja ergodičan.

**Primer 3.3.** Pokazaćemo kroz primer motivaciju Gibsovog algoritma uzorkovanja. Razmatrajmo sledeću raspodelu:

$$p(x, y) = \frac{n!}{(n-x)!x!} y^{(x+\alpha-1)} (1-y)^{(n-x+\beta-1)}, x \in \{0, 1, \dots, n\}, y \in [0, 1].$$

Teško je uzorkovati direktno iz raspodele  $p(x, y)$ , dok nam je uzorkovanje iz marginalnih uslovnih raspodela lakše i poznato:

- $p(x|y) \sim \text{Bin}(n, y)$ , sa zakonom verovatnoće  $p(x = k) = \binom{n}{k} y^k (1-y)^{n-k}$ ,
- $p(y|x) \sim \text{Beta}(x + \alpha, n - x + \beta)$ , sa gustinom raspodele  $f(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}$ ,

stoga možemo primeniti Gibsov algoritam.

**Primer 3.4.** Hijerarhijsko klasterovanje

Neka su nam date 24 nezavisne vrednosti, koje su slučajno raspoređene u 4 različite kategorije. Vrednosti  $y_i, j$  za  $i = 1, \dots, n_j, j = 1, \dots, J$  su nezavisne normalno raspodeljene unutar svake od  $J$  grupa sa srednjom vrednošću  $\Theta_j$  i disperzijom  $\sigma^2$ , tj  $y_{ij} | \theta_j \sim N(\theta_j, \sigma^2)$ . Ukupan broj observacija je  $n = \sum_{j=1}^J n_j$ . Pretpostavljamo da srednja vrednost grupe ima normalnu raspodelu  $\Theta_j \sim N(\mu, \tau^2)$ . Pretpostavićemo da je apriorna raspodela za  $(\mu, \log \sigma, \log \tau)$  uniformna raspodela. Tada je aposteriorna raspodela jednaka

$$(21) \quad p(\theta, \mu, \log \sigma, \log \tau) \propto \tau \prod_{j=1}^J N(\theta_j, \tau^2) \prod_{j=1}^J \prod_{i=1}^{n_j} N(y_{ij} | \theta_j, \sigma^2).$$

1. Uslovna aposteriorna raspodela za  $\theta_j$ :

Izdvojimo delove formule koji sadrže  $\theta_j$  i pojednostavljujavanjem dobijamo

$$(22) \quad \theta_j | \theta_{-j}, \mu, \sigma, \tau, y \sim N(\hat{\theta}_j, V_{\theta_j}),$$

gde je  $\hat{\theta}_j := \frac{\frac{1}{\tau^2} \mu + \frac{n_j}{\sigma^2} \bar{y}_j}{\frac{1}{\tau^2} + \frac{n_j}{\sigma^2}}$  i  $V_{\theta_j} := \frac{1}{\frac{1}{\tau^2} + \frac{n_j}{\sigma^2}}$ .

2. Uslovna aposteriorna raspodela za  $\mu$  :  
Izdvojimo delove formule koji sadrže  $\mu$  i pojednostavljujavanjem dobijamo

$$(23) \quad \mu | (\theta, \sigma, \tau, y) \sim N(\hat{\mu}, \frac{\tau^2}{J}),$$

gde je  $\hat{\mu} := \frac{1}{J} \sum_{j=1}^J \theta_j$ .

3. Uslovna aposteriorna raspodela za  $\sigma^2$  :  
Izdvojimo delove formule koji sadrže  $\sigma$  i pojednostavljujavanjem dobijamo

$$(24) \quad \sigma | (\theta, \mu, \tau, y) \sim Inv - \chi^2(n, \hat{\sigma}^2),$$

gde je  $\hat{\sigma}^2 := \frac{1}{n} \sum_{j=1}^J \sum_{n=1}^{n_j} (y_{ij} - \theta_j)^2$  i  $Inv - \chi^2$  inverzna  $\chi^2$  raspodela.

4. Uslovna aposteriorna raspodela za  $\tau^2$  :  
Izdvojimo delove formule koji sadrže  $\tau$  i pojednostavljujavanjem dobijamo

$$(25) \quad \tau^2 | (\theta, \mu, \sigma, y) \sim Inv - \chi^2(J - 1, \hat{\tau}^2),$$

gde je  $\hat{\tau}^2 := \frac{1}{J-1} \sum_{j=1}^J (\theta_j - \mu)^2$ .

Primer pokazuje kako je kompleksno uzorkovati iz zajedničke raspodele, dok su pojedinačne jednostavnije za ocenjivanje.

### 3.2 Metropolis-Hejstings algoritam

Pretpostavimo da želimo da uzorkujemo iz raspodele nad prostorom  $\Theta$  sa gustinom raspodele  $\pi$ . Kako nam raspodela iz koje želimo uzorkovati nije poznata, potrebno je kreirati predloženu raspodelu  $Q$  sa gustinom  $q$ , na sličan način kao u uzorkovanju odbacivanjem. Nadalje potrebno je definisati verovatnoću prihvatanja svakog od predloženih uzoraka, imajući u vidu gore navedena svojstva koja želimo da postignemo. Odnosno, potrebno je obezbediti postojanje stacionarne raspodele našeg lanca, koja je ujedno i granična raspodela. Kao što smo pokazali u poglavlju koje se odnosilo na Markovljeve lance, dovoljno je da naš lanac zadovoljava uslov balansiranosti.

Definišemo verovatnoću prihvatanja uzorka, takvu da, kako ćemo kasnije pokazati, zadovoljava uslov balansiranosti, a samim tim i obezbeđuje postojanje stacionarne raspodele našeg lanca koja je baš ona raspodela koju želimo da ocenimo gustinom naših uzoraka  $\alpha(\theta'_t | \theta_{(t-1)}) = \min \left( 1, \frac{\pi(\theta'_t) q(\theta_{(t-1)} | \theta_t)}{\pi(\theta_{(t-1)}) q(\theta'_t | \theta_{(t-1)})} \right)$ .

Metropolis-Hejstings (eng. *Metropolis-Hastings*) algoritam sastoji se od sledećih koraka.

Neka je početna tačka  $\theta_0$  i  $t > 0$ . Za svaki korak  $t$  važi sledeći postupak.

1. Uzimamo novi uzorak  $\theta'_t$  iz predložene gustine  $q(\theta'_t | \theta_{(t-1)})$ .
2. Računamo verovatnoću prihvatanja uzorka

$$(26) \quad \alpha(\theta'_t | \theta_{(t-1)}) = \min \left( 1, \frac{\pi(\theta'_t) q(\theta_{(t-1)} | \theta_t)}{\pi(\theta_{(t-1)}) q(\theta'_t | \theta_{(t-1)})} \right)$$

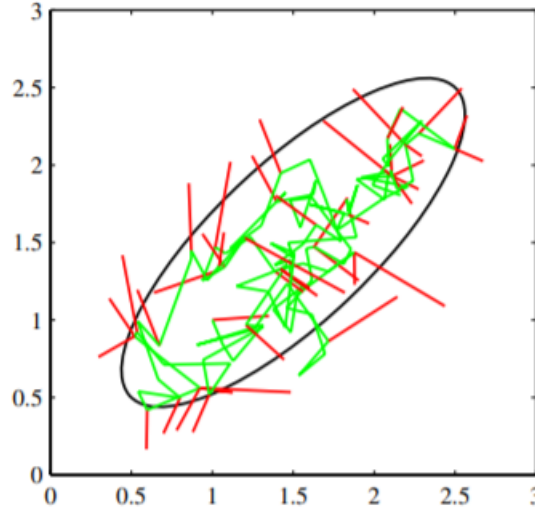
3. Generišemo slučajan broj  $u_t$  iz  $U[0, 1]$ .

4. Ako je  $u_t \leq \alpha(\theta'_t | \theta_{(t-1)})$  prihvataemo tekući uzorak  $\theta_t = \theta'_t$ . U suprotnom  $\theta_t = \theta_{(t-1)}$ .

Verovatnoće prelaska za Metropolis-Hejstings računaju se na sledeći način:

$$(27) \quad K(\theta_{(t-1)}, \theta_t) = \alpha(\theta_t | \theta_{(t-1)}) q(\theta_t | \theta_{(t-1)}) + (1 - \alpha(\theta_{(t-1)})) \delta_{\theta_{(t-1)}}(\theta_t),$$

gde je  $\delta_{\theta_{(t-1)}}(\theta_t)$  Dirakova delta funkcija u  $\theta_{t-1}$ .



Slika 4: Ilustracija uzorkovanja MH algoritmom normalne raspodele čije konture su prikazane elipsom. Predložena raspodela je dvodimenzionalna normalna raspodela sa dijagonalnom kovarijacionom matricom i  $sd=0.2$ . Koraci koji su prihvaćeni obeleženi su zelenom, dok su odbijeni obeleženi crvenom bojom. Od 150 uzoraka, 43 su odbijena.

### 3.2.1 Konvergencija

Možemo dokazati da ovako konstruisan Markovljev lanac zadovoljava uslove dovoljne za konvergenciju.

**Teorema 3.4.** *Matrica verovatnoća prelaska MH algoritma zadovoljava uslov balansiranosti (13).*

$$(28) \quad \pi(\theta_{(t-1)})K(\theta_{(t-1)}, \theta_{(t)}) = \pi(\theta_{(t)})K(\theta_{(t)}, \theta_{(t-1)})$$

*Dokaz.* Važi:

$$\begin{aligned} \pi(\theta_{(t-1)})q(\theta_{(t)}|\theta_{(t-1)})\alpha(\theta_{(t)}|\theta_{(t-1)}) &= \pi(\theta_{(t-1)})q(\theta_{(t)}|\theta_{(t-1)}) \min \left( 1, \frac{\pi(\theta_{(t)})q(\theta_{(t-1)}|\theta_{(t)})}{\pi(\theta_{(t-1)})q(\theta_{(t)}|\theta_{(t-1)})} \right) \\ &= \min \left( \pi(\theta_{(t-1)})q(\theta_{(t)}|\theta_{(t-1)}), \pi(\theta_{(t)})q(\theta_{(t-1)}|\theta_{(t)}) \right) \\ &= \pi(\theta_{(t)})q(\theta_{(t-1)}|\theta_{(t)}) \min \left( 1, \frac{\pi(\theta_{(t-1)})q(\theta_{(t)}|\theta_{(t-1)})}{\pi(\theta_{(t)})q(\theta_{(t-1)}|\theta_{(t)})} \right) \\ &= \pi(\theta_{(t)})q(\theta_{(t-1)}|\theta_{(t)})\alpha(\theta_{(t-1)}|\theta_{(t)}) \end{aligned}$$

Kako je  $\delta_{\theta_{(t-1)}}(\theta_{(t)}) = 0$ , za  $\theta_{(t-1)} \neq \theta_{(t)}$ , važi dalje:

$$\begin{aligned} \pi(\theta_{(t-1)})K(\theta_{(t-1)}, \theta_{(t)}) &= \pi(\theta_{(t-1)})q(\theta_{(t)}|\theta_{(t-1)})\alpha(\theta_{(t)}|\theta_{(t-1)}) + \pi(\theta_{(t-1)})(1 - \alpha(\theta_{(t-1)}))\delta_{\theta_{(t-1)}}(\theta_{(t)}) \\ &= \pi(\theta_{(t)})q(\theta_{(t-1)}|\theta_{(t)})\alpha(\theta_{(t-1)}|\theta_{(t)}) + \pi(\theta_{(t)})(1 - \alpha(\theta_{(t)}))\delta_{\theta_{(t)}}(\theta_{(t-1)}) \\ &= \pi(\theta_{(t)})K(\theta_{(t)}, \theta_{(t-1)}) \end{aligned}$$

Kako je zadovoljen uslov balansiranosti jednačina raspodela  $\pi$  je stacionarna. ■

Potrebno je pokazati i da je  $\pi$  granična raspodela. Kao što smo naveli, dovoljno je pokazati da je lanac nesvodljiv i aperiodičan.

Pod slabim pretpostavkama za predloženu raspodelu, zagarantovano je da možemo dostići bilo koji skup tačaka sa nenegativnom verovatnoćom  $\pi$ .

Kada je  $q(\theta|\theta') > 0, \forall \theta, \theta' \in \text{supp}\pi$ ,<sup>3</sup> tada je lanac strogo nesvodljiv ( $\theta \rightarrow \theta'$  u jednom koraku). Primer za ovo je kada je  $\Theta = \mathbb{R}$ , a  $q(\theta|\theta')$  normalna raspodela sa srednjom vrednošću  $\theta'$  i nedegenerisanom kovarijacionom matricom.

Ako je ciljna gustina ograničena između 0 i  $\infty$  na kompaktnim skupovima, i ako postoji  $\delta, \epsilon > 0$  tako da za svako  $\theta \in \Theta$  važi

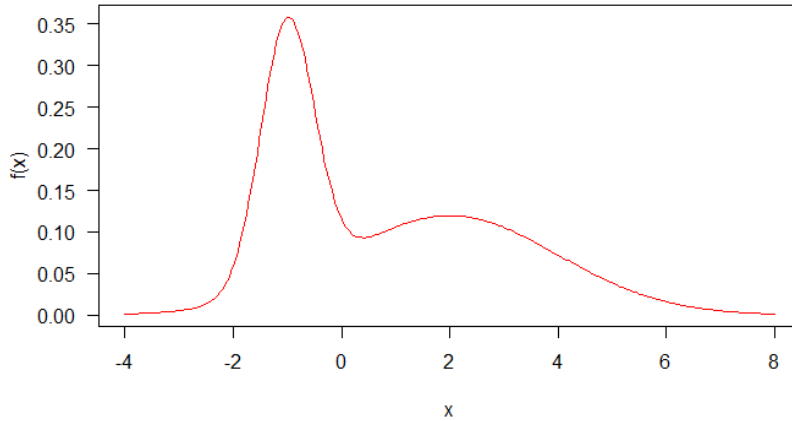
$$(29) \quad |\theta - \theta'| < \delta \Rightarrow q(\theta|\theta') \leq \epsilon,$$

tada se može pokazati da je lanac nesvodljiv u odnosu na raspodelu  $\pi$  (Robert & Rosenthal, 2004).

Dodatno, Markovljev lanac je aperiodičan, ako mu je verovatnoća ostajanja u istom stanju pozitivna, što je uglavnom zadovoljeno za lance MH algoritma.<sup>4</sup>

### 3.2.2 Jednostavan primer MH algoritma

Želimo da prikazemo navedene korake Metropolis Hastings algoritma na jednostavnom primeru, čija je svrha da pokaže funkcionalnost ove metode. Cilj nam je da uzorkujemo nepoznatu raspodelu koja je mešavina dve normalne raspodele. Gustina takve raspodele biće oblika  $f = p \cdot \Phi_1 + (1 - p) \cdot \Phi_2$ , gde je  $\Phi_1$  gustina normalne  $\mathcal{N}(\mu_1, \sigma_1^2)$  i  $\Phi_2$  gustina normalne  $\mathcal{N}(\mu_2, \sigma_2^2)$  raspodele. Parametre biramo proizvoljno:  $p = 0.4$ ,  $\mu_1 = -1$ ,  $\mu_2 = 2$ ,  $\sigma_1 = 0.5$  i  $\sigma_2 = 2$ .



Predložena raspodela biće normalna sa srednjom vrednosti u trenutnoj tački (srednju vrednost poboljšavamo iteracijama) i standardnom devijacijom 4,  $q = \mathcal{N}(x, 4)$ . Pravimo funkciju koja predstavlja jednostavnu implementaciju gore navedenog algoritma:

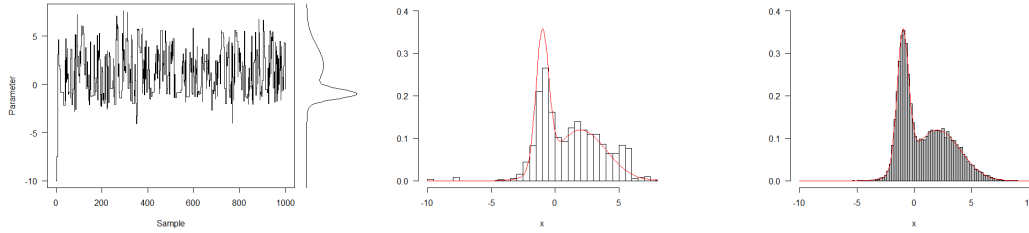
```
funkcija_koraka <- function(x, f, q) {
  ## Predlažemo uzorak iz raspodele q
  xp=q(x)
  ## Računamo verovatnoću prihvatanja predloženog uzorka:
  alpha=min(1, f(xp) / f(x))
  ## Prihvatamo uzorak sa verovatnoćom alpha:
  if (runif(1) < alpha)
    x=xp
  ## Returning the point:
  return x
}
```

Navedena funkcija se iterativno izvršava unapred definisan broj puta čuvajući vrednosti lanca. Inicijalni parametar se bira proizvoljno, u ovom slučaju -10.

<sup>3</sup>  $\text{supp}\pi$  označava nosač raspodele

<sup>4</sup> Ovaj uslov nije zadovoljen kada je  $q(\theta_{(t)}|\theta_{(t-1)}) = \pi(\theta_{(t)})$ , ali tada se lanac sastoji od iid uzoraka, stoga je svakako aperiodičan.

Možemo da prikazemo lanac Markova koji smo generisali. Vidimo da već sa 1000 koraka dobijamo histogram koji liči na našu gustinu. Ukoliko povećamo broj iteracija na 50000, dobijamo jasno dobar rezultat.



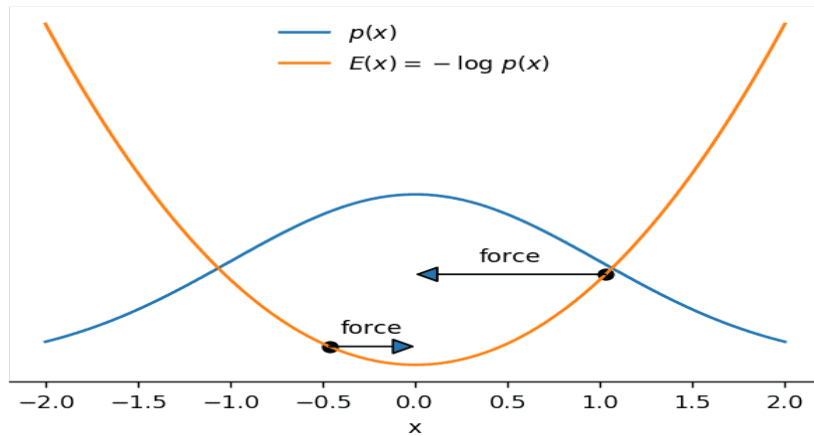
U narednom poglavlju pored ostalih ograničenja, ispitaćemo i kako izbor predložene raspodele može da utiče na kretanje Markovljevog lanca.

### 3.3 Hamiltonovski Monte Karlo algoritam

Hamiltonovski Monte Karlo algoritam (eng. *Hamiltonian Monte Carlo*, skraćeno HMC) je jedan od naprednijih MCMC algoritama, koji koristi gradijent gustine raspodele radi smanjenja međusobne korelacije članova uzoraka. Ukoliko je gustina raspodele koju želimo da uzorkujemo diferencijabilna, mi možemo da odredimo lokalna svojstva funkcije u svakoj tački, koristeći njene izvode - što je osnovna ideja HMC algoritma. Motivaciju za ovaj algoritam nalazimo u fizici. Pretpostavimo da stanje lanca možemo da tumačimo kao česticu nad kojom deluju određene sile koje je guraju u prostor veće verovatnoće. Navedenu silu smatraćemo potencijalnom energijom čestice, i definišemo je kao negativnu gustinu raspodele čestice:

$$(30) \quad E(x) = -\log p(x).$$

Koreni za ovu definiciju potiču iz statističke mehanike, u koju nećemo ulaziti u ovom radu [20]. Na slici 5 vidimo da što je strmije  $E(x)$  to je jača sila koja vuče česticu. Štaviše najniža tačka  $E(x)$  poklapa se sa regionom najviše verovatnoće  $p(x)$ . Ovo znači da možemo da predvidimo kretanje čestice na osnovu njene tekuće pozicije i brzine, i da navedenu predikciju koristimo kao predloženo stanje u standardnim MCMC algoritmima.



Slika 5: Gustina raspodele  $p(x)$  i njen negativni logaritam koji nazivamo potencijalnom energijom

Teorija iza ovog algoritma se oslanja na fiziku, preciznije Hamiltonovu mehaniku. Nećemo ulaziti detaljnije u ove teorijske osnove, dalje od potrebnog, ali za više detalja videti [20]. Kao što smo naveli, negativni logaritam gustine raspodele predstavlja potencijalnu energiju čestice. Pored

potencijalne, definišemo i kinetičku energiju:

$$(31) \quad K(v) = \frac{|v|^2}{2m},$$

gde  $v$  predstavlja impuls, a  $m$  masu čestice. Radi jednostavnosti, unutar HMC algoritma masa se često postavlja na jediničnu masu. Dakle, jednačina ukupne energije čestice, kojom možemo u potpunosti da odredimo kretanje čestice, definiše se kao:

$$(32) \quad H(x, v) = K(v) + E(x).$$

Ovo se još naziva i Hamiltonijan.

HMC tretira impuls kao pomoćnu varijablu, te uzorkuje zajedničku raspodelu pozicije i impulsa čestice  $p(x, v)$ . Pod određenim pretpostavkama, zajedničku raspodelu  $p(x, v)$  definišemo na sledeći način:

$$\begin{aligned} p(x, v) &\propto e^{-H(x, v)} \\ &= e^{-K(v)} \times e^{-E(x)} \\ &= e^{-\frac{|v|^2}{2}} \times p(x). \end{aligned}$$

HMC algoritam sastoji se iz dva koraka. Najpre, iz normalne raspodele uzorkujemo  $v$ , pomoću kojeg vršimo simulaciju kretanja čestice  $x$  u novo stanje  $x^*$  koristeći Hamiltonovu dinamiku. Na kraju simulacije, nalazimo se u stanju  $x^*$  sa impulsom  $v^*$ . Kao i u originalnom MH algoritmu, definišemo verovatnoću prihvatanja:

$$(33) \quad p_{ACC} = \min\{1, e^{-|H(x^*, v^*) - H(x, v)|}\}$$

Ono što nismo imali u prethodnim verzijama MCMC algoritama jeste simulacija kretanja čestice između dva uzastopna uzorka. Potrebno je diskretizovati vreme u korake određene veličine. Kretanje čestice je potpuno određeno Hamiltonovim diferencijalnim jednačinama. Numeričke metode koje rešavaju ove jednačine moraju da zadovoljavaju određena svojstva:

- reverzibilnost - mogućnost da lanac pustimo unazad - bitno za uslov balansiranosti,
- očuvanje zapremine - da bi verovatnoća prihvatanja bila jednostavna,
- simplektičnost - greška energije je mala, što dovodi do visokih verovatnoća prihvatanja.

Metod koji zadovoljava sva navedena svojstva i najčešće se koristi u HMC je skokovita integracija (eng. *leapfrog*). Parametri ovog algoritma su broj koraka  $L$  i veličina koraka  $\epsilon$ .

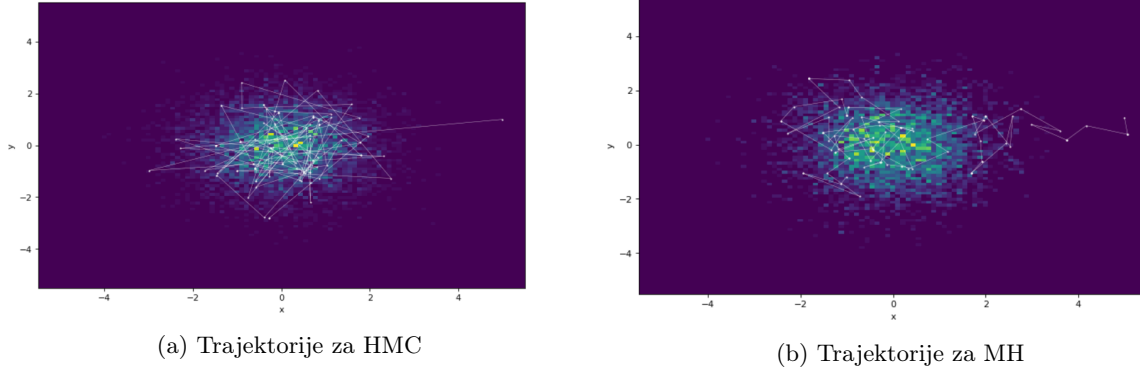
HMC zadovoljava uslov balansiranosti čime pokazujemo da je lanac kreiran ima stacionarnu raspodelu. Ergodičnost se takođe može pokazati. Postoji mogućnost da lanac postane periodičan lošim izborom  $L$  i  $\epsilon$  i time naruši teoremu o ergodičnosti. Preporuka je da se parametri biraju slučajno iz nekog malog intervala.

Osnovna prednost HMC algoritma u odnosu na standardni MH algoritam jeste izbegavanje slučajnog lutanja što ćemo videti u narednom primeru. Kreiranje manje koreliranih uzoraka, dovodi do brže konvergencije ka stvarnoj raspodeli. Mane su što HMC može da se koristi samo u slučaju neprekidnih promenljivih, s obzirom na to da gradijent gustine nije definisan za diskretne raspodele. Dodatno, gradijent nekada može biti zahtevan za izračunavanje.

**Primer 3.5.** Poređenje Hamiltonovskog Monte Karlo algoritma i Metropolis-Hejstings algoritama

Želimo da prikazemo razlike i prednosti Hamiltonovskog Monte Karlo algoritma u odnosu na osnovni Metropolis-Hejstings algoritam. Kod za implementaciju HMC algoritma nalazi se u poglavlju 7. Kao primer uzećemo uzorkovanje dvodimenzionalne normalne raspodele. Cilj je da ocenimo gustinu navedene dvodimenzionalne normalne raspodele. Pustićemo 2 lanca, jedan koristeći HMC algoritam (slika 6a), drugi koristeći MH algoritam (slika 6b). Kao što vidimo na slici 6 Hamiltonovski Monte Karlo (skr. HMC) pravi velike skokove između stanja zahvaljujući simulaciji kretanja između uzorkovanja. S druge strane, Metropolis-Hejstings (skr. MH) zahteva više vremena pre nego što uđe u prostor visoke verovatnoće. Zaključujemo da HMC sa sličnim procentom prihvaćenih uzoraka ima manje visoko koreliranih uzoraka, što nas dovodi do reprezentativnog uzorka ranije.



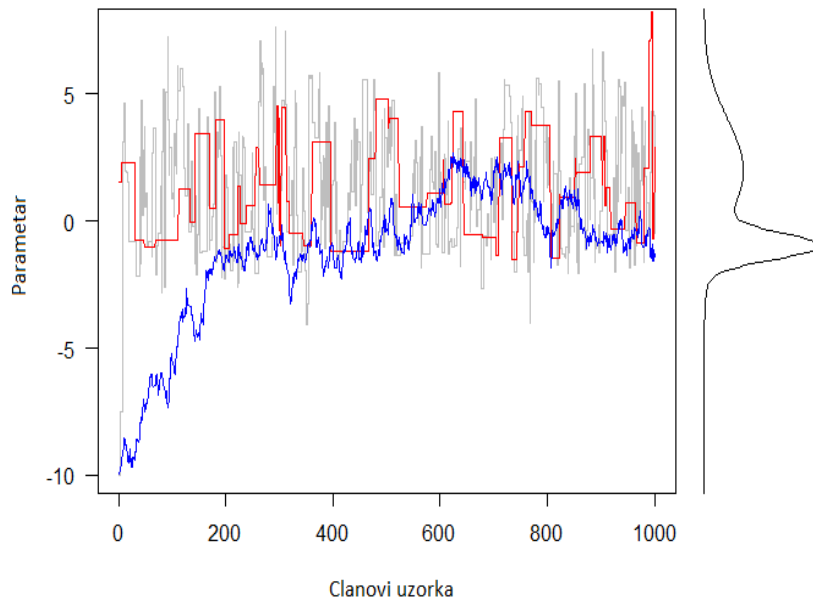


Slika 6: Uzorkovanje dvodimenzionalne normalne raspodele

## 4 Nedostaci MCMC algoritama

### 4.1 Izbor predložene raspodele i inicijalnih parametara

Kako je navedeno na kraju prethodnog poglavlja, želimo da prikazemo i kako izbor predložene raspodele može da utiče na kretanje i brzinu konvergencije Markovljevog lanca. Izabraćemo jednu predloženu raspodelu sa velikom standardnom devijacijom ( $sd = 33$ ) i jednu sa malom standardnom devijacijom ( $sd = 0.3$ ).



Slika 7: Brzina konvergencije lanca u zavisnosti od izbora  $\sigma$

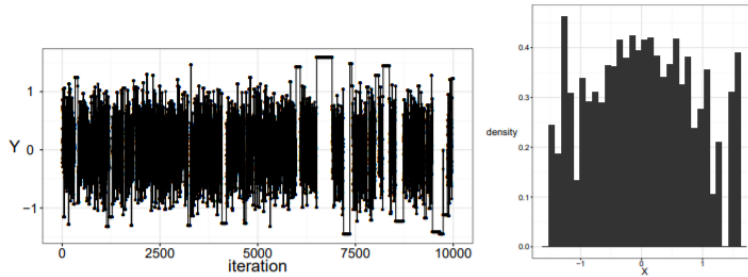
Lanac sa originalnom predloženom raspodelom sa standardnom devijacijom ( $sd = 4$ ) prikazan je sivom bojom. Crvenoj boji odgovara trajektorija lanca sa predloženom raspodelom sa velikom disperzijom. Tu primećujemo često odbacivanje vrednosti, odnosno stagniranje lanca u istoj vrednosti duže vreme. Ovo dovodi do velike korelacije uzoraka i spore konvergencije lanca ka željenoj raspodeli. Raspodela koja odgovara trajektoriji obojenoj plavom bojom predlaže male korake, koji se prihvataju, ali se zato sporije približava modama raspodele. Ova putanja je nalik nekoj stohastičkoj trajektoriji koja sporo istražuje ceo prostor vrednosti.

Vrlo često inicijalni parametri se biraju proizvoljno. Lošim izborom predložene raspodele možemo znatno

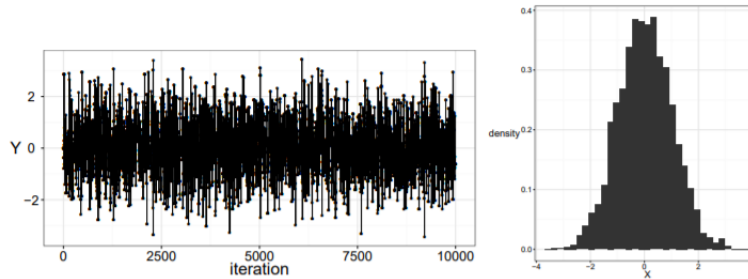
usporiti konvergenciju ka stacionarnoj raspodeli, iako ona postoji. Dobar izbor predložene raspodele i pravilna inicijalizacija parametara doprineće da stignemo brže u visoko verovatni predeo željene raspodele.

**Primer 4.1.** Neka je  $\pi(\theta) = \mathcal{N}(\theta; 0, 1)$  i primenjujemo MH algoritam sa dve različite predložene raspodele.

- $q_1(\theta) = \mathcal{N}(\theta; 0, 0.4^2)$ , takva da  $\frac{\pi(\theta)}{q_1(\theta)} \rightarrow \infty$  kada  $|\theta| \rightarrow \infty$
- $q_2(\theta) = \mathcal{N}(\theta; 0, 5^2)$ , takva da  $\frac{\pi(\theta)}{q_2(\theta)} \leq M < \infty$



(a) Trajektorije i histogram za IMH sa  $q_1$



(b) Trajektorije i histogram za IMH sa  $q_2$

Slika 8: Ishodi MH algoritma u zavisnosti od izbora  $q$

Možemo primetiti na slici 8 da se lanac sa predloženom raspodelom  $q_1$  zaustavlja u repovima raspodele, kako je vrednost  $\frac{p_i(\theta^{(t-1)})}{q_1(\theta^{(t-1)})}$  u prethodnom stanju visoka, verovatnoća prihvatanja nove tačke je niska.

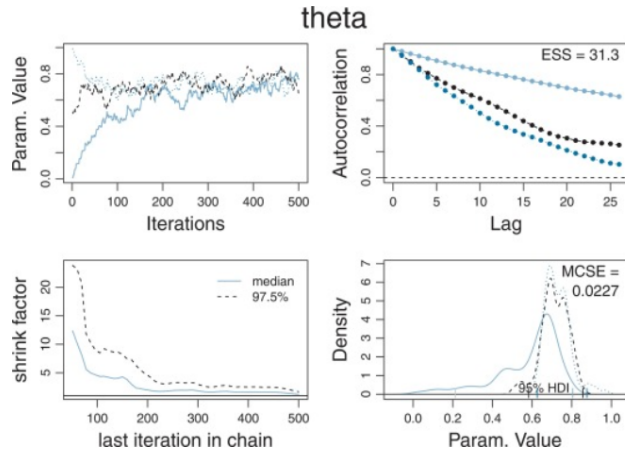
## 4.2 Odbacivanje početnih uzoraka

Kako se početna tačka, kojom se inicijalizuje lanac, može naći u predelima niske gustine uzorkovane raspodele, određen broj prvih članova lanca može dovesti do loših ocena raspodele. Stoga je praksa da se prvih par članova lanca odbaci, (eng. *burn in*), čime obezbeđujemo ulazak u region visoke gustine uzorkovane raspodele.

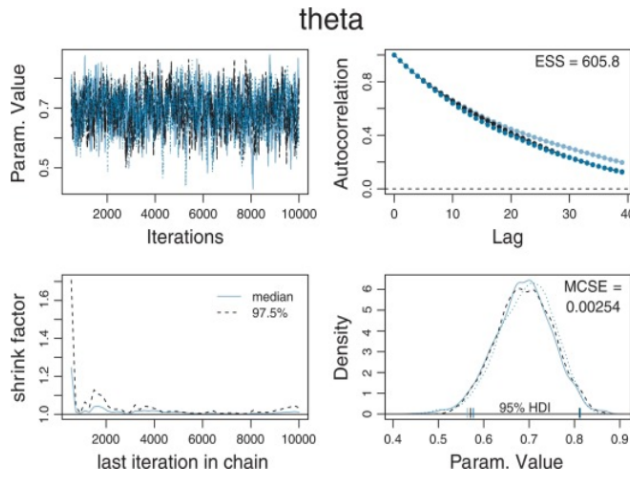
Postavlja se pitanje kako izabrati broj uzoraka koji odbacujemo, odnosno kako proceniti da li se naš lanac nalazi u prostoru visoke gustine. Ne postoji univerzalan način. U praksi se najčešće koristi vizuelna procena trajektorije lanca - kreira se grafik vrednosti uzoraka u zavisnosti od koraka lanca  $t$ . Jedan od predloga za naglašavanje loših uzoraka jeste da se prikažu dva ili više lanaca na istom prostoru. Kako oni svi predstavljaju istu raspodelu, trebalo bi da se preklapaju. Na slici 9a, u gornjem levom uglu, možemo videti trajektorije 3 lanca za prvih 500 iteracija, dok slika 9b prikazuje lanac u narednim iteracijama. Na  $y$  osi nalaze se vrednosti parametara, a na  $x$  osi koraci, odnosno iteracije lanca. Možemo da uočimo da je potrebno par stotina iteracija da bi lanci iskonvergirali u isti predeo vrednosti parametra. Prema tome broj uzoraka za odbacivanje na navedenom primeru, jeste par stotina.

Drugi način je da prikazemo gustinu vrednosti parametara za navedene lance i da vidimo da li se one

poklapaju, kao što možemo videti u donjem desnom uglu na slikama 9a i 9b.



(a) Trajektorije tokom početnog perioda



(b) Trajektorije tokom perioda stabilnosti

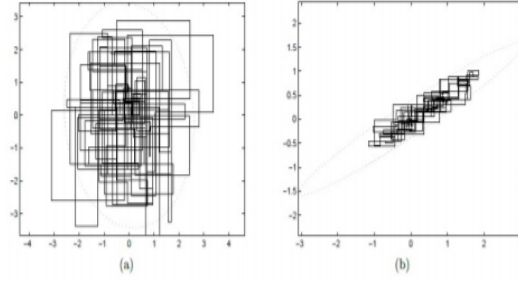
Slika 9: Vizuelna analiza konvergencije

Takođe postoje numeričke metode za ocenu broja odbačenih uzoraka. Najpopularnija je Brooks-Gelman-Rubin statistika (Brooks i Gelman, 1998), koja se još naziva i faktor umanjenja. Ona meri odnos disperzije između različitih lanaca i disperzija unutar jednog lanca. Kada su se lanci našli u prostoru reprezentativnog uzorka, tada će prosečna razlika između lanaca biti ista kao prosečna razlika unutar pojedinačnog lanca u različitim iteracijama. Ukoliko neki lanac sadrži nereprezentativne uzorke, disperzija između lanaca, biće veća nego disperzija unutar lanaca. Tokom početnog perioda ova statistika uzima vrednost veću od 1, dok u periodima stabilnosti bude blizu 1. U praksi, za sve vrednosti veće od 1.1 možemo sumnjati u reprezentativnost naših uzoraka.

### 4.3 Korelacija uzoraka

Naredni izazov oba navedena algoritma jeste korelacija uzastopnih uzoraka. Zašto ovo uopšte predstavlja izazov? Kada su uzorci međusobno visoko korelisani, zavisnost od početnih stanja je veća, pa je samim tim potreban mnogo veći broj iteracija, da bismo dostigli konvergenciju ka željenoj raspodeli.

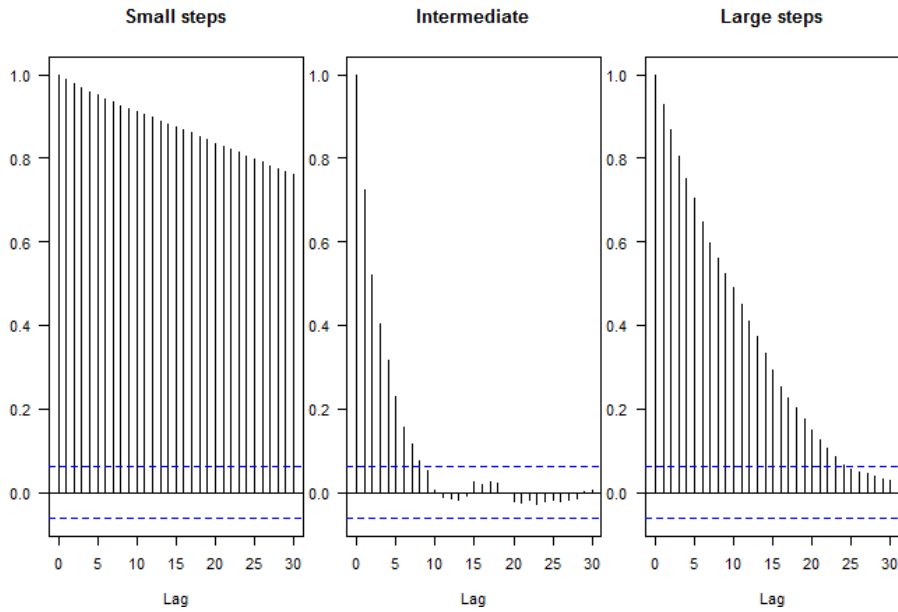
Na slici 10 [24] vidimo 200 uzoraka izvučenih Gibsovim algoritmom za dvodimenzionalnu normalnu raspodelu. U svakom koraku se ažurira jedna komponenta. Sa leve strane imamo normalnu raspodelu sa



Slika 10: Istraživanje prostora u zavisnosti od korelisanosti uzoraka

niskom korelacijom, pa Gibsov algoritam može da prođe kroz sve regione efikasno. Sa desne strane, zbog visoke korelacije Gibsov algoritam je manje efikasan i ne istražuje sve verovatne regione u malom broju iteracija. Kako dolazi do velike korelacije susednih uzoraka?

Svaki novi uzorak  $\theta^t$  se izvlači iz raspodele  $Q$  koji po definiciji zavisi (samo) od prethodnog koraka  $\theta^{t-1}$ . Nadalje uzorak  $\theta^{t+1}$  će biti korelisan sa  $\theta^t$ , koji je korelisan sa  $\theta^{t-1}$  i tako nadalje.



Slika 11: Prikaz autokorelacije uzastopnih uzoraka lanca

Kao jedno od načina za prevazilaženje ovog izazova predlaže se IMH (Independence Metropolis-Hastings) algoritam. U ovom slučaju predložena raspodela ima oblik  $q(\theta^t|\theta^{t-1}) = q(\theta^t)$  - svi uzorci su nezavisni i jednako raspodeljeni. Tada je verovatnoća prihvatanja koraka jednaka

$$\frac{\pi(\theta^t)q(\theta^{(t-1)}|\theta^t)}{\pi(\theta^{(t-1)})q(\theta^t|\theta^{(t-1)})} = \frac{\pi(\theta^t)q(\theta^{(t-1)})}{\pi(\theta^{(t-1)})q(\theta^t)}$$

Primetimo da, iako su izvlačenja iz predložene gustine nezavisna, formirani lanac nije lanac sa nezavisnim vrednostima, jer verovatnoća prihvatanja uzorka  $T$  zavisi od prethodne vrednosti, dok u slučaju odbijanja uzorka dva susedna uzorka su identična, dakle imaju korelaciju 1.

Primer 4.1 je primer lanca IMH algoritma.

Još jedna tehnika, koja se često primenjuje u praksi, jeste odbacivanje svih uzoraka sem svakog  $k$ -tog (eng. *thinning*). Potrebno je voditi računa da se  $k$  ne uzima previše veliko, da se ne bi preveliki broj uzoraka odbacivao. Ovaj broj se određuje empirijski.

## 4.4 Konvergencija ka lokalnim optimumima

Postoji mogućnost da se lanac „zaglavi“ u nekom lokalnom optimumu, stoga postoje predložene raspodele koje mogu da poboljšaju konvergenciju lanca.

Jedna od predloženih raspodela prati trajektoriju slučajnog lutanja, tj.  $\Theta^{(t)} = \Theta^{(t-1)} + W$ , gde  $W \sim g$ , tako da je  $g$  gustina simetrične raspodele  $g(-w) = g(w)$ . Na primer,  $g$  bi mogla da bude gustina normalne ili Studentove raspodele.

U ovom slučaju

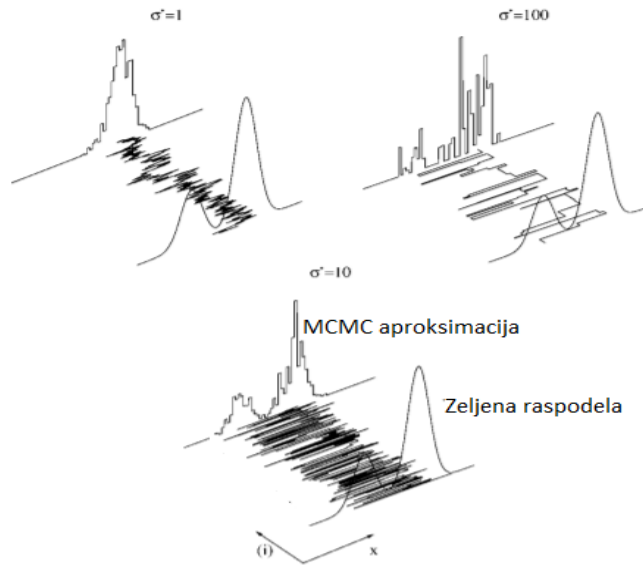
$$q(\theta^t | \theta^{(t-1)}) = g(\theta^t - \theta^{(t-1)}) = g(-(\theta^{(t-1)} - \theta^t)) = q(\theta^{(t-1)} | \theta^t),$$

pa verovatnoća prihvatanja koraka glasi

$$\frac{\pi(\theta^t)q(\theta^{(t-1)} | \theta^t)}{\pi(\theta^{(t-1)})q(\theta^t | \theta^{(t-1)})} = \frac{\pi(\theta^t)}{\pi(\theta^{(t-1)})}.$$

Raspodele sa teškim repovima mogu biti korisne u sprečavanju navedenih problema. Ideja da se disperzija koraka prilagođava u zavisnosti od ishoda lanca je u poslednje vreme jako popularna, ali treba biti oprezan, s obzirom na to da na ovaj način možemo poništiti Markovljevo svojstvo. Ovakve tehnike nazivaju se adaptivne MCMC tehnike.

## 4.5 Adaptivni MCMC algoritmi



Slika 12: Trajektorije lanca u zavisnosti od izbora  $\sigma$ .

Još jednom ćemo primetiti da trajektorija lanca zavisi umnogome od izbora disperzije predložene raspodele. Kada je disperzija dosta velika ( $\sigma = 100$ ), predloženi koraci se odbijaju i lanac ostaje u istom stanju (Slika 12, preuzeto iz rada [1]). Ideja adaptivnih MCMC metoda je da koriste informacije dobijene iz podataka, tako da unapređuju parametre predložene raspodele radi približavanja željenoj raspodeli ili smanjivanja disperzije ocene koju želimo da izračunamo nad željenom raspodelom. Međutim, korišćenjem prošlih informacija dovoljno često, kršimo Markovljevo svojstvo. Jedan od načina prevazilaženja ovog problema je da se adaptacija predložene raspodele vrši u fiksnom broju koraka na početku, a zatim da se sprovede regularna MCMC simulacija radi postizanja konvergencije. Za više detalja videti [1].

## 5 Primena na ocenjivanje parametara statističkih modela

### 5.1 Ocenjivanje parametara linearne regresije

Primer na kome će biti prikazana primena MCMC tehnika na ocenjivanje parametara statističkih modela biće jedan od osnovnih algoritama statističkog učenja - Linearna regresija.

Sve tehnike implementirane su u paketu TensorFlow Probability u programskom jeziku Pajton (eng. *Python*). Sav kod, kao i više o samoj biblioteci može se naći u poglavlju 8.

Linearna regresija predstavlja jedan od najjednostavnijih i najčešće korišćenih statističkih modela. Osnovni zadatak linearne regresije je modeliranje veze između ciljne promenljive  $y$  i atributa  $X$ , gde je  $X$  vektor pojedinačnih atributa  $X_i$ . Pretpostavljamo da se zavisnost ciljne promenljive  $y$  od atributa  $X$  može opisati normalnom raspodelom, odnosno da važi

$$(34) \quad p(y|X) \sim \mathcal{N}(f(x), \sigma^2),$$

gde je  $f$  funkcija koja uspostavlja vezu između atributa i očekivanja ciljne promenljive. Sastavni deo linearne regresije je da je model ima linearnu formu, odnosno:

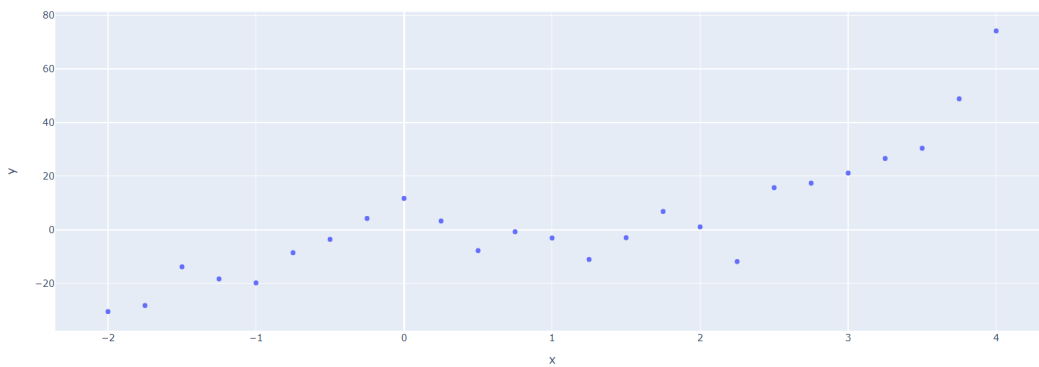
$$(35) \quad f_\beta(x) = \beta \cdot x.$$

Linearnost podrazumeva linearnost po koeficijentima. Najčešće se podrazumeva da je skup atributa proširen atributom koji je konstantne vrednosti 1, kako bi model sadržao slobodni koeficijent. Finalni oblik modela glasi:

$$(36) \quad p(y|x, \beta) = \mathcal{N}(\beta \cdot x, \sigma^2).$$

Radi boljeg prikaza kvaliteta MCMC tehnika generisaćemo naše podatke koje ćemo pokušati da aproksimiramo linearnom regresijom. Cilj nam je da ocenimo koeficijente linearne regresije. Ocene navedenih parametara linearne regresije biće dobijene iz aposteriornih raspodela parametara modela ocenjenih MCMC metodama. Kao što je navedeno u poglavlju 2.3 za računanje aposteriorne raspodele potrebno je definisati apriornu raspodelu i verodostojnost. Sve potrebne elemente algoritma navodimo nadalje u primeru.

Generisaćemo podatke koji mogu da se aproksimiraju polinomom trećeg stepena  $y = 2 - 4x + x^2 + 1.5x^3$ . Kako podaci u stvarnim problemima i skupovima podataka nikad ne odgovaraju jednoznačno jednoj funkciji, dodaćemo Gausov šum sa standardnom devijacijom 6.



Slika 13: Vrednosti tačaka koji mogu da se aproksimiraju polinomom trećeg stepena  $y = 2 - 4x + x^2 + 1.5x^3$

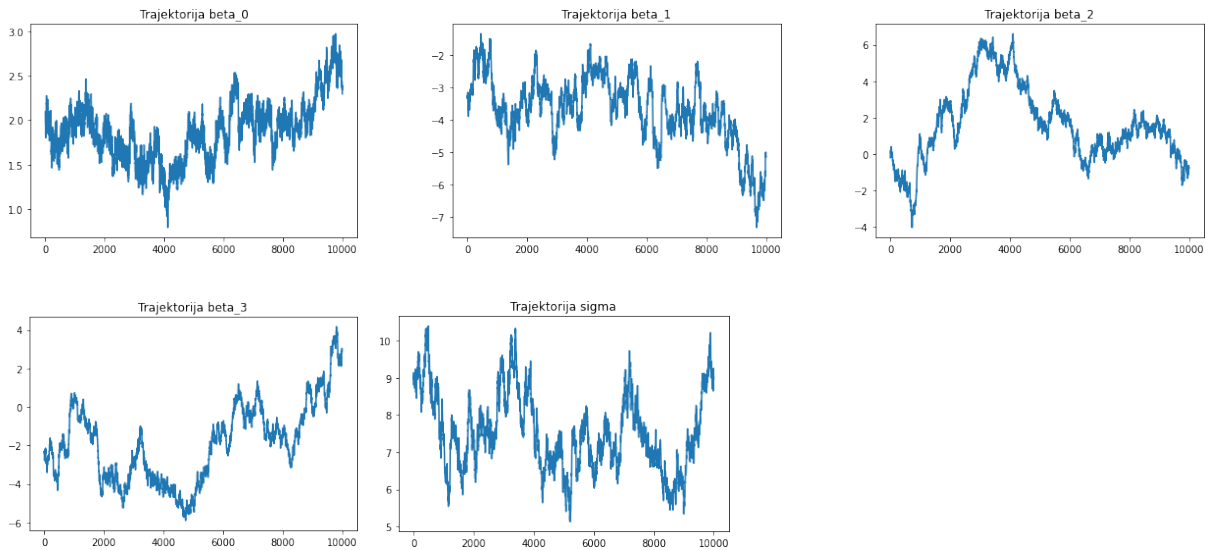
Parametri linearne regresije su kao što smo naveli koeficijenti  $\beta_i$ , gde u našem slučaju  $i \in \{0, 1, 2, 3\}$ . Pored koeficijenata želimo da ocenimo i standardnu devijaciju komponente šuma modela. Potrebno je da ocenimo aposteriorne raspodele navedenih parametara modela. Potrebno je definisati apriornu raspodelu

i verodostojnost. Radi lakšeg računa umesto funkcije verodostojnosti koristi se logaritam funkcije verodostojnosti. Ovo je česta praksa iz više razloga - kako se verodostojnost računa kao proizvod, jako male vrednosti ili 0 mogu dovesti do potkoračenja, analogno važi za velike vrednosti. Slično se definišu i apriorne raspodele u logaritamskom prostoru, da bismo izbegli operaciju množenja na više mesta. Pretpostavićemo da naši podaci imaju normalnu raspodelu - te je verodostojnost oblika  $\mathcal{N}(\mu, \sigma)$ . Nadalje je potrebno definisati apriorne raspodele za svaki od parametara. Radi konjugovanosti biramo normalnu raspodelu. Dodatno, kako nemamo neko apriorno znanje o samim koeficijentima, izabraćemo - normalnu raspodelu  $\mathcal{N}(0, 5)$ . Biramo da su apriorne raspodele koeficijenata međusobno nezavisne, radi jednostavnosti ocenjivanja. U slučaju višedimenzionih apriornih raspodela bilo bi potrebno ocenjivati i sve korelacije, što bi dosta povećalo kompleksnost i broj parametara koji se ocenjuju. Precizniji izbor apriorne raspodele u „pogrešnom smeru” može dovesti do znatno sporije konvergencije algoritma. Kako je standardna devijacija pozitivna, da bismo imali pozitivan nosač, za apriornu raspodelu  $\sigma$  biramo Log-Normalnu raspodelu (2,0.5). Inicijalna stanja parametara za sada biramo proizvoljno:

$$\begin{aligned}\beta_0 &= 0 \\ \beta_1 &= 0 \\ \beta_2 &= 0 \\ \beta_3 &= 0 \\ \sigma &= 1\end{aligned}$$

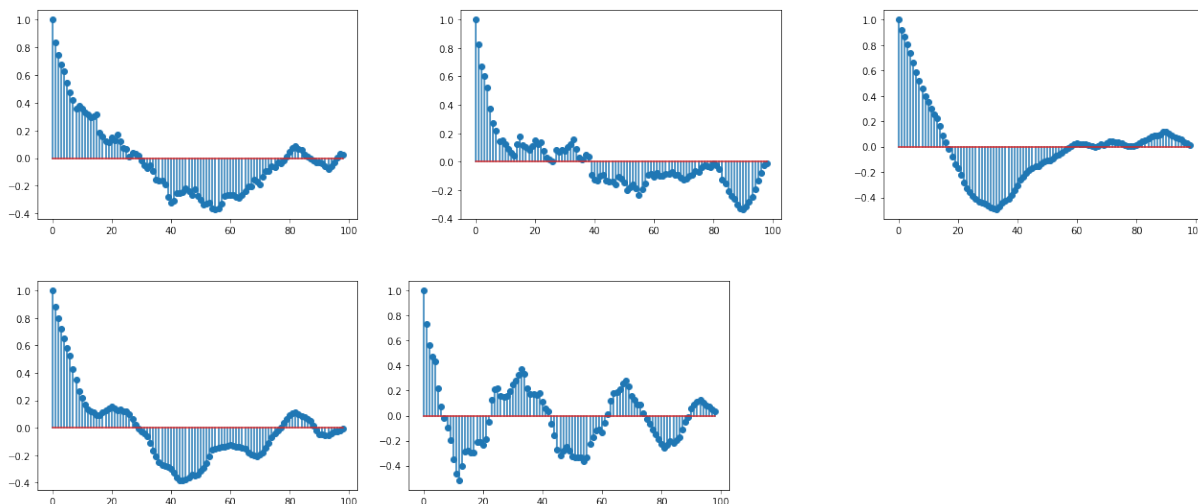
Za predloženu raspodelu  $q$  uzimamo normalnu raspodelu centriranu oko tekućeg uzorka sa standardnom devijacijom 0.1. Algoritam koji ćemo koristiti je algoritam Metropolisovog slučajnog lutanja (eng. *Random Walk Metropolis*) spomenut u sekciji 4. Definišemo broj iteracija da bude 10000 za početak. Odbacićemo prvih 10% uzoraka kao nereprezentativne (burn in).

Možemo da vidimo da nam je prihvaćeno skoro 58% uzoraka. Ovaj broj ne sme biti premali ( $<20\%$ ) jer je tada odbačen preveliki udeo uzoraka, međutim ne sme biti ni preveliki ( $>80\%$ ) kako bi uzorci bili što informativniji. Kako je ovo dosta širok opseg, nastavićemo sa drugim metodama evaluacije našeg modela.



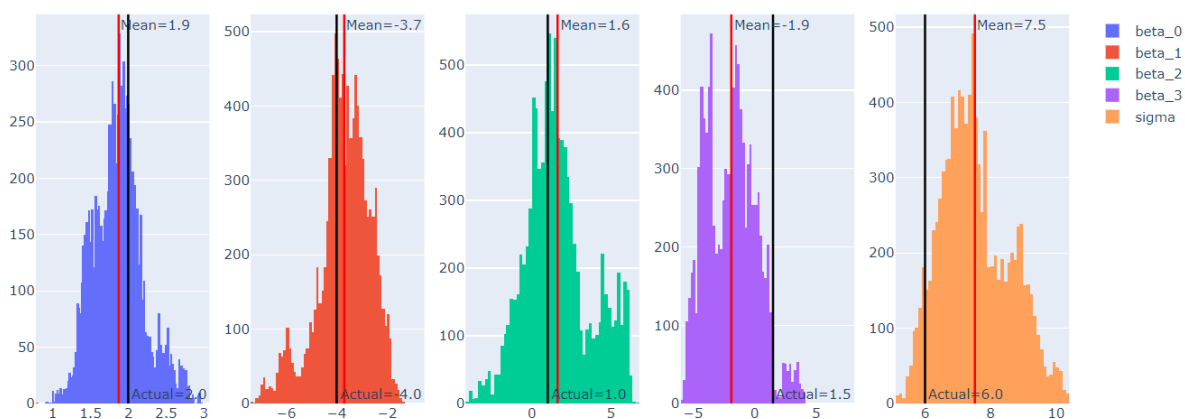
Slika 14: Trajektorije lanca MCMC

Na osnovu trajektorija pojedinačnih parametara (Slika 14) možemo da zaključimo da se konvergencija odvija sporo i da su uzorci dosta korelisani. U nastavku vidimo grafikone autokorelacione funkcije uzoraka po svakoj dimenziji (Slika 15). Kako je za iscrtavanje grafika korišćena veličina koraka 100 uzoraka očigledno je da su nam uzorci jako korelisani.



Slika 15: ACF

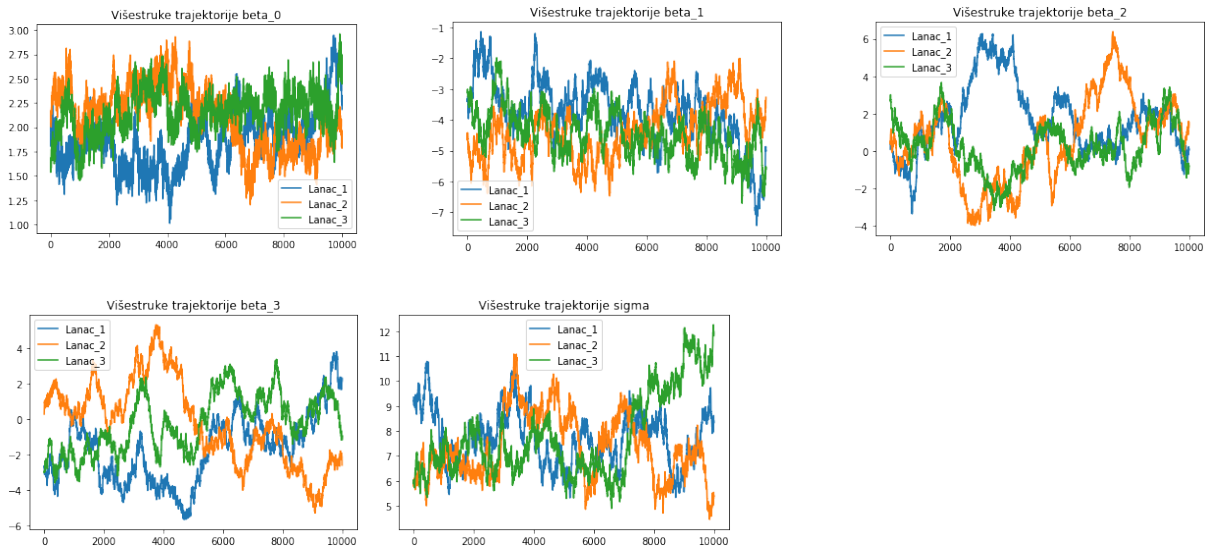
Kako smo sami generisali podatke, možemo uporediti dobijene raspodele sa stvarnom vrednošću parametra (Slika 16). Međutim, vidimo da još uvek nismo dostigli stanje stabilnosti i da se moda normalne aposteriorne raspodele ne poklapa u svim slučajevima sa stvarnom vrednošću parametra.



Slika 16: Uzorci aposteriorne raspodele

Radi ispravnije ocene konvergencije možemo pustiti više lanaca radi vizuelne i statističke provere konvergencije lanaca koristeći Rubin Gelmanovu statistiku, kao što smo naveli u glavi 4. Broj lanaca koje treniramo zavisi od komputativnih resursa kao i veličine samih lanaca. Ono što je prednost TFP paketa, jeste da možemo već definisane funkcije da modifikujemo, tako da se kreira i simulira više lanaca paralelno. Ono što je takođe bitno, u finalnoj predikciji mogu se koristiti uzorci iz svih lanaca. Biramo da pustimo 3 lanca. Broj lanaca se određuje empirijski, ne postoji jasan metod odabira ovog parametra. Početna stanja biramo iz uniformne raspodele da bi se odvijali međusobno nezavisno.





Slika 17: Trajektorije više lanaca

Rubin Gelmanove vrednosti iz Tabele 1 su veće od 1.1, što nam još jednom pokazuje lanci nisu dostigli konvergenciju i da su potrebne korekcije u algoritmu.

$\beta_0$	1.3820778
$\beta_1$	1.2448359
$\beta_2$	1.2382102
$\beta_3$	1.2619094
$\sigma$	1.0363044

Tabela 1: Tabela Rubin Gelmanovih vrednosti

Na osnovu autokorelacione funkcije možemo da procenimo koliko prihvaćenih uzoraka je bilo nezavisno. Medjusobnu zavisnost uzoraka računamo pomoću autokorelacione funkcije i predefinisane granice. - Ovo se još naziva i veličina efektivnog uzorka (eng. *effective sample size*). Iz tabele 2 vidimo da je veličina efektivnog uzorka procentualno mala.

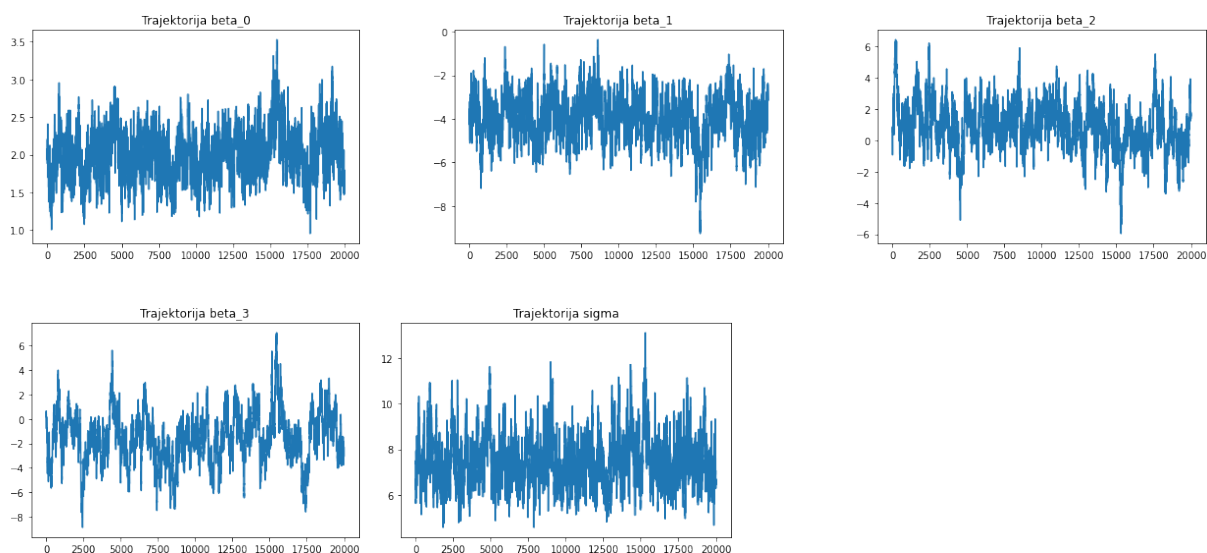
8.474174	9.652344	7.5595603	6.403948	17.626648
4.719346	7.5230517	5.357511	4.202653	11.738779
17.57335	12.0805645	12.215468	5.5638475	5.6712236

Tabela 2: Tabela ESS

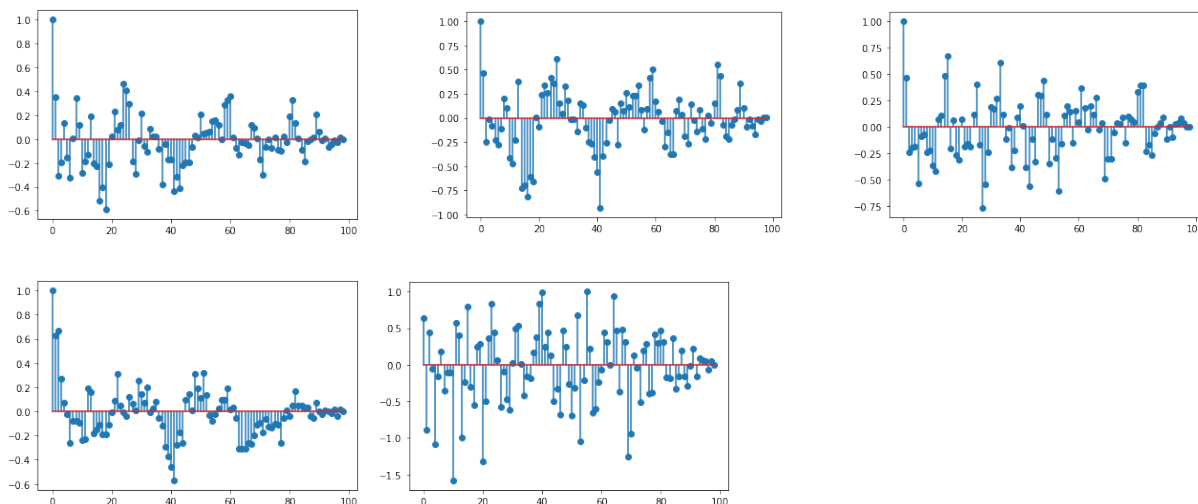
Ovo možemo popraviti na par načina:

- Uvesti parametar u funkciju koji uzima tek svaki  $n$ -ti uzorak u finalni lanac. Ovim postupkom se jedan deo uzoraka odbacuje, ali će svaki koji bude prihvaćen biti dosta informativniji u odnosu na ostale. Mana ovog pristupa je što nam je potrebno  $n$  puta više iteracija algoritma.
- Povećati broj iteracija da bismo ušli u period veće stabilnosti lanca. Mana ovog pristupa je što se period simulacije povećava.
- Povećati standardnu devijaciju predložene raspodele čime će se naredni uzorak potencijalno više „udaljavati” od prethodnog
- Korigovati apriorne raspodele

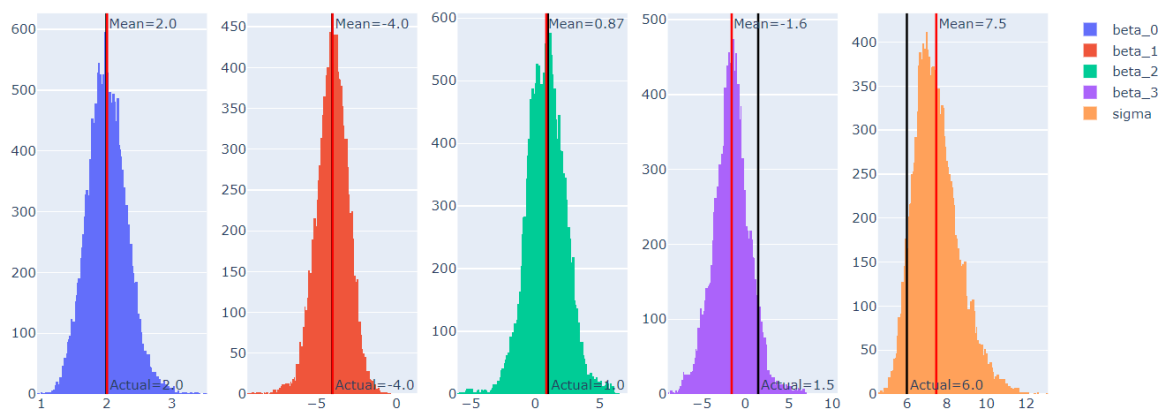
U narednoj iteraciji (V2) pokušaćemo da poboljšamo rezultate i performanse metoda koristeći neke od navedenih primera. Uvodimo parametar kojim uzimamo tek svaki 10 uzorak, da bismo izbegli visoko korelisane susedne uzorke (eng. *thinning*). Pored toga povećaćemo broj iteracija radi bolje konvergencije. Na sledećim graphicima može se videti, da su navedeni zahtevi ispunjeni. Možemo primetiti i da su eliminisane manje mode u aposteriornim raspodelama koje su se pojavljivale kao lokalni ekstremumi. Međutim usložnjavanje algoritma i povećavanje iteracija, u pojedinim slučajevima može biti skupo ili čak neizvodljivo u zavisnosti od dostupnih resursa.



Slika 18: Trajektorije V2

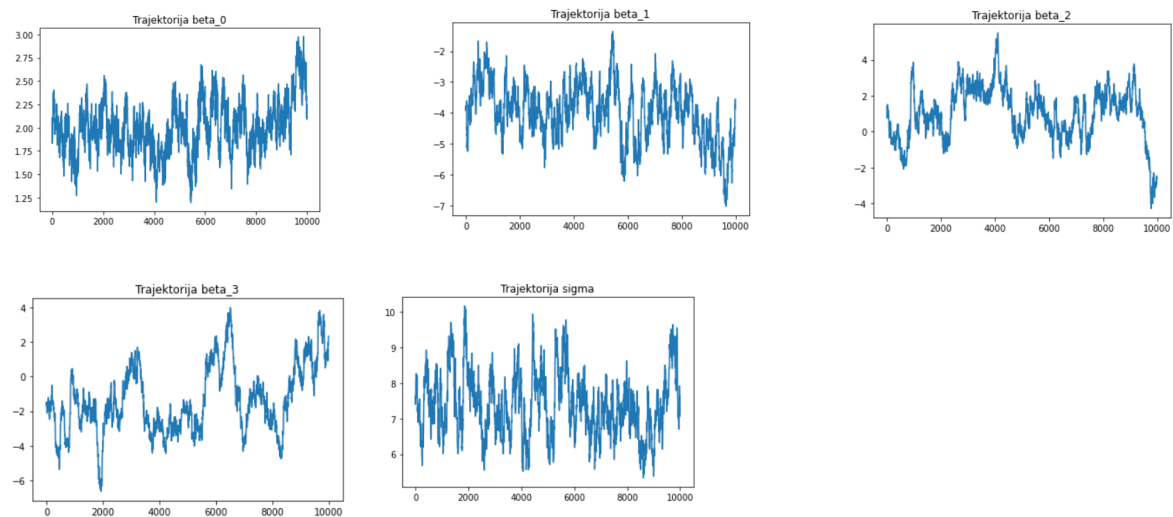


Slika 19: ACF V2

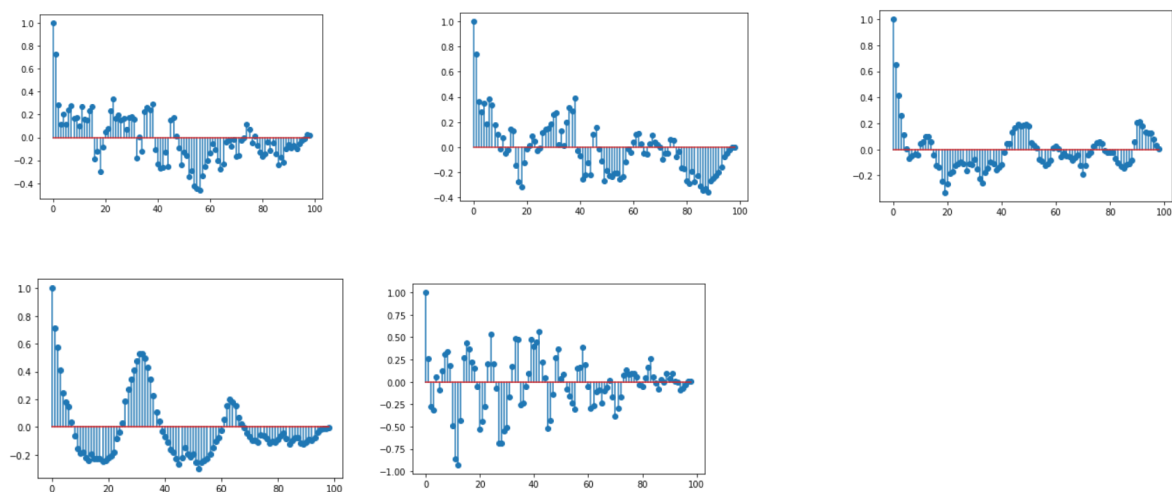


Slika 20: Uzorci aposteriorne raspodele V2

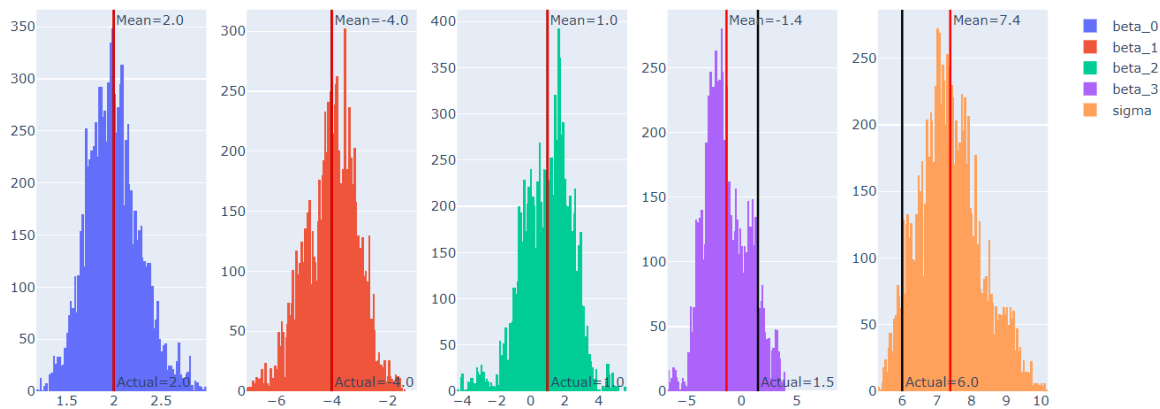
Drugi način uklanjanja korelacije je da povećamo standardnu devijaciju predložene raspodele, da bi uzorci više istraživali prostor (V3). Neka je predložena raspodela slučajno lutanje sa  $\sigma = 0.5$ . Sada je procenat prihvaćenih uzoraka svega 13%. Ovo je dosta mala vrednost, stoga ćemo ipak korigovati vrednost na 0.2. Udeo prihvaćenih uzoraka iznosi 35%. ACF se zaista smanjuje, međutim konvergencija se usporava - dakle potrebno je još uzoraka, slično kao i u prethodnoj verziji.



Slika 21: Trajektorije V3

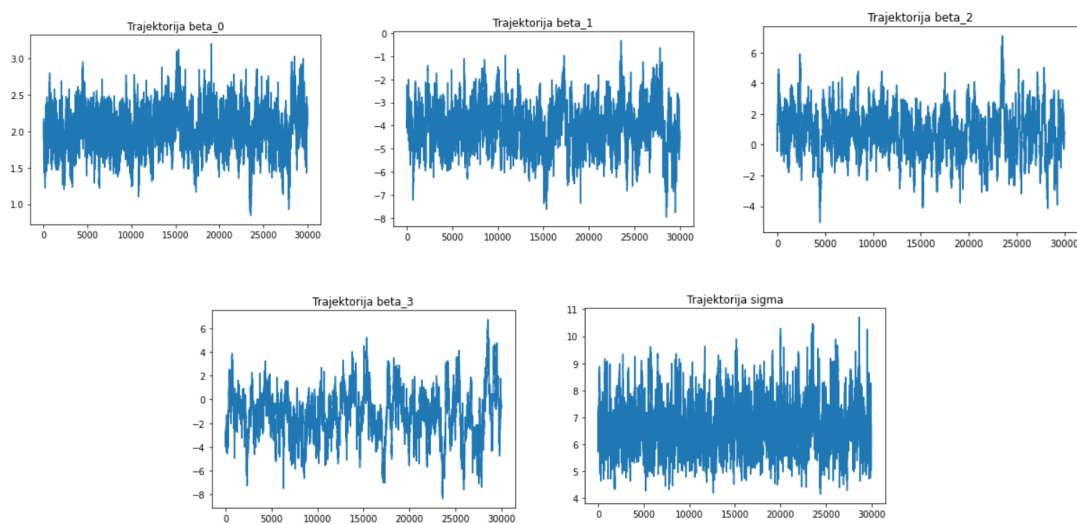


Slika 22: ACF V3

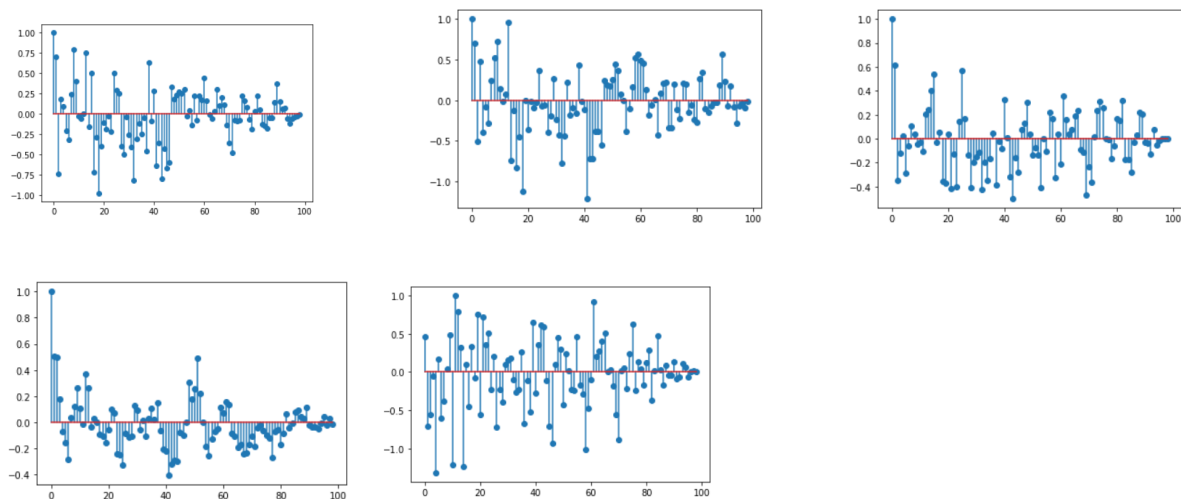


Slika 23: Uzorci aposteriorne raspodele V3

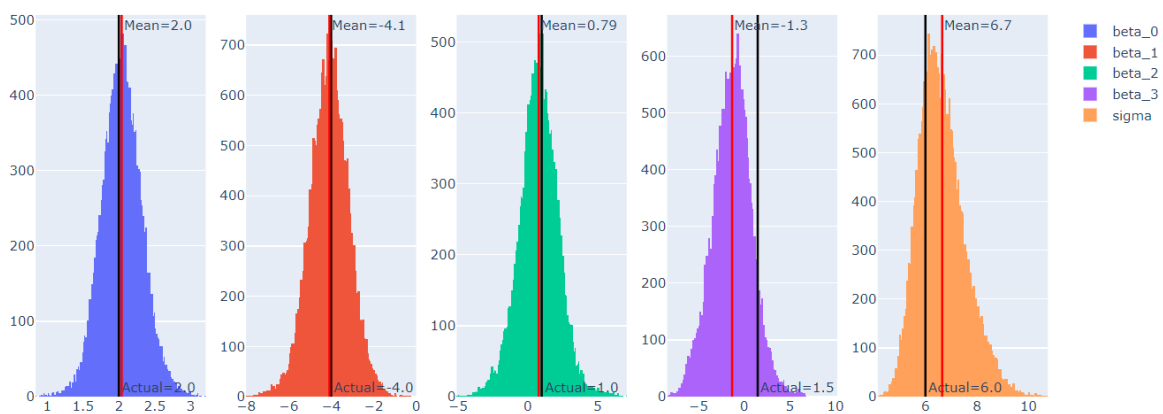
Dodatno, primetimo da standardna devijacija komponente šuma  $\sigma$  slabo prilazi svojoj pravoj srednjoj vrednosti i da vrednosti nisu stabilne. Stoga primenjujemo poslednju preporuku - korekciju apriorne raspodele za  $\sigma$  unutar verzije 2. (V4)



Slika 24: Trajektorije V4



Slika 25: ACF V4



Slika 26: Uzorci aposteriorne raspodele V4

Nezavisno od apriorne raspodele potrebno je više iteracija i ocenjeni parametri biće sve bliže stvarnoj raspodeli.

## 5.2 Ocenjivanje parametara mešavine normalnih raspodela

Mešavina normalnih raspodela često se koristi za modelovanje podataka koji nemaju normalnu raspodelu. Neka su nam dostupni podaci  $Y_1, \dots, Y_n$  koji su međusobno nezavisni i svaki od njih pripada jednoj od  $K$  komponenti odnosno populaciji. Pretpostavljamo da je raspodela unutar  $K$ -te populacije normalna  $\mathcal{N}(\mu_k, \sigma_k^2)$  i definišemo verovatnoću pripadanja  $K$ -toj populaciji sa  $p_k$ . Važi:

$$(37) \quad Y_i | \theta \sim \sum_{k=1}^K p_k \mathcal{N}(\mu_k, \sigma_k^2),$$

gde je  $\theta = (p_1, \dots, p_K, \mu_1, \dots, \mu_K, \sigma_1^2, \dots, \sigma_K^2)$ .

Cilj nam je da ocenimo parametre  $\theta$  koristeći Gibsovo uzorkovanje. Po Bajesovoj teoremi iz poglavlja 2.3 potrebno je da definišemo apriornu raspodelu i verodostojnost za sve parametre. Definišemo verodostojnost parametara na sledeći način:

$$(38) \quad p(y_1, \dots, y_n | \theta) = \prod_{i=1}^n p(y_i | \theta) = \prod_{i=1}^n \left( \sum_{k=1}^K \frac{p_k}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(y_i - \mu_k)^2}{2\sigma_k^2}\right) \right).$$

Definišemo apriornu raspodelu za  $\theta$  kao:

$$(39) \quad p(\theta) = p(p_1, \dots, p_K) \prod_{k=1}^K p(\mu_k, \sigma_k^2).$$

$p(p_1, \dots, p_K)$  ima Dirihleovu raspodelu  $\mathcal{D}(\gamma_1, \dots, \gamma_K)$  gde je  $\gamma_1, \dots, \gamma_K > 0$ , odnosno važi:

$$(40) \quad p(p_1, \dots, p_K) = \frac{\Gamma(\sum_{k=1}^K \gamma_k)}{\prod_{k=1}^K \Gamma(\gamma_k)} \prod_{k=1}^K p_k^{\gamma_k - 1},$$

tako da  $\{(p_1, \dots, p_K) : p_i \geq 0 \forall i, \sum_{k=1}^K p_k = 1\}$ . Apriorne raspodele za  $\mu_i$  i  $\sigma_i^2$ , gde  $i \in 1, \dots, K$  definišemo kao:

$$(41) \quad p(\mu_k, \sigma_k^2) = p(\mu_k | \sigma_k^2) p(\sigma_k^2),$$

gde je  $p(\mu_k | \sigma_k^2) = \mathcal{N}(\alpha_k, \frac{\sigma_k^2}{\lambda_k})$  i  $p(\sigma_k^2) = \mathcal{IG}(\frac{\lambda_k + 2}{3}, \frac{\beta_k}{2})$ .

Uvodimo pomoćne promenljive  $Z_1, \dots, Z_n$  koje će biti indikatori u kojoj populaciji se nalazi opservacija.

$$(42) \quad P(Z_i = k) = p_k, Y_i | Z_i = k \sim \mathcal{N}(\mu_k, \sigma_k^2).$$

Sada je tražena raspodela:

$$(43) \quad p(\theta, z_1, \dots, z_n | y_1, \dots, y_n) \propto \left( \prod_{i=1}^n \frac{p_{z_i}}{\sigma_{z_i}} \exp\left(-\frac{(y_i - \mu_{z_i})^2}{2\sigma_{z_i}^2}\right) \right) \prod_{k=1}^K p_k^{\gamma_k - 1} \times \prod_{k=1}^K \exp\left(-\frac{\lambda_k (\mu_k - \alpha_k)^2}{2\sigma_k^2}\right) \left(\frac{1}{\sigma_k^2}\right)^{\frac{\lambda_k + 3}{2} - 1} \exp\left(-\frac{\beta_k}{2\sigma_k^2}\right).$$

Možemo da implementiramo Gibsov algoritam uzorkovanja iterativnim uzorkovanjem iz sledećih raspodela:

- $P(z_{1:n} | y_{1:n}, \theta)$ ,
- $p(p_{1:n} | y_{1:n}, z_{1:n}, \mu_{1:K}, \sigma_{1:K}^2) = p(p_{1:n} | z_{1:n})$ ,
- $p(\mu_{1:K}, \sigma_{1:K}^2 | y_{1:n}, z_{1:n}, p_{1:n})$ ,

Važi:

$$(44) \quad P(z_{1:n} | y_{1:n}, \theta) = \prod_{i=1}^n P(z_i | y_i, \theta),$$

gde je

$$(45) \quad P(z_i|y_i, \theta) = \frac{\frac{p_{z_i}}{\sigma_{z_i}} \exp\left(\frac{-(y_i - \mu_{z_i})^2}{2\sigma_{z_i}^2}\right)}{\sum_{k=1}^K \frac{p_k}{\sigma_k} \exp\left(\frac{-(y_k - \mu_k)^2}{2\sigma_k^2}\right)}.$$

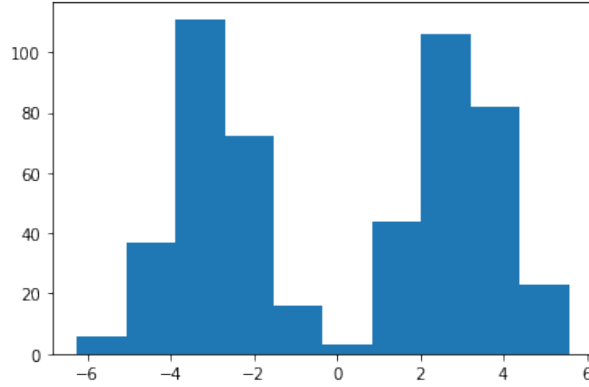
Uvodimo i:

- $n_k = \sum_{i=1}^n 1_{\{k\}}(z_i)$  (broj elemenata u k-toj komponenti),
- $n_k \bar{y}_k = \sum_{i=1}^n y_i 1_{\{k\}}(z_i)$ , (suma elemenata u k-toj komponenti)
- $s_k^2 = \sum_{i=1}^n (y_i - \bar{y}_k)^2 1_{\{k\}}(z_i)$ .

Sada možemo da uzorkujemo iterativno iz sledećih poznatih raspodela:

- $p_1, \dots, p_K | z_{1:n} \sim \mathcal{D}(\gamma_1 + n_1, \dots, \gamma_K + n_K)$ ,
- $\sigma_k^2 | z_{1:n}, y_{1:n} \sim \mathcal{IG}\left(\frac{\lambda_k + n_k + 3}{2}, \frac{\lambda_k s_k^2 + \beta_k + s_k^2 + (\lambda_k + n_k)^{-1} (\lambda_k \alpha_k + n_k \bar{y}_k)^2}{2}\right)$ ,
- $\mu_k | \sigma_k^2, z_{1:n}, y_{1:n} \sim \mathcal{N}\left(\frac{\lambda_k \alpha_k + n_k \bar{y}_k}{\lambda_k + n_k}, \frac{\sigma_k^2}{\lambda_k + n_k}\right)$ .

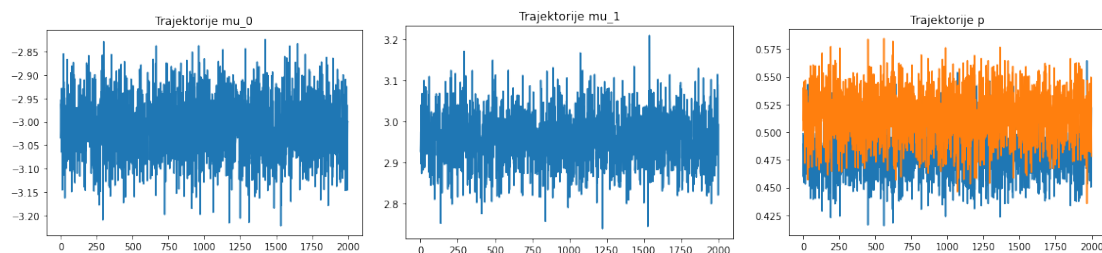
Ilustrovaćemo primer za  $K=2$ . Sav kod je implementiran u Pajtonu i nalazi se u dodatku. Generisaćemo podatke iz mešavine Gausovih raspodela sa srednjim vrednostima u -3 i 3 i jediničnom standardnom devijacijom.



Slika 27: Generisani podaci

Radi jednostavnosti pretpostavićemo da je  $\sigma_k = 1, \forall k$ . Ovaj parametar ćemo fiksirati i nećemo uzorkovati kroz gore navedeni proces. On će se svakako koristiti u uslovnim raspodelama ostalih parametara. Takođe, neka su  $\alpha_k = 0$  i  $\lambda_k = 1, \forall k$ . Tada apriorna raspodela za svako  $\mu_k$  postaje  $\mathcal{N}(0, 1)$ . Za početna stanja izabraćemo proizvoljne vrednosti:  $\mu_1 = -2$ ,  $\mu_2 = 2$ ,  $p_1 = 0.7$ ,  $p_2 = 0.3$ , dok ćemo podatke nasumično podeliti u komponente 0 i 1. Kako su inicijalni parametri dosta zgodno postavljeni, već nakon 2000 iteracija, vidimo da su vrednosti za  $\mu_1$  i  $\mu_2$  iskonvergirale ka svojim stvarnim vrednostima. Dodatno,  $p_1$  i  $p_2$  su se približile jedna drugoj, odnosno 0.5.





Slika 28: Trajektorije parametara

## 6 Zaključak

MCMC algoritmi nude široke mogućnosti za efikasno ocenjivanje gustine aposteriorne raspodele. Nadalje koristeći dobijene uzorke aposteriorne raspodele, pokušali smo da ocenimo parametre linearne regresije nad generisanim podacima. Osnovni izazov svakako predstavlja konvergencija algoritma. Pokazali smo više načina za proveru konvergencije lanca, kako statistički tako i vizuelno. Brzina konvergencije i efikasnost algoritma umnogome zavise od parametara samog algoritma - inicijalna stanja, dužina lanca, definicija apriorne raspodele, itd. Pokušali smo da prikazemo proces optimizacije ovih parametara kroz navedeni primer linearne regresije. Iako smo u primeru koristili osnovnu implementaciju Metropolis-Hejstings algoritma, u radu se spominju i naprednije metode kao što su Hamiltonovski Monte Karlo. S obzirom na to da je cilj rada prikaz osnovnih metoda MCMC i njihovih primena, nisu prikazane različite naprednije tehnike koje su svakako zanimljive za dalje istraživanje (NUTS, Replica exchange, itd. [22]). Akcenat na primene se ogleda kroz više iteracija ocenjivanja parametara linearne regresije, kao i prateći kod koji se može naći u dodatku. Koriste se najsavremenije biblioteke u programskom jeziku Pajton, ali je pokazano i kako možemo samostalno implementirati navedene algoritme.

## A Dodatak A: Pregled biblioteke TensorFlow probabilitety i primeri

TensorFlow Probability (TFP) je python biblioteka izgrađena unutar TensorFlow biblioteke. Po-drazumevamo osnovno razumevanje principa paketa TensorFlow.

Ova biblioteka je namenjena statističkoj analizi i kreiranju probabilističkih modela. Između ostalog, TFP sadrži:

- Veliki broj raspodela
- Algoritme varijacionog zaključivanja i MCMC
- Različite algoritme za optimizaciju

TFP je pogodan za kombinovanje probabilističkih modela i dubokog učenja, kako su u njemu sadržane i sve standardne TensorFlow funkcije.

Unutar modula `tfd.distributions` sadržane su različite raspodele, koje možemo da uzorkujemo, računamo vrednosti gustine i funkcije raspodele u pojedinačnim tačkama, itd.

```
import tensorflow as tf
import tensorflow_probability as tfp
tfd = tfp.distributions

n = tfd.Normal(loc=0., scale=1.)
n
<tfp.distributions.Normal 'Normal' batch_shape=[] event_shape=[] dtype=float32>
n.sample(3)
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([-1.4658079, -0.5653636,  0.9314412],
            dtype=float32)>
n.log_prob(0.)
<tf.Tensor: shape=(), dtype=float32, numpy=-0.9189385>

nd = tfd.MultivariateNormalDiag(loc=[0., 10.], scale_diag=[1., 4.])
nd
<tfp.distributions.MultivariateNormalDiag 'MultivariateNormalDiag' batch_shape=[]
event_shape=[2] dtype=float32>
```

Primitimo da su izlazne vrednosti svih funkcija klase `Tensor` (vektori) i da je potrebno voditi računa o dimenzijama izlaza i mogućim operacijama nad njima.

Unutar modula `tfp.mcmc` možemo naći osnovne algoritme MCMC:

- MetropolisHastings
- HamiltonianMonteCarlo
- NoUTurnSampler
- RandomWalkMetropolis
- ReplicaExchangeMC

### A.1 Primena na ocenjivanje parametara linearne regresije

Generisanje podataka:

```
def forward_model(x_values, coefficients):
    """
    Definisanje polinoma n-tog stepena, gde se n odredjuje duzinom vektora koeficijenata.

    Args:
        x_values (tf.Tensor): Vrednosti x-ose
        coefficients (tf.Tensor): Koeficijenti polinoma
    """
    return tf.math.polyval(tf.unstack(coefficients), x_values)
```

```

# vrednosti x-ose
x_values = tf.linspace(-2.0, 4.0, 25)

# Vrednosti beta koeficijena
true_coefficients = tf.constant(np.array([2.0, -4.0, 1.0, 1.5]), tf.float32)

# Standardna devijacija komponente suma
true_sigma = 6.0

# y vrednosti bez suma
idealized_data = forward_model(x_values, true_coefficients)

# y vrednosti sa sumom
y_values = tfd.Normal(loc=idealized_data, scale=true_sigma).sample()

```

Definisanje verodostojnosti podataka:

```

def log_likelihood(x_values, y_values, coefficients, sigma):
    idealized_data = forward_model(x_values, coefficients)
    log_likelihoods = tfd.Normal(loc=idealized_data, scale=sigma).log_prob(y_values)
    # Koristimo operaciju sum zato to se nalazimo u log prostoru
    return tf.math.reduce_sum(log_likelihoods)

```

Definisanje apriorne raspodele:

```

def log_prior(coefficients, sigma):
    log_coefficients_prior = tfd.Normal(loc=0.0, scale=5.0).log_prob(coefficients)
    log_sigma_prior = tfd.LogNormal(loc=2, scale=.5).log_prob(sigma)
    return tf.math.reduce_sum(log_coefficients_prior) + log_sigma_prior

```

Računanje aposteriorne raspodele:

```

def log_posterior(parameters):
    coefficients = parameters[:-1]
    sigma = parameters[-1]
    return log_likelihood(x_values, y_values, coefficients, sigma) + log_prior(
        coefficients, sigma)

```

Inicijalna stanja:

```

initial_state = np.array([0.0, 0.0, 0.0, 0.0, 1.0], dtype=np.float32)

```

Definicija lanca:

```

kernel = tfp.mcmc.RandomWalkMetropolis(
    # Koristimo gore izracunatu aposteriornu funkciju
    target_log_prob_fn=log_posterior,

    # Za predlozenu raspodelu koristimo ugradjenu normalnu raspodelu
    random_walk_normal_fn
    new_state_fn=tfp.mcmc.random_walk_normal_fn(scale=0.1)
)

```

Funkcija simulacije MCMC:

```

@tf.function
def run_chain(initial_state, num_samples=10000, num_burnin_steps=1000):
    return tfp.mcmc.sample_chain(
        num_results=num_samples,
        num_burnin_steps=num_burnin_steps,
        num_steps_between_results=10,
        current_state=initial_state,
        kernel=kernel,
        trace_fn=lambda current_state, kernel_results: kernel_results
    )

```

Izvršavanje funkcije:

```
samples, kernel_results = run_chain(initial_state)
```

Procenat prihvatanja uzoraka:

```
kernel_results.is_accepted.numpy().sum() / len(kernel_results.is_accepted)
```

Iscrtavanje histograma:

```
samples_array = samples.numpy()
fig = make_subplots(rows=1, cols=samples_array.shape[1])
for i in range(samples_array.shape[1]):
    mean = samples_array[:, i].mean()
    if i < samples_array.shape[1] - 1:
        title = f"beta_{i}"
    else:
        title = "sigma"
    hist = go.Histogram(x=samples_array[:, i], name=title)
    fig.append_trace(hist, 1, i+1)
    if i < samples_array.shape[1] - 1:
        actual = true_coefficients.numpy()[i]
        fig.add_vline(actual, 1, i + 1, annotation_text=f"Actual={actual:.2}",
            annotation_position="bottom right", line={"color": "black"})
    else:
        fig.add_vline(true_sigma, 1, i + 1, annotation_text=f"Actual={true_sigma:.2}",
            annotation_position="bottom right", line={"color": "black"})
        fig.add_vline(mean, 1, i + 1, annotation_text=f"Mean={mean:.2}", line={"color": "red"})
fig.update_layout(title="Posterior samples")
```

Iscrtavanje trajektorija:

```
beta_0_trace=pd.DataFrame(samples.numpy())[0]
beta_0_trace.plot()
```

Iscrtavanje autokorelacione funkcije:

```
def acorr(x, ax=None):
    if ax is None:
        ax = plt.gca()

    x = x - x.mean()

    autocorr = np.correlate(x, x, mode='full')
    autocorr = autocorr[x.size:]
    autocorr /= autocorr.max()

    return ax.stem(autocorr)
```

```
fig, ax = plt.subplots()
acorr(beta_0_trace[range(0,10000,100)])
plt.show()
```

Puštanje više lanaca istovremeno:

```
def forward_model(x_values, coefficients):
    return tf.vectorized_map(lambda c: tf.math.polyval(tf.unstack(c), x_values),
        coefficients)

def log_likelihood(x_values, y_values, coefficients, sigma):
    idealized_data = forward_model(x_values, coefficients)
    sigma = tf.ones((coefficients.shape[0], x_values.shape[0])) * sigma[:, None]
    log_likelihoods = tfd.Normal(loc=idealized_data,
        scale=sigma).log_prob(y_values)
    return tf.math.reduce_sum(log_likelihoods, axis=1)
```

```

def log_prior(coefficients, sigma):
    log_coefficients_prior = tfd.Normal(loc=0.0, scale=5.0).log_prob(coefficients)
    log_sigma_prior = tfd.LogNormal(loc=2, scale=5).log_prob(sigma)
    return tf.math.reduce_sum(log_coefficients_prior, axis=1) + log_sigma_prior

def log_posterior(parameters):
    coefficients = parameters[:, :-1]
    sigma = parameters[:, -1]

    l = log_likelihood(x_values, y_values, coefficients, sigma)
    p = log_prior(coefficients, sigma)

    return l + p

kernel = tfp.mcmc.RandomWalkMetropolis(
    # Aposteriorna raspodela
    target_log_prob_fn=log_posterior,

@tf.function
def run_chain(initial_state, num_samples=10000, num_burnin_steps=1000):
    return tfp.mcmc.sample_chain(
        num_results=num_samples,
        num_burnin_steps=num_burnin_steps,
        num_steps_between_results=10,
        current_state=initial_state,
        kernel=kernel,
        trace_fn=lambda current_state, kernel_results: kernel_results
    )

n_chains = 3

def mk_init_state():
    return np.array([
        np.random.uniform(-2, 2), # beta0
        np.random.uniform(-2, 2), # beta1
        np.random.uniform(-2, 2), # beta2
        np.random.uniform(-2, 2), # beta3
        np.random.uniform(0, 1)  # sigma
    ], dtype=np.float32)

initial_state = np.vstack([mk_init_state() for _ in range(n_chains)])

samples, kernel_results = run_chain(initial_state)

```

ESS:

```
tfp.mcmc.effective_sample_size(samples)
```

## A.2 Primena na ocenjivanje parametara mešavine Gausovih raspodela

Definisanje modela:

```

class State:
    def __init__(self, z, mu, sigma_sq_mu, sigma_sq_n, pi):
        self.z = z # Indikator pripadnosti komponentama
        self.mu = mu # Srednja vrednost komponente
        self.sigma_sq_mu = sigma_sq_mu # Disperzija unutar komponente
        self.sigma_sq_n = sigma_sq_n # Disperzija podataka
        self.pi = pi # Verovatnoce komponenti

class GaussianDistribution:
    def __init__(self, mu, sigma_sq):

```

```

    self.mu = mu
    self.sigma_sq = sigma_sq
def log_p(self, x):
    return -0.5 * np.log(2*np.pi) + \
        -0.5 * np.log(self.sigma_sq) + \
        -0.5 * (x - self.mu) ** 2 / self.sigma_sq
def sample(self):
    return np.random.normal(self.mu, np.sqrt(self.sigma_sq))

class DirichletDistribution: #K=2
    def __init__(self, alpha):
        self.alpha = alpha
    def log_p(self, x):
        return -scipy.special.gamma(self.alpha)+self.alpha*np.log(x)
    def sample(self):
        return np.random.dirichlet(self.alpha)

class MultinomialDistribution: #K=2
    def __init__(self,pi,n):
        self.pi=pi
        self.n=n

    def from_log_odds(x):
        sumexp=np.exp(x).sum(1)
        sumexp2=np.vstack((sumexp, sumexp)).T
        final=np.exp(x)/sumexp2
        return np.array(final)

    def sample(self):
        s=np.random.multinomial(self.n,self.pi)
        return np.where(s==1)[0]

class InverseGammaDistribution:
    def __init__(self,a,b):
        self.a=a
        self.b=b
    def sample(self):
        return stats.invgamma(self.a,self.b).rvs()

class Model:

    def __init__(self, alpha, K, sigma_sq_mu_prior, sigma_sq_n_prior):
        self.alpha = alpha # Parametar Dirihleove raspodele
        self.K = K # broj komponenti
        self.sigma_sq_mu_prior = sigma_sq_mu_prior
        self.sigma_sq_n_prior = sigma_sq_n_prior

    def cond_pi(self, state):
        counts = np.bincount(state.z)
        counts.resize(self.K)
        return DirichletDistribution(self.alpha + counts)
    def cond_z(self, state, X):
        nax = np.newaxis
        prior = np.log(state.pi)
        evidence = GaussianDistribution(state.mu[nax, :, :], state.sigma_sq_n).log_p(X[:, nax
        ,:]).sum(2)
        flo=MultinomialDistribution.from_log_odds(x=(prior[nax, :] + evidence))
        ss=[]
        for i in range(data.shape[0]):
            s=MultinomialDistribution((flo[i]),self.K-1)#.sample()
            ss.append(s)
        return ss
    def cond_mu(self, state, X):
        ndata, ndim = X.shape
        #print(ndim)
        h = np.zeros((self.K, ndim))
        #print(h)
        lam = np.zeros((self.K, ndim))

```

```

for k in range(self.K):
    idxs = np.where(state.z == k)[0]
    if idxs.size > 0:
        h[k, :] = X[idxs, :].sum(0) / state.sigma_sq_mu
        lam[k, :] = idxs.size / state.sigma_sq_mu + 1. / state.sigma_sq_mu
    else:
        h[k, :] = 0.
        lam[k, :] = 1. / state.sigma_sq_mu
    return GaussianDistribution(h / lam, 1. / lam)

def gibbs_step(self, state, X):
    state.pi = self.cond_pi(state).sample()
    for i in range(data.shape[0]):
        state.z[i] = self.cond_z(state, X)[i].sample()
    state.mu = self.cond_mu(state, X).sample()

```

Poziv modela i inicijalna stanja:

```

sigma_sq_mu_prior=InverseGammaDistribution(2,1)
sigma_sq_n_prior=InverseGammaDistribution(1,1)
model = Model([0.7,0.3],2,sigma_sq_mu_prior,sigma_sq_n_prior)
#Inicijalna stanja
K=2
initial_state=State(np.random.choice([0,1],500), np.array([[2],[2]]), 1, 1, [0.7,0.3])

```

Generisanje podataka:

```

np.random.seed(42)

k = 2
ndata = 500
spread = 3
centers = np.array([-spread, spread])

v = np.random.randint(0, k, ndata)

data = centers[v] + np.random.randn(ndata)
nax=np.newaxis
data=data[:,nax]

```

Kreiranje lanca:

```

num_iterations=2000
pis=[]
zs=[]
mus=[]
sigma_sq_mus=[]
sigma_sq_ns=[]

for i in range(num_iterations):
    print(i)
    model.gibbs_step(initial_state,data)
    pis.append(initial_state.pi)
    zs.append(initial_state.z)
    mus.append(initial_state.mu)
    sigma_sq_mus.append(initial_state.sigma_sq_mu)
    sigma_sq_ns.append(initial_state.sigma_sq_n)

```

## A.3 Implementacija HMC u Pajtonu

Implementacija skokovite integracije:

```

def leapfrog(x, v, gradient, timestep, trajectory_length):
    v -= 0.5 * timestep * gradient(x)
    for _ in range(trajectory_length - 1):
        x += timestep * v
        v -= timestep * gradient(x)
    x += timestep * v
    v -= 0.5 * timestep * gradient(x)

```

```
return x, v
```

Uzorkovanje pomoću Hamiltonovskog Monte Karlo algoritma:

```
def sample_HMC(x_old, log_prob, log_prob_gradient, timestep, trajectory_length):
    # definicije iz fizike
    def E(x): return -log_prob(x)
    def gradient(x): return -log_prob_gradient(x)
    def K(v): return 0.5 * np.sum(v ** 2)
    def H(x, v): return K(v) + E(x)

    # Verovatnoca prihvatanja za Metropolis algoritam u logaritamskom prostoru
    def log_p_acc(x_new, v_new, x_old, v_old):
        return min(0, -(H(x_new, v_new) - H(x_old, v_old)))

    # Slučajno odabrano inicijalno stanje za impuls iz normalne raspodele p(v)
    v_old = np.random.normal(size=x_old.shape)

    # Aproksimacije novih pozicija nakon simulacije skokovite integracije posle
    # trajectory_length koraka veli ine timestep
    x_new, v_new = leapfrog(x_old.copy(), v_old.copy(), gradient,
                           timestep, trajectory_length)

    # Prihvatanje na osnovu verovatnoce prihvatanja
    accept = np.log(np.random.random()) < log_p_acc(x_new, v_new, x_old, v_old)

    # U lanac uključujemo samo vrednosti x, koje su nam od interesa
    if accept:
        return accept, x_new
    else:
        return accept, x_old
```

Kreiranje lanca:

```
def build_HMC_chain(init, timestep, trajectory_length, n_total, log_prob, gradient):
    n_accepted = 0
    chain = [init]

    for _ in range(n_total):
        accept, state = sample_HMC(chain[-1].copy(), log_prob, gradient,
                                   timestep, trajectory_length)
        chain.append(state)
        n_accepted += accept

    acceptance_rate = n_accepted / float(n_total)

    return chain, acceptance_rate
```

Primer dvodimenzionalne gausove raspodele:

```
def log_prob(x): return -0.5 * np.sum(x ** 2)
def log_prob_gradient(x): return -x
chain, acceptance_rate = build_HMC_chain(np.array([5.0, 1.0]), 1.5, 10, 10000,
                                       log_prob, log_prob_gradient)
```



## Literatura

- [1] An introduction to MCMC for Machine Learning, 2003, Cristophe Andrieu (University of Bristol), Nando de Freitas (University of British Columbia), Arnaud Doucet (University of Melbourne), Michael Jordan (University of California at Berkeley)
- [2] A Conceptual Introduction to Markov Chain Monte Carlo Methods, 2020, Joshua S. Speagle, Harvard,
- [3] Advanced Simulation Methods, 2020, George Deligiannidis, Oxford
- [4] Markov Chain Monte Carlo Analysis of Option Pricing Models, 2007, Matthew C. Pollard and Tom Smith (supervisor), Department of Finance and Applied Statistics, Australian National University
- [5] MCMC and Bayesian Modeling, 2017, Martin Haugh, Columbia
- [6] Pattern Recognition and Machine Learning, 2006, Christopher M. Bishop
- [7] Doing Bayesian Data Analysis (Second Edition), 2015, John K. Kruschke, Indiana University
- [8] Stohastički modeli u operacionim istraživanjima, Lenka Glavaš, Slobodanka Janković, 2016, MATF
- [9] Monte Carlo Statistical Methods, 2004, Robert, Christian, Casella, George
- [10] Monte Carlo sampling, 2010, Michael Jordan, Berkeley
- [11] Primena MCMC metoda u finansijskoj matematici, Katarina Svorcan, Marko Radosavljević, Elementi finansijske matematike, Matematički fakultet, 2017
- [12] tfp.mcmc: Modern Markov Chain Monte Carlo Tools Built for Modern Hardware, 2020, Junpeng Lao, Christopher Suter, Ian Langmore, Cyril Chimisov, Ashish Saxena, Pavel Soutsov, Dave Moore, Rif. A. Saurous, Matthew D. Hoffman, Joshua V. Dillon, Google Research
- [13] <https://www.tensorflow.org/probability/>
- [14] <https://towardsdatascience.com/an-overview-of-monte-carlo-methods-675384eb1694>
- [15] <https://nicercode.github.io/guides/mcmc/>
- [16] Coursera NHE Introduction to bayesian learning
- [17] Intro to TensorFlow Probability and Bayesian Modeling, 2020, Simeon Carstens, Darron Howell, Data science conference
- [18] Mašinsko učenje, 2019, Mladen Nikolić, Andjelka Zecevic, Matematički fakultet
- [19] <https://chi-feng.github.io/mcmc-demo/app.html>
- [20] MCMC using Hamiltonian dynamics, 2012, Radford M. Neal, University of Toronto
- [21] On thinning of chains in MCMC, 2012, William A. Link and Mitchell J. Eaton
- [22] Introduction to Markov Chain Monte Carlo (MCMC) sampling, part 3: Hamiltonian Monte Carlo, 2020, Simeon Carstens
- [23] Testing MCMC code, 2014, Roger B. Grosse, David K. Duvenaud
- [24] Bayesian reasoning and Machine Learning, 2012, David Barber

## Biografija

Katarina Svorcan je rođena u Beogradu 3. aprila 1994. godine, gde završava osnovnu školu „Starina Novak”, sa Vukovom diplomom. Završila je Petu beogradsku gimnaziju, prirodno matematički smer, sa Vukovom diplomom. 2013. godine upisuje Matematički fakultet u Beogradu, na modulu Statistika, aktuarska i finansijska matematika. Osnovne akademske studije završava u julu 2017. godine, sa prosečnom ocenom 9.07. 2017. godine upisuje master studije, na istom modulu. Položila je sve ispite na master studijama, sa prosečnom ocenom 9. Paralelno sa master studijama, započinje i profesionalnu karijeru u oblasti nauke o podacima.