Kontrollcsoport-generálási lehetőségek retrospektív egészségügyi vizsgálatokhoz

Szekér Szabolcs[1], Dr. Fogarassyné dr. Vathy Ágnes[2]
[1]Pannon Egyetem Rendszer- és Számítástudományi Tanszék,
szekersz@gmail.com
8200 Veszprém Egyetem utca 10.
[2] Pannon Egyetem Rendszer- és Számítástudományi Tanszék,
vathy@dcs.uni-pannon.hu
8200 Veszprém Egyetem utca 10.

Összefoglaló: **Retrospektív vizsgálatok esetén az eredmények validálása gyakran igényli olyan független kontrollcsoport kialakítását, amely bizonyos tulajdonságok mentén hasonló eloszlást mutat, mint az elemzés alapját adó mintacsoport. Jelen publikációban ezen feladat megoldására alkalmas lehetőségeket mutatunk be, rávilágítva azok előnyeire és hátrányaira.**

Bevezető

A retrospektív vizsgálatok hátránya, hogy a vizsgálatok eredményeinek értékeléséhez szükséges kontrollcsoport a vizsgálat jellegéből fakadóan az adatok rögzítésekor nem került meghatározásra, így az nem áll az elemzők rendelkezésére. A vizsgálati eredmények kontrollcsoporttal történő összevetése azonban elengedhetetlen az eredmények értékelése szempontjából, így a kontrollcsoport utólagos kialakítása szükséges. A kontrollcsoportba bevonható személyeket oly módon kell meghatározni, hogy azok a vizsgált kezelés/beavatkozás tekintetében eltérjenek a vizsgált betegcsoporttól, és az eredményekre hatással lévő paraméterek tekintetében hasonló eloszlást mutassanak a betegcsoport egyedeihez. Ezen hasonló eloszlások hiányában a paraméterek kiigazítása szükséges. Munkánk során olyan algoritmusok kialakításával és vizsgálatával foglalkozunk, amelyek egy előre meghatározott populációból olyan kontrollcsoport kiválasztását teszik lehetővé, amely kontrollcsoport meghatározott tulajdonságok tekintetében hasonló eloszlást mutat egy adott mintacsoporthoz képest.

Módszer

Kutatásaink során négy algoritmust vizsgáltunk, melyek közül három algoritmus a *k*-legközelebbi szomszéd elvén alapul, míg a negyedik algoritmus alapját a klasszikus rétegképzéses mintavételezés gondolata adja.

Az első algoritmus (továbbiakban *Algoritmus1*) a *k*-legközelebbi szomszéd algoritmus alapváltozata, amely a vizsgálat szempontjából releváns tulajdonságok által meghatározott N-dimenziós térben keresi a mintacsoport egyedeihez legközelebb eső kontrollcsoportbeli egyedeket. Az algoritmus működése során az adatok normálását követően az Euklideszi-normát alkalmazva meghatározza a mintacsoport és a populáció minden egyes elemének távolságát, majd a kontrollcsoport kívánt méretétől függően kiválasztja a mintacsoport összes eleméhez a *k*-legközelebbi populációbeli egyedet. Az így kiválasztott egyedek képezik a generált kontrollcsoport egyedeit.

A második algoritmus (*Algoritmus2*) az első algoritmus továbbfejlesztésének tekinthető. Az Algoritmus1 alkalmazásának eredményeképpen előfordulhat, hogy a populáció egyedeinek eloszlásától függően egyes populációbeli egyedek több mintacsoportbeli egyedhez is hozzárendelésre kerülnek. Ily módon az Algoritmus1 bizonyos esetekben csak alulról közelíti a kontrollcsoport kívánt méretét. A második algoritmus ezt a problémát küszöböli ki oly módon, hogy azon populációbeli egyedet, amely több mintacsoportbeli egyedhez is hozzátartozna a *k*-legközelebbi szomszéd elve alapján, ahhoz a mintacsoportbeli egyedhez rendeli, melyhez a legközelebb található. Azon mintacsoportbeli egyedek esetében, ahol a *k*-legközelebbi szomszéd elvén kiválasztott egyedek száma nem éri el *k*-t, iteratívan addig növeljük a figyelembe vehető legközelebbi szomszédok számát, amíg a párosított populációbeli egyedek száma el nem éri a kívánt értéket.

A harmadik algoritmus (*Algoritmus3*) az Algoritmus2 továbbfejlesztett változatának tekinthető, s szintén garantálja a kialakítandó kontrollcsoport pontos méretét. Az Algoritmus3 működése során a kiválasztáskor jelentkező ütközések feloldását az előző algoritmustól eltérően már nem a legkisebb Euklideszi távolság elvén valósítja meg, hanem egy általunk definiált hibafüggvényt értékel ki. Ezen hibafüggvény (E_{dist}) a normák távolságát veszi figyelembe és az ütközésben részt vevő populációbeli egyed esetében a mintaegyed és a populációegyed összerendelését úgy valósítja meg, hogy az ütköző egyedet helyettesítő egyedek által meghatározott mintaegyed-populációegyed távolságok összege minimális legyen. Az E_{dist} hibafüggvény P_j ütközésben részt vevő populációbeli egyedre és egy A_i mintacsoportbeli egyedre vonatkozóan a következőképpen határozható meg:

$$E_{dist_{A_iP_j}} = \left\| A_i - NN_{A_i}^{(1)} \right\| - \left\| A_i - NN_{A_i}^{(2)} \right\| = \left\| P_j \right\| - \left\| A_i - NN_{A_i}^{(2)} \right\|,$$

ahol $NN_{A_i}^{(1)}$ az A_i mintacsoportbeli elem első, $NN_{A_i}^{(2)}$ pedig az A_i mintacsoportbeli elem második legközelebbi szomszédját jelöli, j pedig a beválasztásra jelölt populációbeli elem indexe, amely egyben az A_i egyed legközelebbi szomszédja is. Az alkalmazott hibafüggvény minden mintacsoportbeli elem esetén kiszámolja a hozzá legközelebb és második legközelebb eső populációbeli elem távolságát. Amennyiben egy populációbeli elem több mintacsoportbeli elemnek is a legközelebbi szomszédja, akkor azon mintacsoportbeli egyedhez választjuk ki ezt a populációbeli egyedet, amely esetében a hibafüggvény értéke a legnagyobb. Ezt a kiválasztási folyamatot addig folytatjuk iteratívan, míg minden ütközést fel nem oldunk a hibafüggvény kiértékelése által.

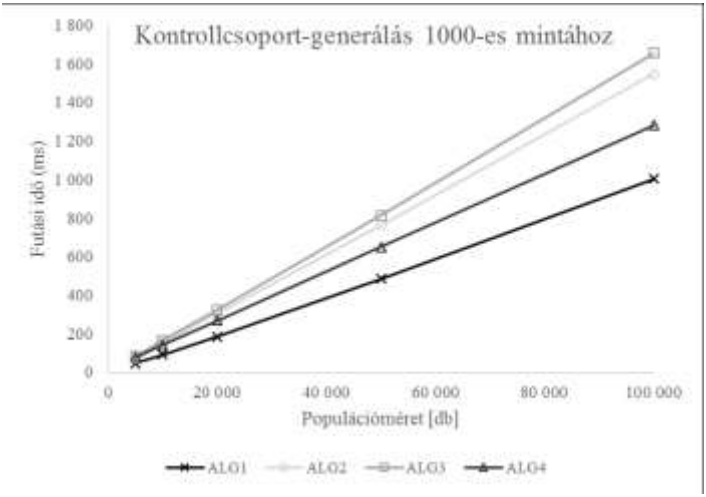
A negyedik algoritmus (*Algoritmus4*) az előző három algoritmushoz képest teljesen más megközelítést alkalmaz. Ezen algoritmus működése során a klasszikus rétegképzéses mintavételezés módszerét felhasználva legenerálja a mintacsoportban előforduló rétegeket, majd megvizsgálja, hogy melyik réteghez mennyi elem tartozik. Ezt követően a populációban is meghatározza a mintacsoportban előforduló rétegeket, majd a mintacsoportban előforduló réteggyakoriságoknak megfelelően a populációbeli rétegek elemeiből random módon kiválogatja a kontrollcsoport egyedeit.

Összehasonlítás

Az Algoritmus1 által szolgáltatott kontrollcsoport hasonlósága nagymértékben függ a kiindulási populáció méretétől és egyedeinek eloszlásától. Ha a populációból leválogatott kontrollcsoport mérete pontosan megegyezik a kívánt kontrollcsoportmérettel, akkor az eredményhalmaz a kívánt eloszlást mutatja, ellenkező esetben a kontrollcsoport egyedeinek eloszlása eltérést mutat a mintacsoport eloszlásához képest. Az Algoritmus2 az alapalgoritmus esetleges hibáját hivatott kijavítani, azonban kis populáció esetében a kontrollcsoport eloszlásában torzulásokat eredményezhet. A harmadik algoritmus az előző két algoritmus hibáit küszöböli ki. Garantálja, hogy a generált kontrollcsoport mérete megfelel az elvárásoknak és csökkenti az Algoritmus2 kezdetleges megközelítésének pontatlanságát. Mindezen három algoritmus alapvető hibája azonban, hogy a mintacsoport és a populáció méretének növelésével megnő a memóriaigényük, és nagyobb elemszámú mintacsoport-populáció páros esetén egy átlagos asztali számítógép már nem képes végrehajtani őket.

Az Algoritmus4 előnye, hogy a populáció megfelelő mérete és egyedeinek megfelelő eloszlása esetén mindig a mintacsoporttal teljesen azonos eloszlású kontrollcsoportot eredményez. Működésének alapja azonban egyben a hátránya is, hiszen kontrollcsoport alapját adó populációbeli egyedek nem megfelelő eloszlása esetén (pl. kevés elem egy rétegben) működése nehezen hangolható. Mindemellett azonban a rétegek meghatározása nem igényel sok memóriát, ezért futtatása során nem fordul elő memóriahiány.

A fent említett négy algoritmus futási ideje lineáris. A mintacsoport-populáció páros méretétől függően pár ezredmásodperctől pár másodpercig terjed, ami megfelelően gyorsnak tekinthető. Ezen futási időket szemlélteti az 1. ábra, amely a 4 algoritmus futási idejét hasonlítja össze abban az esetben, ha egy 1000 fős mintacsoporthoz keresünk 1000 elemet tartalmazó kontrollcsoportot, eltérő méretű (5000, 10000, 20000, 50000 100000 főt tartalmazó) populáció esetén. Jól látható, hogy az Algoritmus1 futási ideje kisebb, mint a második és harmadik algoritmusé, mivel nem követeli meg, hogy a generált kontrollcsoport mérete pontosan megfeleljen az elvártaknak. Az Algoritmus4 futási ideje ugyan meghaladja az Algoritmus1 futási idejét, de gyorsabb, mint a második és harmadik módszer.



1. ábra Futási idők 1000-es minta és 1000-es kontrollcsoport esetén

Fontos kiemelni, hogy bár mindegyik algoritmus futási ideje lineáris, nagyobb elemszámok esetén az Algoritmus4 jobban teljesít, mivel a problématér mérete additívan és nem multiplikatívan nő, vagyis x méretű minta és y méretű populáció esetén, ha bármelyik csoport méretét k-val növeljük, akkor az elvégzendő lépések száma $x + k$-ra vagy $y + k$-ra nő, nem pedig $(x + k) * y$-ra vagy $(y + k) * x$-re. Ez skálázhatóság szempontjából azt jelenti, hogy jóval robosztusabb, vagyis kevésbé érzékeny a problématér méretének növekedésére. Ugyanez a tulajdonság nyilvánul meg a korábban említett memória-felhasználás terén is.

Konklúzió

Kutatásunk során nyilvánvalóvá vált, hogy bizonyos áldozatokat kell hoznunk a kívánt eredmény eléréshez – beszéljünk akár futásidőbeli, memóriaigénybeli vagy pontosságbeli megkötésekről. Az optimális algoritmus megvalósításához hosszú és rögös út vezet, mely rengeteg próbálkozást és tesztelést tartalmaz. A fentebb taglalt algoritmusok közül kettő, az Algoritmus3 és Algoritmus4, a jövőbeli kutatásaink alapját képező algoritmusok. Futási idejük megfelelő, pontosságuk növelését pedig olyan további hibafüggvények bevezetésével tervezzük, melyek segítségével nem csupán lokális optimumot érünk el, hanem globálisan optimális döntést tudunk hozni a kontrollcsoport egyedeinek kiválasztásához.

Köszönetnyilvánítás

A publikációt és a kapcsolódó kutatásokat a VKSZ_12-1-2013-0012 azonosítójú "Világszínvonalú Intelligens és Inkluzív Egészségügyi Információs és Döntéstámogató Keretrendszer (Analytic Healthcare Quality User Information) kutatása" című projekt keretében Magyarország Kormánya támogatta.