Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Generovanie realizácií rovnomerného rozdelenia pravdepodobnosti na mnohorozmerných polyédroch

Bakalárska práca

2018
Slavomír Hanzely

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Generovanie realizácií rovnomerného rozdelenia pravdepodobnosti na mnohorozmerných polyédroch

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Školiteľ:	doc. Mgr. Radoslav Harman, PhD.

Bratislava, 2018
Slavomír Hanzely

55658269

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Slavomír Hanzely
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Generovanie realizácií rovnomerného rozdelenia pravdepodobnosti na mnohorozmerných polyédroch
Random sampling from the uniform distribution on multidimensional polyhedra

Anotácia: V Monte-Carlo metódach výpočtu pravdepodobností a v znáhodnených optimalizačných metódach je často potrebné generovať realizácie z rovnomerného rozdelenia na mnohorozmerných polyédroch. Tieto polyédre môžu byť zadané buď systémom konečného počtu lineárnych nerovníc (takzvaná H-reprezentácia), alebo ako konvexný obal konečnej množiny bodov (takzvaná V-reprezentácia). V prípade oboch typov reprezentácií je rovnomerné generovanie vo vnútri všeobecného polyédra netriviálna úloha, kombinujúca techniky a poznatky z matematiky, štatistiky a informatiky.

Cieľ: Cieľom bakalárskej práce je: Po prvé vypracovať prehľad existujúcich prístupov generovania realizácií z rovnomerného rozdelenia na polyédroch (priame generovanie pre špeciálne polyédre, zamietacie algoritmy, MCMC algoritmy a iné); po druhé vypracovať a programovo implementovať vlastnú metódu založenú na elipsoide najmenšieho objemu obsahujúceho zadaný polyéder.

Vedúci: doc. Mgr. Radoslav Harman, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Daniel Ševčovič, DrSc.

Dátum zadania: 14.10.2018

Dátum schválenia: 24.10.2018 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie:

Abstrakt

Kľúčové slová:

Abstract

Keywords:

Obsah

Úvod

V rámci tejto práce sa budeme zaoberať metódami na rovnomerné generovanie bodov vo veľarozmernom polyédri (konvexnom mnohostene). Našou úlohou je vytvoriť generátor, ktorý bude čo najrýchlejšie generovať body vo vnútri polyédru rovnomerne náhodne, tj. pravdepodobnosť, že dostaneme bod vnútri ľubovoľnej oblasti polyédra je lineárne závislá iba od objemu danej časti.

Vo všeobecnosti možno polyéder reprezentovať viacerými spôsobmi, napríklad ako konvexný obal bodov (V–reprezentácia) alebo ako sústavu lineárnych nerovníc (H reprezentáca). Obidve spomenuté reprezentácie možno v prípade potreby previesť na tú druhú. Prevod medzi nimi síce nie je lacný, no daný výpočet je nutné spraviť len raz pred začatím generovania. Pre účely tejto práce budeme pracovať s polyédrom reprezentovaným sústavou lineárnych nerovníc, riešení systému $Ax \leq b$ ($x \in X$ ak $Ax \leq b$). Generovanie bodov v polyédre predstavuje generovanie bodov, ktoré spĺňajú sústavu lineárnych obmedzení (viď H–reprezentácia polyédru).

Rovnomerné generovanie bodu v polyédre je problém s prirodzeným uplatnením v praxi. Mnoho algoritmov, napríklad z triedy Monte Carlo alebo z triedy znáhodnených optimalizačných metód, je závislých na rovnomernom generovaní bodov splňujúcich určité požiadavky.

Ako základ náhody bude náš generátor bodu v polyédre používať rovnomerný generátor čísel $[0, 1]$. Pomocou generátoru na $[0, 1]$ možno triviálne generovať bod na $[0, k]$ (prenásobením konštantou k), tiež možno generovať bod na $[a, b]$ (vygenerovaním bodu na $[0, -a + b]$ a pripočítaním konštanty a, alebo bod na $[0, 1]^n$ (postupným vygenerovaním súradníc). Generovanie na iných polyédroch, najmä v priestoroch vysokej dimenzie, je však vo všeobecnosti netriviálny problém.

Cieľom tejto práce je jednak poskytnúť prehľad známych metód, ktoré je možné použiť na rovnomerné generovanie v polyédroch a taktiež implementovať čo najefektívnejší generátor.

V prvej kapitole sa budeme zaoberať známymi metódami, ktoré možno použiť na generovanie na polyédroch. Medzi ne patria Metropolis–Hastings metódy (z triedy Markov Chain Monte Carlo), ktoré sa snažia simulovať komplexné rozdelenia výberom. To možno použiť aj v našom prípade, keď je cielené rozdelenie uniformné. V triede Metropolis–Hastings metód sa špecificky zameriame na Gibbsov generátor, ktorý je

možno jednoducho implementovať práve pre generovanie na mnohorozmerných polyédroch. Okrem toho sa budeme zaoberať aj zamietacími metódami, ktoré namiesto generovania bodov priamo v polyédri vygeneruju bod na jednoduchšej nadmnožine polyédra rovnomerne náhodne. Po vygenerovaní bodu overia, či leží v polyédre. Ak nie, tak generujú znovu.

Druhá kapitola je venovaná problému optimálneho návrhu experimentov (optimal design problem), pomocou ktorého predstavíme algoritmus Randomized Exchange Algorithm. Daný algoritmus vieme použiť aj na nájdnenie elipsoidu s minimálnym objemom obaľujúci zadaný polyéder. Tento elipsoid možno jednak priamo použiť ako nadmnožinu pri zamietacej metóde, no taktiež možno jeho hlavné osi využiť na zistenie natočenia polyédra a obalenie polyédra kvádrom s malým objemom.

Tretia kapitola bude obsahovať porovnania algoritmov spomenutých v prvej kapitole a zdrojový kód čo najrýchlejšieho algoritmu na generovanie bodov vnútri polyédru.

Táto kapitola sa bude meniť

Kapitola 1

Metódy generovania vnútri polyédru

V tejto kapitole sa budeme zaoberať známymi metódami na generovanie z určitého rozdelenia, ktorá je v našom prípade rovnomerná vnútri polyédra a nulová mimo polyédra. V prvej podkapitole sa budeme zaoberať triedou Metropolis–Hastings algoritmov, v druhej podkapitole sa budeme zaoberať zamietacími metódami.

1.1 Metropolis–Hastings metódy

V tejto sekcii si predstavíme triedu Metropolis–Hastings algoritmov na generovanie bodov z ľubovoľného rozdelenia. Na postupnosť bodov generovnaných algoritmami z triedy Metropolis–Hastings sa dá pozerať ako na postupnosť stavov markovovských reťazcov (Markov Chain Monte Carlo, ďalej MCMC). Daným spôsobom je ich možné analyzovať a dokázať, že generujú body podľa žiadaného rozdelenia (kvôli časovému obmedzeniu práce je táto časť vynechaná).

V nasledujúcej časti predstavíme všeobecný Metropolis–Hastings algoritmus, podrobnejšie vysvetlenie viete nájsť napríklad v [2]. V následných častiach opíšeme jeho konkrétne realizácie.

1.1.1 Všeobecný Metropolis–Hastings algoritmus

Majme cieľovú hustotu Q z ktorej chceme generovať, v prípade rovnomerného generovania vnútri polyédra je rovnomerná v polyédri a nulová mimo neho.

Metropolis–Hastings algoritmus [2] sa nachádza v stave $x^{(i)}$ reprezentovanom bodom v polyédri, stav určuje *kandidátsku hustotu* $Q(x^{(i)})$ závislú na $x^{(i)}$. Táto kandidátska hustota (proposal density) je volená tak, aby sa z nej bolo možné jednoducho generovať a môže byť značne odlišná od cieľovej hustoty Q.

Algoritmus postupuje iteratívne, v jednom kroku vygeneruje ďalší potenciálny stav y podľa hustoty $Q(x^{(i)})$. Ďalší stav algoritmu $x^{(i+1)}$ bude y s pravdepodobnosťou $\alpha(y|x^{(i)})$, inak to bude $x^{(i)}$.

Algorithm 1 Všeobecný Metropolis–Hastings algoritmus [2]

1: inicializuj $x^{(0)}$
2: **for** $i = 0, 1, \ldots, N$ **do**
3: Vygeneruj bod y z $Q(x^{(i)})$
4: Vygeneruj u z $U(0, 1)$.
5: **if** $u \leq \alpha(y|x^{(i)})$ **then**
6: Nastav $x^{(i+1)} = y$
7: **else**
8: Nastav $x^{(i+1)} = x^{(i)}$
9: Vráť $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$.

Môžeme si všimnúť, že v Metropolis–Hastings algoritme je bod $x^{(i)}$ závislý od predchádzajúceho bodu $x^{(i-1)}$. Podľa [2] je pri vhodnej voľbe kandidátskej hustoty $Q(x^{(i)})$ a pravdepodobnosti α možné dokázať, že napriek závislosti po sebe idúcich bodov je pre $N \to \infty$ limitné rozdelenie náhodneho vektora $x^{(N)}$ rovné Q. Potrebná veľkosť N na dosiahnutie dostatočne presného odhadu hustoty Q sa nazýva burn–in period.

V ďalších častiach si ukážeme niekoľko konkrétnych realizácii Metropolis–Hastings algoritmu. Každá z tých metód obsahuje určité predpoklady na distribúciu, z ktorej chceme generovať, no dá použiť aj na rovnomerné generovanie bodov v polyédri.

1.1.2 Hit–and–Run generátor

Ako jedna z možností na realizáciu Metropolis–Hastings algoritmu prichádza do úvahy Hit–and–Run generátor. Algoritmus je analogický s algoritmom Metropolis–Hasting, pričom hustota $Q(x^{(i)})$ je určená priamkou d_i s náhodným smerom cez bod $x^{(i)}$. Označme si S zadaný polyéder. Hustota $Q(x^{(i)})$ je rovnomerná na úsečke $S \cap d_i$ a nulová inde. Pri danej hustote je markovovský reťazec reverzibilný, t.j. $Q(x^{(i+1)}|x^{(i)}) = Q(x^{(i)}|x^{(i+1)})$ [1] **TODO overit citaciu** preto je funkcia α konštantne 1, t.j. algoritmus sa každým krokom pohne do iného bodu.

Hit–and–Run generátor funguje nasledovne:

Algorithm 2 Hit–and–Run generátor [1]

1: Inicializuj $x^{(0)}$
2: **for** $i = 0, \ldots, N - 1$ **do**
3: Vygeneruj smer d_i z distribúcie D na povrchu sféry
4: Nájdi množinu $S_i(d_i, x^{(i)}) = \{\lambda \in \mathbb{R}; x^{(i)} + \lambda d_i \in S\}$
5: Vygeneruj $\lambda_i \in S_i$ podľa hustoty $Q_i(\lambda|d_i, x^{(i)})$
6: Nastav $x^{(i+1)} = x^{(i)} + \lambda_i d_i$
7: Vráť $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$.

Použiteľnosť Hit–and–Run generátora závisí od toho, ako rýchlo vieme generovať smery d_i z rozdelenia D. Ak by dimenzia priestoru bola príliš veľká, nebolo by možné generovať rýchlo z rozdelenia D a preto celý algoritmus by bol pomalý.

1.1.3 Gibbsov generátor

V tejto podsekcii sa budeme zaoberať Gibbsovým generátorom, metódou generovania z triedy MCMC vhodnou na generovanie vo viacrozmernom priestore. Na Gibbsov generátor sa možno dívať ako na špeciálny prípad Metropolis–Hastings algoritmu.

Našou úlohou je generovať z n–rozmernej distribúcie Q, pričom z Q nevieme generovať priamo. Predpokladajme, že nevieme použiť priamo Hit–and–Run generátor, lebo $Q(x^{(i)}) = Q(x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})$ je príliš zložitá na generovanie. Taktiež predpokladajme, že ak $Q(x^{(i)})$ obmedzíme na jeden rozmer, tak v ňom vieme generovať rýchlo, tj. možno generovať rýchlo z $Q(x_j^{(i)}|x_1^{(i+1)}, x_2^{(i+1)} \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, x_{j+2}^{(i)} \ldots, x_n^{(i)})$.

Gibbsov generátor funguje nasledovne:

Algorithm 3 Gibbsov generátor [5]

1: inicializu $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$
2: **for** $i = 1, \ldots, N$ **do**
3: **for** $j = 0, 1, \ldots, n$ **do**
4: $x_j^{(i)} \sim Q(x_j|x_1^{(i+1)}, x_2^{(i+1)} \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, x_{j+2}^{(i)} \ldots, x_n^{(i)})$
5: $x^{(i+1)} = (x_1^{(i+1)}, x_2^{(i+1)} \ldots, x_n^{(i+1)})$
6: Vráť $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$.

Ako špeciálny prípad Metropolis–Hastings algoritmu má Gibbsov generátor podobné vlastnosti ako Metropolis–Hastings algoritmus. Jeho hlavnou výhodou je, že je jednoduchý a že neobsahuje žiadne parametre.

Gibbsov generátor možno použiť aj pri rovnomernom generovaní v polyédroch, vďaka linearite nerovníc (pri H–reprezentácii polyédra) sa môžeme ľahko obmedziť na jeden rozmer. Z vlastností polyédru plynie, že rozdelenia $x_j^{(i)} \sim Q(x_j|x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)}$ sú rovnomerné rozdelenia na úsečke, ktorej hraničné body vieme vypočítať veľmi rýchlo, čiže generovať z týchto rozdelení je obzvlášť jednoduché.

1.2 Zamietacie metódy

Zamietacie metódy nám poskytujú jeden zo spôsobov rovnomerného generovania bodov na určitej množine. Myšlienka za nimi je nasledovná: Označme si X množinu, na ktorej chceme rovnomerne náhodne generovať prvky. Predpokladajme, že nevieme

priamo rovnomerne generovať body na X (X je veľarozmerná alebo komplikovane zadaná), no vieme rovnomerne generovať na množine S, $X \subset S$.

Náš generátor G_S bude pracovať nasledovne:

Algorithm 4 Zamietacia metóda

1: **for** $i = 1, \ldots, N$ **do**
2: **repeat**Vygeneruj bod $x^{(i)} \in S$ rovnomerne náhodne
3: **until** $x^{(i)} \in X$
4: Vráť $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$

Generátor G_S vygeneruje bod $x \in S$ rovnomerne náhodne, ak je ten bod aj z X, tak ho vráti ako výstup, inak vygeneruje nový bod $x \in S$. Všimnime si, že generátor G_S je závislý iba od S a že generuje body na $X \cap S = X$ rovnomerne náhodne.

Očakávaná rýchlosť generovania závisí od toho, koľkokrát G_S vygeneruje bod mimo X. Z rovnomernosti G_S je tá pravdepodobnosť rovná $\frac{|S-X|}{|S|} = 1 - \frac{|X|}{|S|}$. Označme si p_k pravdepodobnosť, že G_S vygeneruje bod z X na k–ty pokus, t.j. najprv $k-1$ krát vygeneruje bod mimo X a potom vygeruje bod z X. Platí $p_k = (1-\frac{|X|}{|S|})^{k-1}\frac{|X|}{|S|}$. Očakávý počet generovaní G_S je $E(G_S) = \sum_0^\infty kp_k = \frac{|X|}{|S|}\sum_0^\infty k(1-\frac{|X|}{|S|})^{k-1} = \frac{|X|}{|S|}\frac{1}{((1-\frac{|X|}{|S|})-1)^2} = \frac{|S|}{|X|}$.

Táto metóda generovania je vhodná, ak je $\frac{|S|}{|X|}$ dostatočne malé, t.j. ak je obal S relatíve malý oproti polyédru X. Ak je $\frac{|S|}{|X|} \sim \infty$, tak je táto metóda nepoužiteľná.

Navyše, ak poznáme objem S, tak táto metóda nám ako vedľajší produkt poskytne aj štatistické intervaly spoľahlivosti pre objem X.

TODO rozpísať

1.2.1 Použitie na generovanie bodu vnútri polyédru

Zamyslime sa nad tým, ako by sme vedeli použiť túto metódu na generovanie bodu vnútri polyédru. Ako množinu možných S, $X \subset S$ môžeme použiť najmenší kváder so stranami rovnobežnými s osami. Vypočítať súradnece kvádra je ľahké, stačí nám to spraviť raz pred (začatím generovania) pomocou lineárneho programovania.

Žiaľ, pre takúto množinu S môže byť podiel $\frac{|S|}{|X|}$ byť ľubovoľne veľký. Ako príklad na takú množinu X uveďme kváder s obsahom k pozdĺž diagonály kocky $[0,1]^n$, dotýkajúci sa každej steny kocky $[0,1]^n$. Zrejme najmenšia množina S (kváder so stranami rovnobežnými s osami) obaľujúca X je kocka $[0,1]^n$, ktorá má obsah 1. Platí $\frac{|S|}{|X|} = \frac{1}{k}$. Keďže vieme nájsť kváder taký, že sa dotýka stien kocky $[0,1]^n$ a k je ľubovoľne malé, tak očakávaná dĺžka generovania touto metódou (pre danú množinu S) je ľubovovoľne veľká.

Ako ďalšia možná množina S prichádza do úvady elipsoid obaľujúci polyéder (označme si ho S_{MVEE}). Keďže chceme, aby bol podiel $\frac{|S_{MVEE}|}{|X|}$ čo najmenší, budeme skúmať elipsoid s najmenším obsahom obaľujúci polyéder — Minumum Volume Enclosing Elipsoid (ďalej MVEE). Môžeme si všimnúť, že MVEE elipsoid obsahuje veľa informácie o tom, ako vyzerá polyéder. Keďže elipsoid je jednotková guľa zobrazená lineárnou transformáciou, preto možno generovať body vnútri elipsoidu ako obrazy bodov vygenerovaných vnútri jednotkovej gule v lineárnej trasformácii. Nájsť daný elipsoid vieme pomocou REX algoritmu, ktorému je venovaná ďalšia kapitola.

Keďže najjednoduchšia množina S, v ktorej vieme generovať je kváder, uvažujme prípad, keď S je najmenší kváder obaľujúci MVEE elipsoid. Zrejme osi kvádra budú zhodné s osami MVEE elipsoidu. Označme si daný kváder S_K a metódu generovania založenú na S_K kvádrova metóda. Pre S_K možno spraviť odhad veľkosti $\frac{|S_K|}{|X|}$ **TODO**. Tiež pre neho platí, že je obrazom kocky $[0,1]^n$ v zobrazení, ktoré zobrazí jednotkovú guľu na MVEE elipsoid.

Avšak, keďže tento kváder S_K nie je rovnobežný s osami sústavy, ako prirodzený spôsob rovnomerného generovania vnútri tohoto kvádra možno použiť generovanie vnútri kvádra (resp. kocky) s osami rovnobežnými s osami sústavy a následne zobrazené lineárnou transformáciou.

Tento prístup je analogický prístupu s MVEE elipsoidom. Dokonca, lineárne zobrazenia pri S_K a S_{MVEE} sú rovnaké. Rozdiel je jedine v tom, že pri S_{MVEE} rovnomerne generujeme na N–rozmernej jednotkovej guli (kde N je dimenzia priestoru), čo následne zobrazujeme lineárnym zobrazením. Pri S_K rovnomerne generujeme na N–rozmernej jednotkovej kocke (obaľujúcej N–rozmernú jednotkovú guľu), čo zobrazujeme lineárnym zobrazením na kváder S_K obaľujúci MVEE elipsoid.

Ukážme si, že pri dostatočne rýchlom rovnomernom generovaní bodov v guli kvádrova metóda nie je rýchlejšia ako MVEE metóda. Porovnajme očakávaný čas týchto prístupov. Označme si t_z očakávaný čas výpočtu lineárneho zobrazenia, S_G N–rozmernú guľu a t_G očakávaný čas vygenerovania bodu v nej, S_K N–rozmernú kocku a t_K očakávané čas vygenerovanie bodu v nej. Nakoniec si označme S_P množinu bodov hľadaného polyédra.

Očakávaný čas na vygenerovanie bodu pomocou MVEE elipsoidu je $\frac{|S_G|}{|S_P|}t_z t_G = \frac{t_z}{|S_P|}t_G|S_G|$, očakávný čas na vygenerovanie pomocou obaľujúceho kvádra $\frac{|S_K|}{|S_P|}t_z t_K = \frac{t_z}{|S_P|}t_K|S_K|$. Podiel očakávaných časov je

$$\frac{\text{očakávaný čas kvádrovej metódy}}{\text{očakávaný čas MVEE metódy}} = \frac{\frac{|S_K|}{|S_P|}t_z t_K}{\frac{|S_G|}{|S_P|}t_z t_G} = \frac{|S_K|t_K}{|S_G|t_G}.$$

Ak použijeme metódu na rovnomerné generovanie bodov vnútri gule s očakávaným

časom $t_G \leq \frac{|S_K|t_K}{|S_G|}$, tak podiel očakávaných trvaní metód bude menší–rovný ako jedna, teda kvádrová metóda bude pomalšia.

Môžeme si všimnúť, že zamietacia metóda na generovanie bodov v S_G pomocou nadmnožiny S_K má očakávaný počet generovaní rovný $\frac{|S_K|t_K}{|S_G|}$, preto očakávané trvanie vygenerovania bodu je $\frac{|S_K|t_K}{|S_G|}$.

Týmto sme ukázali, že ak by sme pri generovali body v polyédri pomocou MVEE metódy, pričom rovnomerné generovanie bodov v S_G by sme realizovali pomocou zamietaciu metódu s nadmnožinou S_K, dostali by sme presne rýchlosť kvádrovej metódy. Kedže existujú aj rýchlejšie metódy rovnomerného generovania vnútri gule, kvádrová metóda je pomalšia ako MVEE metóda. Ďalej sa ňou nebudeme zaoberať.

1.3 Rovnomerné generovanie bodov v polyédri pomocou MVEE elipsoidu

V tejto podkapitole sa budeme bližšie zaoberať MVEE metódou, ako pomocou už vypočítaného MVEE elipsoidu možno rovnomerne generovať body vnútri zadaného polyédra. Budeme používať zamietaciu metódu, vygerujeme bod v elipsoide a overíme, či je daný bod tiež v polyédre.

Nakoľko každý elipsoid je jednotková guľa zobrazená lineárnym zobrazením, na rovnomerné generovanie v MVEE elipsoide najprv rovnomerne vygenerujeme bod v jednotkovej guli a následne ho zobrazíme daným lineárnym zobrazením do bodu v MVEE elipsoide. Rovnomernosť generovania sa zachová **TODO prečo**.

Na rovnomerné generovanie bodu x v d–rozmernej jednotkovej guli so stredom v nule najprv vygenerujeme d–rozmerný uhol rovnomerne náhodne a následne vygenerujeme vzdialenosť bodu x od nuly tak, aby mali oblasti rôzne vzdialené od 0 rovnakú pravdepodobnosť na vygenerovanie.

Na rovnomerné vygenerovanie d–rozmerného uhlu vygenerujeme bod v d–rozmernom centrálne symetrickom rozdelení (napríklad normálnom) a následne ho zobrazíme na jednotkovú sféru. Takto zrejme dostaneme rovnomerné rozdelenie na d–rozmernej sfére. Následne vygenerujeme vzdialenosť od nuly podľa rozdelenia daného funkciou $f : [0,1] \to [0,1]$:

$$f(x) = \frac{x^d}{\int_0^1 y^d dy}$$

. Pri danej funkcii má generovaný bod x rovnakú pravdepodobnosť, že padne do ľubovoľne vzdialenej oblasti s rovnakým obsahom **TODO prečo**, preto je dané rozdelenie

vnútri gule rovnomerné.

Kapitola 2

Metódy na riešenie problému optimálneho návrhu

Cieľom tejto kapitoly je predstaviť Randomized exchange algoritmus [3] (ďalej REX) ako metódu na riešenie problému optimálneho návrhu (optimal design problem).

Vďaka ekvivalencii problému D–optimálneho návrhu a minimum volume enclosing elipsoidu (MVEE) [3] možno REX využiť aj na riešenie MVEE problému. Ak body polyédra P vo V–reprezentácii stotožníme s lineárnymi regressormi v probléme D–optimálneho návrhu, z riešenia D–optimálneho návrhu pre dané regressory môženo vypočítať MVEE elipsoid prislúchajúci k polyédru P.

Pre polyéder v H–reprezentácii zadaný bodmi $\{z_1, \ldots, z_n\}$ ($z_i \in \mathbb{R}^m$ pre $i = 1, \ldots, n$) je možno MVEE vypočítať ako

$$P(H, Z) = \{z \in \mathbb{R}^m | (z - Z)' H (z - Z) \leq 1\},$$

kde $Z = \sum_{i=1}^{n} w_i^* z_i$, $H = \frac{1}{m} [\sum_{i=1}^{n} w_i^* (z_i - Z)(z_i - Z)']^{-1}$, pričom w^* je D–optimálny návrh pre regressory $f_i = (1, z_i')'$ pre $i = 1, \ldots, n$.

Vzhľadom na dôležitosť MVEE elipsoidu pre túto prácu a kvôli lepšiemu pochopeniu REX algoritmu sa najprv pozrieme na jednoduchšie metódy riešenia problému optimálneho návrhu.

2.1 Základné metódy na riešenie problému optimálneho návrhu

Najprv si predstavíme metódu Subspace Ascend Method (ďalej SAM) ako všeobecnú iteratívnu metódu na riešenie problému optimálneho návrhu a Vertex Exchange

Method (VEM) ako jej konkrétnu realizáciu. Následne sa pozrieme na REX algoritmus ako na špeciálny prípad SAM, ktorý kombinuje VEM metódu s pažravým prístupom.

Označme návrh na X ako n–rozmerný vektor w s nezápornými prvkami so súčtom 1. Komponent w_x vektoru w predstavuje počet pokusov v bode $x \in X$ **TODO vysvetlit**. Označme nosič návrhu w ako $supp(w) = \{x \in X | w_x > 0\}$. Množina všetkých návrhov tvorí pravdepodobnostný simplex v $\mathbb{R}^n$, označme ju Ξ (je kompaktná a konvexná). Označme si $M(w)$ informačnú maticu prislúchajúcu k návrhu w. Označme si $\Phi : S_+^m \to \mathbb{R} \cup \{-\infty\}$, kritérium optimality, buď D–optimalitu Φ_D alebo A–optimalitu Φ_A. Naším cieľom bude maximalizovať $\Phi(M(w))$, t.j. nájsť optimálny návrh

$$w^* = \arg\max_{w \in \Xi} \in \{\Phi(M(w))\}$$

.

2.1.1 Subspace Ascend Method

SAM algoritmus postupuje iteratívne. V každej iterácii si vyberie podpriestor v ktorom sa bude hýbať a následne spraví optimálny krok v danom podpriestore:

Algorithm 5 Subspace Ascend Method (SAM) [3]

1: Zvoľ regulárny n rozmený návrh $w^{(0)}$
2: **while** $w^{(k)}$ nespĺňa podmienky zastavenia **do**
3: Zvoľ podmnožinu bodov $S_k \subset X$
4: Nájdi aktívny podpriestor Ξ ako $\Xi_k \leftarrow \{w \in \Xi | w_x = w_x^{(k)}, x \notin S_k\}$
5: Vypočítaj $w^{(k+1)}$ ako riešenie $\max_{w \in \Xi_k} \Phi(M(w))$ spĺňajúce $\Phi(M(w^{(k+1)})) \geq \Phi(M(w^{(k)}))$
6: Nastav $k \leftarrow k + 1$
7: Vráť w

SAM algoritmus každým krokom nezmenší funkciu Φ, teda sa hýbe smerom k optimu.

2.1.2 Vertex Exchange Method

Algoritmus VEM postupuje taktiež iteratívne. V kroku z návrhu w nájde návrh k minimalizujúci nosič w, návrh $l$minimalizujúci X. Ako ďalší návrh w' zvolí návrh z úsečky kl maximalizujúci $\Phi(M(w'))$. **TODO čo je X?**

Algorithm 6 Vertex Exchange Method (VEM) [3]

1: Zvoľ regulárny n rozmerný návrh w
2: **while** w nespĺňa podmienky zastavenia **do**
3: Vypočítaj $k \leftarrow \arg\min_{u \in supp(w)}\{d_u(w)\}$
4: Vypočítaj $l \leftarrow \arg\max_{v \in X}\{d_v(w)\}$
5: Vypočítaj $\alpha^* \leftarrow \arg\max_{\alpha \in [-w_l, w_k]}\{\Phi_D(M(w + \alpha e_l - \alpha e_k))\}$
6: Nastav $w_k \leftarrow w_k - \alpha^*$
7: Nastav $w_l \leftarrow w_l + \alpha^*$
8: Vráť w

Krok VEM algoritmu sa označuje ako leading Bohning exchange (ďalej LBE). Dvojica (k, l) sa označuje ako pár LBE.

2.2 Radomized Exchange Algoritmus

V tejto podkapitole popíšeme randomized exchange algoritmus (REX) predstavený v [3], dá sa na neho pozerať ako na špeciálny prípad SAM algoritmu [3]. REX algoritmus kombinuje kroky VEM algoritmu a pažravých algoritmov. Podľa [3] je REX algoritmus v praxi rýchlejší ako všetky state–of–the–art algoritmy na riešenie problému optimálneho riadenia. Na uvedenie predstavy o rýchlosti REX algoritmu, o viacerých (empiricky pomalších) algoritmoch na riešenie problému optimálneho riadenia bolo dokázané, že konvergujú v lineárnom čase. T.j. do vzialenosti ϵ od optima dôjdu v čase $\mathcal{O}(log(\frac{1}{\epsilon}))$ (napr. Khachiyanov algoritmus **TODO citovat**)

Nech w je regulárny návrh, nech $g(w)$ je n–rozmerný vektor s komponentami $g_x(w)$. Hlavná myšlienka REX algoritmu je počnúc inicializovaným regulárnym návrhom w iteratívne vyberať niekoľko návrhov (ich počet sa bude líšiť v rámci iterácii) a náhodne vykonať optimálnu výmenu váh medzi vybranými bodmi. Optimálna výmena váh je analogická LBE kroku VEM algoritmu. Voľba návrhov závisí na $g(w)$. Algoritmus bude postupovať nasledovne:

- **Krok LBE.** Pri danom návrhu w, vypočítaj $g(w)$ a urob LBE krok daný nasledovne:

$$\alpha^* \leftarrow \underset{\alpha \in [-w_l, w_k]\}}{\arg\max} \{\Phi_D(M(w + \alpha e_l - \alpha e_k)),$$

kde $k \in \arg\min_{u \in supp(w)}\{d_u(w)\}, l \in \arg\max_{v \in X}\{d_v(w)\}$. Optimálny krok $\alpha^*_{k,l}(w)$ nazvime *nulujúci*, ak je rovný buď $-w_l$ alebo w_k. To zodpovedá prípadu, keď sme sa optimálnym krokom pohli do niektorého z návrh w_l alebo w_k.

- **Výber aktívneho podpriestoru.** Podpriestor $S \subset X$, v ktorom sa pohneme bude zvolený ako zjednotenie dvoch množín. Jednou vybranou pažravým procesom (S_{greedy}) a druhou ako nosič návrhu w ($S_{support}$).

  - **Pažravá množina.** Nech $L = \min(\gamma m, n)$ je počet návrhov, ktoré vyberieme. Potom zvoľ S_{greedy} ako

    $$S_{greedy} = \{l_1^*, \ldots, l_L^*\} \subset X,$$

    kde l_i^* je najväčšia zložka vektoru $g(w)$.

  - **Nosič.** Nastav

    $$S_{support}(w) = supp(w).$$

    Označme K veľkosť nosnej množiny $K = |supp(w)|$.

  - **Aktívny podpriestor.** Aktívny podpriestor S je definovaný ako

    $$S = S_{greedy} \cup S_{support}.$$

    Váhy návrhov mimo aktívneho podpriestoru nebudú upravované v tejto iterácii.

- **Krok v aktívnom podpriestore.** Teraz vykonáme krok, v ktorom aktualizujeme hodnoty w_v pre $v \in S$. Návrhy w_v pre $v \notin S$ ostanú nezmenené.

  - **Tvorba párov.** Nech $(k_1, \ldots, k_K)$ je uniformne náhodná premutácia $S_{support}$ a nech $(l_1, \ldots, l_L)$ je uniformne náhodná permutácia S_{greedy}. Potom postupnosť aktívnych návrhov je

    $$(k_1, l_1), (k_2, l_1), \ldots, (k_1, l_L), (k_2, l_L), \ldots, (k_K, l_L)$$

    .

  - **Aktualizácia.** Vykonaj postupne všetky Φ–optimálne LBE kroky medzi návrhmi z $(k_1, l_1), \ldots, (k_K, l_L)$ s prisluchajúcimi aktualizáciami w a $M(w)$.

Algorithm 7 REX algoritmus [3]

Zvoľ regulárny n–rozmerný návrh w
while w nespĺňa podmienky zastavenia **do**
    Urob LBE krok vo w
    Nech k je vektor zodpovedajúci náhodnej permutácii prvkov $supp(w)$
    Nech l je vektor zodpovedajúci náhodnej permutácii $L = \min(\gamma m, n)$ indexov prvkov $g(w)$
    for $l = 1 \dots L$ **do**
        for $l = 1 \dots K$ **do**
            $\alpha^* \leftarrow \arg\max_{\alpha \in [-w_l, w_k]} \{\Phi_D(M(w + \alpha e_l - \alpha e_k))\}$
            if LBE krok bol nulujúci alebo $\alpha^* = -w_l$ alebo $\alpha^* = w_k$ **then**
                $w_k \leftarrow w_k - \alpha^*$
                $w_l \leftarrow w_l + \alpha^*$
Vráť w

Kapitola 3

Porovnanie generátorov

Táto kapitola bude obsahovať praktické porovnania algoritmov spomenutých v prvej kapitole a zdrojový kód čo najrýchlejšieho algoritmu na generovanie bodov vnútri polyédru.

Predpokladáme, že najrýchlejší generátor bude využívať MVEE elipsoid, teda táto kapitola bude obsahovať taktiež implementáciu REX algoritmu.

3.0.1 Generovanie polyédrov

Metódy boli porovnávané na veľarozmerných polyédroch. Ako vhodná množina polyédrov boli zvolené náhodne generované polyédre. Týmto sa vyhneme degenerovaným polyédrom.

Na generovanie bodov pomocou Metropolis–Hastings metód potrebujeme mať polyéder zadaný v H–reprezentácii, no REX algoritmus na nájdenie MVEE elipsoidu potrebuje ako vstup polyéder vo V–reprezentácii. Preto je nutné v rámci generovania vygenerovať polyéder zároveň v H–reprezentácii aj vo V–reprezentácii. Teda potrebujeme vygenerovať polyéder v jedenej reprezentácii a previesť ho do druhej. Daný prevod bude pre každý polyéder spravený len raz, preto nie je nutné, aby bol rýchly.

Kedže topologickú štruktúru polyédra možno zapísať ako planárny graf, možno jednoducho ukázať, že minimálna množina stien (nerovností) v H–reprezentácii a minimálna množina vrcholov vo V–reprezentácii sú asymptoticky rovnako veľké (až na lineárny faktor). Presnejšie, medzi počtom stien F a počtom vrcholov V v polyédri platí vzťah $F + V = E - 2$ (kde E je počet hrán polyédra, ktorý možno odhadnúť $E \leq 3V - 6$ (pre $V > 2$). Tým pádom by v rámci porovnania metód malo byť pri rozumných transformáciach viac–menej jedno, či najprv náhodne vygenerujeme polyéder v H–reprezentácii, ktorú prevedieme do V–reprezentácie alebo či najprv vygenerujeme V–reprezentáciu, ktorú prevedieme do H–reprezentácie. V obidvoch prípadoch dostaneme asymptoticky rovnako veľké reprezentácie.

Ak máme danú aj stenovú aj vrcholovú reprezentáciu polyédra, ktoré nie sú nutne minimálne (vzhľadom na počet vrcholov vo V–reprezentácii a počet stien v H–reprezentácii), môžeme jednoducho dané reprezentácie minimalizovať odstránením prebytočných informácii. Z H–reprezentácie možno odstrániť tie nadroviny, ktoré neprechádzajú aspoň tromi bodmi z V–reprezentácie. Z V–reprezentácie možno odstrániť tie body, ktoré neležia na aspoň troch nadrovinách z H–reprezentácie. Takto získané reprezentácie sú zrejme minimálne. **TODO Isto? Ak nie, tak s pravdepodobnostou 1 pri rozumnom provede** Overiť, či bod x leží na nadrovine danej vektorom a a konštantou c je triviálne, stačí overiť rovnosť $a^T x = c$.

Ako alternatíva ku generovaniu polyédrov v jednej reprezentácii a prevádzaniu do druhej reprezentácie možno porovnávať metódy aj inak. Predpokladajme, že máme generátor G_H polyédrov v H–reprezentácii a generátor G_V polyédrov vo V–reprezentácii, pričom G_H a G_V generujú z rovnakého rozdelenia polyédrov. Ak by sme testovali Metropolis–Hastings metódy na veľkom množstve polyédrov vygenerovaných pomocou G_H a metódy využívajúce REX na veľkom množstve polyédrov vygenerovaných pomocou G_V, výsledky budú podobné ako keby sme spomínané metódy testovali na rovnakých polyédroch. Nakoľko nájdenie generátorov G_H a G_V s rovnakým rozdeleným polyédrov je netriviálny problém (už len zabezpečenie rovnakého rozdelenia obsahov polyédrov je netriviálne), tomuto prístupu sa z dôvodu obmedzenému časovému rámcu práce venovať nebudeme.

Generovanie polyédru vo H–reprezentácii

V rámci tejto práce je použitý algoritmus na generovanie náhodných polyédrov popísaný v [**?**]. Výstupom algoritmu je polyéder v H–reprezentácii, taký, že každý bod má rovnakú pravdepodobnosť byť vnútri.

Algoritmus využíva prístup Monte Carlo, funguje nasledovne: Najprv náhodne zvolí m nadrovín $p_1, \ldots, p_m$, tie rozdeľujú priestor na niekoľko nie nutne ohraničených oblastí. Následne rovnomerne náhodne vygeneruje bod c v priestore, ako polyéder P_c zvolí oblasť vymedzenú priamkami p_i, v ktorej leží c. Na záver overí, či je vygenerovaný polyéder P_c ohraničený vopred zvolenou hyperkockou. Ak áno, tak vráti P_c. Ak nie je, tak daný polyéder zahodí a generuje znovu.

Algorithm 8 Generátor náhodných polyédrov [?]

1: Náhodne vyber bez návratu n z $m + 2n$ indexov obmedzení $i_1, i_2, \ldots, i_n$
2: Nastav $B = [p^{i_1}, p^{i_2}, \ldots, p^{i_n}]^T$, zrejme B^{-1} existuje s pravdepodobnosťou 1
3: Nastav $V = B^{-1}[||p^{i_1}||^2, \ldots, ||p^{i_n}||^2]^T$
4: Náhodne zvoľ $c \in \mathbb{R}^n$
5: Nastav $y = c^T B^{-1}$, zvoľ nerovnosti P_c nasledovne:
6: **for** $i = 0, 1, \ldots, n$ **do**
7: **if** $y_i > 0$ **then**
8: Nastav i–tu nerovnosť na $\geq$
9: **else**
10: Nastav i–tu nerovnosť na $\leq$
11: **if** P_c nie je celý v hyperkocke **then**
12: Zamietni polyéder P_c, vráť sa na 1
13: **else**
14: Odstráň obmedzenia hyperkocky, vráť P_c

Pri takomto generovaní v rámci P_c získame nadroviny p_i, ktoré neobsahujú žiadnu stenu polyédra. Tieto nadroviny sú zrejme nadbytočné, preto ich môžeme odtiaľ odstrániť. Na nájdenie, ktoré nadroviny sú prebytočné možno použiť lineárne programovanie. **TODO v prípade potreby rozšír**

Prevod polyédru z H–reprezentácie do V–reprentácie

TODO vymysliet/najst

Prevod polyédru z V–reprezentácie do H–reprentácie

Majme množinu V bodov V–reprezentácie polyédru. Pre každú trojicu bodov z V sa pozrime na rovinu r prechádzajúcu nimi. Daná rovina je stenou polyédru, ak všetky body z V ležia v rovnakom podpriestore vymedzenom r.

Takto zrejme vieme dostať všetky steny polyédru v čase $\mathcal{O}(|V|^4)$.

Ajajaj, toto funguje len v 3D

TODO vymysliet/najst

Záver

V tejto kapitole budú zhrnuté výsledky práce. T.j. prehľad použiteľných metód spolu s ich výsledkom praktického porovnania. Okrem toho tu bude spomenutý (vlastný) prínos, čo nové daná práca prináša. Taktiež tu budú spomenuté možné smery, ktorými možno rozšíriť prácu (keďže práca má obmedzený časový rámec).

Táto kapitola je nedokončená.

Literatúra

[1] Ming-Hui Chen and Bruce W. Schmeiser. General Hit-and-Run Monte Carlo sampling for evaluating multidimensional integrals. *Operations Research Letters*, 19(4):161–169, October 1996.

[2] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4):327–335, November 1995.

[3] Radoslav Harman, Lenka Filová, and Peter Richtárik. A Randomized Exchange Algorithm for Computing Optimal Approximate Designs of Experiments. *arXiv:1801.05661 [stat]*, January 2018. arXiv: 1801.05661.

[4] Radoslav Harman and Vladimír Lacko. On decompositional algorithms for uniform sampling from n-spheres and n-balls. *Journal of Multivariate Analysis*, 101(10):2297–2304, November 2010.

[5] D. J. C. Mackay. Introduction to Monte Carlo Methods. In Michael I. Jordan, editor, *Learning in Graphical Models*, NATO ASI Series, pages 175–204. Springer Netherlands, Dordrecht, 1998.

[6] Gareth O. Roberts and Jeffrey S. Rosenthal. Convergence of Slice Sampler Markov Chains. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):643–660, January 1999.