Intervalový odhad strednej hodnoty

Nech X má rozdelenie N(m, 0.3^2). Realizujme náhodný výber v rozsahu n = 30:

```
>> x = rand(1,30)*25;
```

Bodový odhad strednej hodnoty m je

```
>> xm = mean(x)

xm =    12.7693217282068
```

Hľadajme intervalový odhad strednej hodnoty so spoľahlivosťou 0.8, teda 80%. Na to potrebujeme zistiť číslo u, o ktorom pri normovanom normálnom rozdelení N(0,1) platí:

```
quad(f, -u, u) = 0.8  ,
```

kde f je funkcia hustoty rozdelenia N(0,1) daná predpisom

```
f=inline('exp(-x.^2/2)/sqrt(2*pi)');
```

Ideálne by bolo použiť na výpočet matlabovskú štatistickú pomôcku *normcdf*, ktorá však nie je štandardnou výbavou Matlabu a preto nie je vždy k dispozícii. Musíme si preto poradiť inak. Pripomíname, že hlavným problémom plánovaného výpočtu je nevyjadriteľnosť primitívnej funkcie k f v normálnom analytickom tvare.

1. Presvedčme sa o tom, že s primitívnou funkciou k f_N to naozaj nejde. Prikážeme Matlabu, aby ju funkciu našiel:

```
>> s=sym('s')
>> F=int(f(s))

F = 1125899906842624/5644425081792261*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*s)
```

Matlab sa tvári, že ju našiel, ale pozornejší pohľad odhalí záhadnú *erf*, ktorej identitu zistíme po ďalšom výsluchu:

```
>> help erf
 ERF Error function.
    Y = ERF(X) is the error function for each element of X.  X must be
    real. The error function is defined as:

      erf(x) = 2/sqrt(pi) * integral from 0 to x of exp(-t^2) dt.
```

Za menom *erf* je ten integrál, ktorý mal Matlab vyrátať a nevyrátal...

V tejto chvíli netreba podľahnúť skepse. Posledná zlá správa je v skutočnosti dobrou. Hoci *erf* predstavuje to, čo Matlab nevedel zintegrovať, neznamená to, že s tým Matlab nevie pracovať. Práve naopak – máme v rukách Matlabu zrozumiteľný predpis hľadanej funkcie. Ten už len musíme upraviť pripočítaním konštanty (vyššie uvedená F sa z hľadiska matematickej analýzy musí interpretovať ako F+c) tak, aby napr. F(0) = 0,5:

```
>> F= inline('1125899906842624/5644425081792261*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+0.5');
```

Pre kontrolu dosaďme pár čísel (overte podľa tabuliek, či všetko sedí):

```
>> F(0)
                  ans =    5.000000000000000e-001
>> F(-4)
                  ans =    3.167124183311998e-005
>> F(3.58)
                  ans =    9.998282028962541e-001
```

A teraz k samotnému výpočtu. Ak hľadáme také u, aby quad(f, -u, u) = 0.8, potrebujeme vlastne zistiť, kedy je F(u)=0.9:

```
>>F9=inline('1125899906842624/5644425081792261*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)+0.5-0.9');
>> u=fzero(F9,0)
                           u =    1.281551565544601e+000
>> F(u)
                  ans =    9.000000000000000e-001
```

2. Hodnotu u môžeme nájsť aj čisto numericky. Vzhľadom na symetriu grafu okolo osi y si môžeme podmienku zjednodušiť a hľadať u, pre ktoré platí:

quad(f, 0, u) = 0.4 .

Výpočet uskutočníme najjednoduchšou (pomalou, ale spoľahlivou) numerickou metódou bisekcie intervalu. Najprv skúsime orientačne zistiť, kde sa približne hľadané u nachádza. Urobme niekoľko pokusov:

```
>> quad(f,0,1)
ans =
   3.413447424066043e-001
>> quad(f,0,2)
ans =
   4.772498569707956e-001
>> quad(f,0,3)
ans =
   4.986499549840275e-001
>> quad(f,0,4)
ans =
   4.999679643418110e-001
```

Z výsledkov vidno, že quad(f,0,1) < 0.4 < quad(f,0,2). Naše u budeme hľadať preto na intervale [1,2].

Stred intervalu je 1.5. Ak quad(f,0,1.5) bude menej než 0.4, je zrejmé že *u* sa nachádza v [1.5, 2]. Inak by malo byť *u* v [1, 1.5]. Máme teda na polovicu zúžený interval, v ktorom hľadáme u. Pokračujme ďalej v delení intervalu na polovicu, až kým interval nebude dostatočne malý. Aby to netrvalo dlho, využijeme matlab a prácu s cyklom. Začneme s intervalom [1,2].

```
>> a=1; b=2; n=40; delt=1/2^40
delt =
   9.094947017729282e-013
```

Ak východzí interval 40-krát podelíme na polovicu, bude mať veľkosť delt. Hľadané u bude teda týmto intervalom určené s celkom slušnou presnosťou.
Pozor – aby sme získali výsledok s požadovanou presnosťou, musíme funkciu quad prinútiť s touto presnosťou pracovať (viď posledný parameter).

```
>> for k=1:n, s=(a+b)/2; if (quad(f,0,s,1e-12)<0.4) a=s; else b=s; end, end, a,b

        a =    1.281551565544599
        b =    1.281551565544599
```

S presnosťou zhruba 5e-13 (tj. delt/2) môžeme tvrdiť, že u = (a+b)/2 = 1.281551565544599. (porovnajte s výsledkom postupu 1.)

Zistili sme teda hodnotu *u* a interval [-u, u], ktorý zodpovedá spoľahlivosti 0,8 pri normovanom normálnom rozdelení. Vrátime sa teraz k nášmu rozdeleniu N(m, 0.3^2) a zistíme, ako bude vyzerať hľadaný interval [xm-d, xm+d]. Podľa toho, čo bolo na prednáške a čo nájdeme v skriptách, vieme určiť d = sigma * u / sqrt(n), čo je u nás:

```
>> d = 0.3*1.281551565544599/sqrt(30)

d =    7.019347010548373e-002
```

Pri požadovanej spoľahlivosti 0.8 teda možno hľadať strednú hodnotu m na intervale
[12.69912825911816, 12.83951519729544].

Úlohy:

Riešte vyššie uvedený problém (tj. odhad m) pre požadovanú spoľahlivosť 0.9, 0.95, 0.99 a 0.999. To isté pri východzom rozdelení N(m, 0.1^2) a N(m, 2.5^2). Ako sa mení interval odhadu v závislosti od požadovanej spoľahlivosti a od sigmy východzieho rozdelenia?

Neznáma sigma:

Ak sigmu nepoznáme, nahradíme ju jej odhadom S=sqrt(var(x)).

```
>> S=sqrt(var(x))
S =
   7.818833709393946
```

Ak miesto presnej sigmy používame odhad S, „zaplatíme za to“ tým, že interval [-u,u] normovanej premennej budeme hľadať (namiesto normálneho rozdelenia) zo Studentovho rozdelenia stupňa voľnosti 29, čo bude vyzerať predbežne (bez normovania) takto:

```
>> g=inline('(1+x.^2/29).^(-15)');
```

Potrebujeme predovšetkým zistiť normovací koeficient c_k. Jeho hodnotu zistíme ako integrál z funkcie g na intervale (-∞,∞).

A. Jedna možnosť je zintegrovať (nájsť primitívnu funkciu ku g) a potom vypočítať jej limitu v nekonečne. Tento postup je presný, ale nevyhneme sa symbolickému módu.

```
>> s=sym('s'); int(g(s))
ans =

297558232675799463481/28*s/(29+s^2)^14+277036975249882259103/728*s/(29+s^2)^13+
79608326221230534225/5824*s/(29+s^2)^12+63137638037527665075/128128*s/(29+s^2)^11+
1306295959397124105/73216*s/(29+s^2)^10+95094341871821295/146432*s/(29+s^2)^9+
55744959028309035/2342912*s/(29+s^2)^8+28833599497401225/32800768*s/(29+s^2)^7+
331420683878175/10092544*s/(29+s^2)^6+2285659888815/1835008*s/(29+s^2)^5+
709342724115/14680064*s/(29+s^2)^4+8153364645/4194304*s/(29+s^2)^3+
1405752525/16777216*s/(29+s^2)^2+145422675/33554432*s/(29+s^2)+
5014575/33554432*29^(1/2)*atan(1/29*s*29^(1/2))
```

Kto sa nezdesil a vládze sa lepšie pozrieť na výsledok, môže si všimnúť, že je to súčet 15 sčítancov. Zaujíma nás iba hodnota výrazu v nekonečne, preto prvých 14 členov môžeme ignorovať, totiž ich limita v nekonečne je nulová. Limita atan(...) v nekonečne je pi/2 a v mínus nekonečne -pi/2, takže hodnota integrálu na intervale (-∞,∞) bude

5014575/33554432*29^(1/2)*pi/2 - 5014575/33554432*29^(1/2)*(-pi/2) =
5014575/33554432*29^(1/2)*pi .

Po vyčíslení je teda c_k = 2.528326235644034. ///

B. Iný postup, ako zistiť hodnotu integrálu, je numerický. Budeme funkciu g integrovať na intervale [-w, w], pričom hranice budeme posúvať stále viac k nekonečnu. Budeme sledovať, k čomu hodnota integrálu konverguje a zároveň musíme dávať pozor, či matlab nenarazí na hranice svoje presnosti. Pozor na prehnaný optimizmus – tu neplatí nutne, že čím väčšie w, tým presnejší výsledok.

```
>> w=100000000000; quad(g, -w,w, 1e-15)
ans =     2.528326235644036e+000

>> w=1000000000000; quad(g, -w,w, 1e-15)
ans =    2.528326235644035e+000

>> w=10000000000000; quad(g, -w,w, 1e-15)
ans =    2.528326235644035e+000

>> w=100000000000000; quad(g, -w,w, 1e-15)
ans =    2.528326235644035e+000

>> w=1000000000000000; quad(g, -w,w, 1e-15)
ans =    2.528326235644035e+000

>> w=10000000000000000; quad(g, -w,w, 1e-15)
Warning: Minimum step size reached; singularity possible.
(Type "warning off MATLAB:quad:MinStepSize" to suppress this warning.)
> In C:\MATLAB6p5p1\toolbox\matlab\funfun\quad.m at line 85
ans =    2.528326570925481e+000
```

Tu sme narazili na hranice schopností matlabu, preto posledný výsledok už neberieme do úvahy. Podľa predošlých vidíme, že hodnota sa ustálila na 2.528326235644035, čo viac-menej zodpovedá presne zistenému výsledku v bode A. ///

Potrebné Studentovo rozdelenie získame normovaním a bude vyzerať takto:

```
>> gs=inline('(1+x.^2/29).^(-15)/2.528326235644034');
```

Pre spoľahlivosť 0.8 potrebujeme príslušné u, ktoré zistíme numericky ako v predošlom prípade (postup 2.):

```
>> a=1; b=2; for k=1:40, s=(a+b)/2; if (quad(gs,0,s, 1e-13)<0.4) a=s; else b=s; end, end, a,b

a =    1.311433647300873
b =    1.311433647301783

>> u=(a+b)/2

        u = 1.311433647301328
```

Stred posledného vypočítaného intervalu je naše u. Pokračujme prechodom k nášmu konkrétnemu prípadu:

```
>> d = s*u/sqrt(30)
d =    1.872094086440170

>> xm-d, xm+d
ans =    1.089722764176663e+001
ans =    1.464141581464697e+001
```

Hľadané m je na 80% na [10.89722764176663, 14.64141581464697].

Úlohy:

Riešte vyššie uvedený problém (tj. odhad m) pre požadovanú spoľahlivosť 0.9 (x dĺžky 40), 0.95 (x dĺžky 40), 0.99 (x dĺžky 100) a 0.999 (x dĺžky 100 a 1000).

Výsledky:

Výsledky (okrem c_k a u) sú závislé od náhodného vektora x a preto môžu vychádzať rôzne. Do istej miery by však mali byť podobné nasledujúcim:

0.9 – 40:

x = rand(1,40)*25;
S = 6.319242499118779
c_k = 2.522746244608449
u = 1.684875148523588
eps = 1.310331859910942
xm = 13.28626999114242
m ∈ [11.97593813123148, 14.59660185105337]

0.95 – 40:

u = 2.022690563520428
eps = 2.020991441163362
m ∈ [11.26527854997906,15.30726143230579]

0.99 – 100:

x = rand(1,100)*25;
S = 7.019357412541726
c_k = 2.512966035103634
u = 2.626405887084275
eps = 1.843568163184823
m ∈ [10.41730670207863, 14.10444302844828]

0.999 – 100:

u= 3.391533401525066
eps = 2.380638512187783
m ∈ [9.880236353075674, 14.64151337745124]

0.999 – 1000:

x = rand(1,100)*25;
S = 7.400471713237401
c_k = 2.507255637374814 (výpočet IBA numericky, integrovanie v symb. móde zlyháva)
u = 3.300296431014886
eps = 0.7723468021376573
m ∈ [11.98011343968271, 13.52480704395802]