UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Finančna matematika – 1. stopnja

Enja Erker, Jakob Svetina

Aproksimacija povprečne razdalje med točkami v dvodimenzionalnem evklidskem prostoru

Poročilo o projektu pri predmetu Finančni praktikum

Mentorja: prof. dr. Sergio Cabello, asist. dr. Janoš Vidali

Ljubljana, 10. 1. 2021

Kazalo

Aproksimacija povprečne razdalje med točkami v dvodimenzionalnem evklidskem prostoru

Povzetek

Osrednji proučevani problem projekta pri predmetu Finančni praktikum je aproksimacija povprečne razdalje med točkami v dvodimenzionalnem evklidskem prostoru. Teoretičen del naloge temelji na konstrukciji algoritmov. V programu R sva napisala algoritem za omenjeno aproksimacijo, poleg tega sva napisala še časovno bolj zahteven algoritem, ki računa točno povprečno razdaljo med točkami. Nato sva v programu R izvedla več poskusov, s katerimi sva preverila natančnost aproksimacije. Ključni ugotovitvi eksperimentalnega dela sta dve. Prva, da se z večanjem števila točk zmanjšuje število potrebnih ponovitev izvedbe algoritma in druga, da se z večanjem natančnosti povečuje število izvedenih ponovitev. Glede na rezultate sva zaključila, da je skonstruirani algoritem časovno dokaj učinkovit.

Ključne besede: povprečna razdalja med točkami, implementacija objekta, naključen enotski vektor, naključna projekcija

1. Uvod

Osrednji proučevani problem projekta pri predmetu Finančni praktikum je aproksimacija povprečne razdalje med točkami v dvodimenzionalnem evklidskem prostoru. Namen naloge je čim bolj ekonomično, s čim manjšo časovno zahtevnostjo, oceniti približek povprečne razdalje med vsemi točkami dane množice v ravnini. Najpreprostejši pristop k problemu je postopno seštevanje vseh možnih razdalj med danimi točkami, vendar je ta metoda časovno zelo potratna.

Z namenom poiskati metodo, katere časovna zahtevnost bi bila manjša od navedene, je bilo v preteklosti objavljenih kar nekaj študij. Ključno literaturo, na kateri temelji teoretično ozadje najine naloge, sestavljata dve deli. Gre za članek avtorjev K. Barhum, O. Goldreich in A. Shraibman z naslovom 'On Approximating the Average Distance Between Points' [1] in diplomsko delo avtorja S. Kolar Celarc 'Aproksimacijski algoritmi za računanje povprečne razdalje med točkami' [2]. Pristop, ki ga bova predstavila in implementirala v tem projektu je t. i. 'implementacija objekta' [1] oz. 'naključna projekcija' [2].

2. Opis problema

Izhodišče problema aproksimacije povprečne razdalje med točkami v dvodimenzionalnem evklidskem prostoru je ravnina ($\mathbb{R}^2$), v kateri leži množica n točk. V nadaljevanju bo ta množica označena s S, velja $S = \{P_1, P_2, ...P_n\} \subset \mathbb{R}^2$. Povprečna razdalja med točkami v S je definirana kot [2]

$$D(S) = \frac{1}{n^2} \sum_{(P,Q)\in S\times S} d(P,Q),$$

kjer je $d(P,Q)$ razdalja med točkama P in Q.

Izbran pristop 'implementacije objekta' oz. 'naključne projekcije' je osnovan na konceptu prevedbe problema iz dvodimenzionalnega na enodimenzionalnega. V primeru ene dimenzije je namreč mogoče problem rešiti zgolj v času $O(n \cdot log(n))$. Problem v dveh dimenzijah prevedemo na eno dimenzijo, tako da točke iz ravnine projiciramo na naključno premico z naključnim smernim vektorjem, ki gre skozi izhodišče. Nato seštejemo ustrezne dolžine med projekcijami točk, ki so sedaj kolinearne. Pričakovana vrednost povprečne razdalje med točkami, ki jih projiciramo na naključno premico, se razlikuje od dejanske povprečne razdalje med točkami za konstantni faktor in je neodvisna od izbire premice. Izkaže se, da lahko na premici vsoto vseh razdalj med točkami izračunamo bistveno hitreje kot v splošnem prostoru [2].

V projektu bova tako zasledovala dvojni cilj. Prvič, zasnovati natančen algoritem, ki oceni približek povprečne razdalje med danimi točkami v ravnini v čim krajšem času (skoraj linearnem času števila točk). Drugič, odgovoriti na ključno vprašanje glede napake in sicer, koliko eksperimentalnih ponovitev je treba narediti, da bo napaka čim manjša (npr. 5 %, 2 %).

3. Programsko okolje in implementacija

Za implementacijo predstavljenega problema sva uporabila program R. Ta program sva izbrala, ker je od nama znanih programov oz. programskih jezikov najbolj primeren za izvedbo naloge. Najprej sva napisala dva krajša oz. pomožna algoritma predstavljena v nadaljevanju (Algoritma 1 in 2), ki sva ju nato združila v osrednji

algoritem, ki izračuna približek povprečne razdalje med točkami v ravnini (Algoritem 3). V zaključku tega poglavja je predstavljen tudi postopek za izračun točne povprečne razdalje med točkami (Algoritem 4), ki služi za primerjavo. Vsi algoritmi v nadaljevanju so predstavljeni s psevdokodo.

Prvi izmed skonstruiranih algoritmov je funkcija F (Algoritem 1), ki ima za vhodni podatek dimenzijo prostora d, v katerem se nahajajo točke. Vrne pa pričakovano vrednost vsote absolutnih vrednosti produktov vektorja v z vektorji iz naključne ortonormirane baze $\mathbb{R}^d$. V specifičnem primeru te naloge, je dimenzija prostora vedno enaka 2. Vrednost $F(d)$ sva zato izračunala le enkrat. Časovna zahtevnost tega algoritma je v splošnem enaka $O(d)$ oz. $O(2)$.

Algoritem 1 Funkcija F

Vhod: d
Izhod: pričakovana vrednost vsote abosolutnih vrednosti produktov vektorja v z vektorji iz naključne ortonormirane baze $\mathbb{R}^d$

1: **function** F(d)
2: $\frac{(d-2)!!}{(d-1)!!} \cdot \sqrt{\frac{2}{\pi}}$
3: **end function**

Postopek za pridobivanje naključnih enakomerno porazdeljenih enotskih vektorjev (Algoritem 2) brez vhodnih podatkov oz. argumentov zgenerira dve naključni neodvisni slučajni spremenljivki U in V z enakomerno porazdelitvijo $U[0, 1]$. Ti slučajni spremenljivki nato z Box-Mullerjevo metodo pretvori v standardizirano normalni slučajni spremenljivki [2]. Izhodni podatek je tako naključen enotski vektor.

Algoritem 2 Generiranje naključnega vektorja

Vhod: –
Izhod: naključen enotski vektor

1: **function** NaključenVektor( )
2: $U \leftarrow runif(1)$
3: $V \leftarrow runif(1)$
4: $M \leftarrow \sqrt{-2log(U)}cos(2\pi V)$
5: $N \leftarrow \sqrt{-2log(U)}sin(2\pi V)$
6: **return** $\frac{c(M,N)}{norm(c(M,N),type="2")}$
7: **end function**

Osrednji algoritem za izračun približka povprečne razdalje med točkami (Algoritem 3) ima dva vhodna podatka: seznam točk S in zahtevano natačnost ε. Prvi del algoritma predstavlja definiranje posameznih vrednosti, ki jih uporabimo v nadaljevanju. For zanka, ki se ponovi tolikokrat, da je natačnost ε ustezna, je osrednji del algoritma. Prvi korak znotraj zanke zgenerira naključen vektor z uporabo pomožne funkcije (Algoritem 2). Naslednji korak predstavlja izračun skalarnih produktov med seznamom točk S in zgeneriranim vektorjem. Skalarne produkte agloritem sortira, funkcija Sort je definirana kot običajno urejanje seznama. Sortiranje je časovno najzahtevnejši del tega algoritma, saj traja $O(n \cdot log(n))$. V kolikor je dimenzija d dovolj velika v primerjavi z n, prevlada računanje skalarnih produktov $\langle r, S[i]\rangle$, ki traja $O(dn)$ oz. $O(2n)$ [2]. Sortiranju sledi računanje razdalj. Pri vsaki

ponovitvi 'for' zanke se vrednosti *skupna_razdalja* prišteje vsota definirana v deseti vrstici algoritma. Izhodni podatek predstavljenega algoritma je približek povprečne razdalje med pari točk v S, definiran kot kvocient vrednosti $2 \cdot$*skupna_razdalja* in produkta fn^2m [2]. Časovna zahtevnost osrednjega algoritma znaša $O(\frac{(d+log(n))n}{\varepsilon^2})$ oz. $O(\frac{(2+log(n))n}{\varepsilon^2})$.

Algoritem 3 Izračun približka povprečne razdalje med točkami

Vhod: seznam točk S, zahtevana natančnost ε
Izhod: približek povprečne razdalje med pari točk v S

1: **function** PribližekPovprečneRazdalje(S,ε)
2: $n \leftarrow$ *nrow(S)*
3: $d \leftarrow$ *ncol(S)*
4: $f \leftarrow \mathrm{F}(d)$
5: $m \leftarrow \left\lceil \frac{1}{\varepsilon^2} \right\rceil$
6: *skupna_razdalja* $\leftarrow 0$
7: **for** $i \leftarrow 1$ **to** m
8: $r \leftarrow$ NaključenVektor
9: $S_r \leftarrow \mathrm{Sort}\langle S, r \rangle$
10: *skupna_razdalja* $\leftarrow$ *skupna_razdalja* $+ \sum_{n=1}^{dim(S_r)} S_r \cdot seq(1-n, n-1, 2)$
11: **end for**
12: **return** $\frac{2 \cdot skupna\_razdalja}{fn^2m}$
13: **end function**

Napisala sva tudi časovno bolj zahteven algoritem, ki izračuna točno povprečno razdaljo med točkami (Algoritem 4). Vhodni podatek predstavlja tabela točk oz. matrika dimenzije $n \times 2$. Algoritem za vsaki dve točki po standardni formuli izračuna razdaljo med točkama v ravnini ob upoštevanju simetričnosti ($d(P,Q) = d(Q,P)$), dobljene vrednosti sproti sešteva. Predstavljena direktna metoda ima časovno zahtevnost $O(n^2)$.

Algoritem 4 Izračun točne povprečne razdalje med točkami

Vhod: seznam točk S dimenzije $n \times 2$
Izhod: povprečna razdalja med pari točk v S

1: **function** TočnaPovprečnaRazdalja(S)
2: $s \leftarrow 0$
3: $n \leftarrow$ *nrow(S)*
4: **for** $i \leftarrow 1$ **to** $n-1$
5: **for** $j \leftarrow i+1$ **to** n
6: $s \leftarrow s + \sqrt{sum((S[i,] + S[j,])^2)}$
7: **end for**
8: **end for**
9: *povprečna_razdalja* $\leftarrow \frac{2s}{n^2}$
10: **return** *povprečna_razdalja*
11: **end function**

4. Generiranje podatkov

Generiranja podatkov (Algoritem 5) sva se lotila tako, da sva najprej naključno število točk izbrala sama, to označuje a. Algoritem spodaj ustvari matriko primerne velikosti glede na število točk. Koordinati vsake točke sva določila naključno, pri čemer sva generirala $2 \cdot a$ naključnih števil iz intervala $[b, c]$, ki sva jih nato uredila v a parov. Vsak par tako predstavlja točko. Predstavljeni algoritem glede na izbrano število točk naključno zgenerira le-te v omejeni ravnini.

Algoritem 5 Generiranje naključnih točk

Vhod: želeno število naključnih točk a, meji osi v ravnini b in c
Izhod: seznam točk S dimenzije $n \times 2$

function NaključneTočke(a, b, c)
    $S \leftarrow matrix\,(runif\,(2 \cdot a, b, c), nrow = a, ncol = 2)$
    return S
end function

5. Eksperimenti

Da bi odgovorila na naslednji dve raziskovalni vprašanji:

- kako se spreminja število korakov ob spremembi števila točk in
- kako se spreminja število korakov ob spremembi natančnosti,

sva v drugem delu naloge izvedla dva eksperimenta. Opazujemo število korakov osrednjega algoritma (Algoritem 3). Za prvi preizkus sva za število točk izbrala vrednosti 5, 15, 50 in 100 točk. Za drugi preizkus pa sva izbrala natančnosti 5 %, 3 %, 2 % ter 1 %.

Algoritmom, ki sva jih skonstruirala v poglavjih 3 in 4 sva dodala še dodatni funkciji predstavljeni v nadaljevanju, s katerima sva preverila natančnost aproksimacije. S prvo funkcijo (Algoritem 6) ugotovimo, koliko ponovitev je potrebnih za želeno natančnost. Pri tem zanka 'while' upošteva želeno natančnost in zaustavitveni pogoj. Zaustavitveni pogoj predstavlja omejenost števca z namenom, da ne bi prišlo do 'ponavljanja v neskončnost' oz. t. i. 'ciklanja'.

Algoritem 6 Test natančnosti

Vhod: seznam točk S dimenzije $n \times 2$, napaka ε
Izhod: število ponovitev

function TestNatančnosti(S, ε)
    točna $\leftarrow$ *as.numeric*(TočnaPovprečnaRazdalja(S))
    približek $\leftarrow 0$
    približek2 $\leftarrow 0$
    števec $\leftarrow 0$
    while $|\frac{približek}{točna} - 1| >$ *napaka* **and** *števec* < 50
        števec $\leftarrow$ *števec* $+1$
        približek2 $\leftarrow$ *približek2* $+$ PribližekPovprečneRazdalje(S)
        približek $\leftarrow \frac{približek2}{števec}$
    end while
    return *števec*
end function

Druga dodatna funkcija (Algoritem 7) vrne vektor s številom ponovitev postopka v d različnih ponovitvah testa natančnosti.

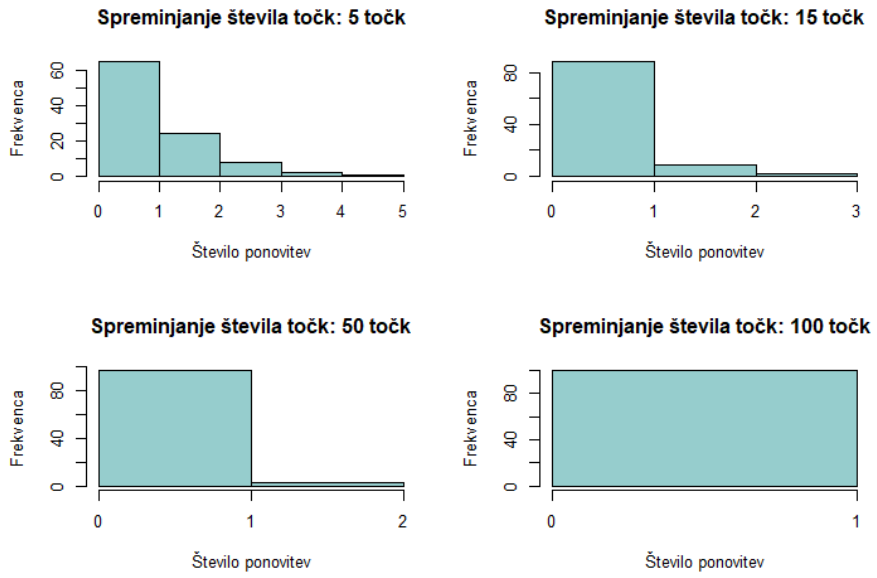
Algoritem 7 Test števila potrebnih ponovitev

Vhod: seznam točk S dimenzije $n \times 2$, napaka ε, dolžina d
Izhod: vektor števila ponovitev v d ponovitvah testa natančnosti

```
1: function TestŠtevilaPotrebnihPonovitev(S, ε, d)
2:     števci ← c( )
3:     for i ← 1 to d
4:         števci[i] ← TestNatančnosti(S,ε)
5:     end for
6:     return števci
7: end function
```

6. Rezultati

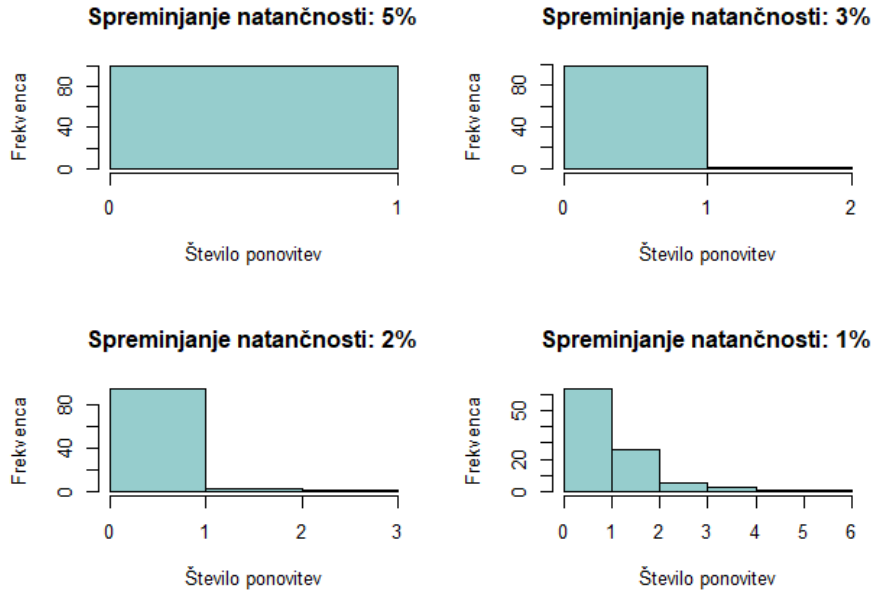
Ključna ugotovitev prvega izvedenega eksperimenta je, da se z večanjem števila točk zmanjšuje število potrebnih korakov izvedbe osrednjega algoritma za izračun približka povprečne razdalje med točkami. Dobljeni rezultat je pričakovan, saj je v primeru majhnega števila točk napaka večja in se mora postopek zato izvesti večkrat, da doseže želeno natančnost. Natančnost, s katero sva izvedla spodaj predstavljene eksperimente (Spreminjanje števila točk), je bila 2 %. Vizualizacija rezultatov projektnega dela je prav tako kot algoritimi izvedena v programu R.



Grafi 1: Spreminjanje števila točk.

Rezultat drugega izvedenega eksperimenta je, da se z večjo natančnostjo povečuje število potrebnih korakov izvedbe osrednjega algoritma za izračun približka povprečne razdalje med točkami. Dobljeni rezultat je podobno kot prvi pričakovan, saj je v primeru nižje natančnosti dovolj manj ponovitev postopka. V kolikor pa je

želena natančnost visoka, je število ponovitev temu primerno večje. Za ta eksperiment sva število točk nastavila na 10. Rezultati so spodaj predstavljeni še grafično (Spreminjanje natačnosti).



Grafi 2: Spreminjanje natančnosti.

7. Zaključek

V nalogi sva proučevala problem aproksimacije povprečne razdalje med točkami v dvodimenzionalnem evklidskem prostoru. Namen naloge je bil čim bolj ekonomično, s čim manjšo časovno zahtevnostjo, oceniti približek povprečne razdalje med vsemi točkami dane množice v ravnini. Teoretični del naloge zaobjema več algoritmov s poudarkom na osrednjem algoritmu, ki aproksimira povprečno razdaljo med točkami v ravnini (Algoritem 3) s časovno zahtevnostjo $O(\frac{(2+log(n))n}{\varepsilon^2})$. Praktičen del naloge pa temelji na dveh izvedenih eksperimentih. Ključni ugotovitvi eksperimentalnega dela sta dve. Prva, da se z večanjem števila točk zmanjšuje število potrebnih korakov izvedbe algoritma in druga, da se z večanjem natančnosti povečuje število izvedenih korakov. Glede na rezultate lahko zaključiva, da je v splošnem napaka algoritma relativno majhna.

Literatura

[1] K. Barhum, O. Goldreich in A. Shraibman *On Approximating the Average Distance Between Points*, Lecture Notes in Computer Science (2007) 296–310.

[2] S. Kolar Celarc, *Aproksimacijski algoritmi za računanje povprečne razdalje med točkami*, Delo diplomskega seminarja (2019) 4–17.