Döntéstámogatás a médiainformatikában

Nemhivatalos jegyzet

v0.3

A jegyzet a 2009/10 őszi félévében készült, így annak anyagát követi. A jegyzet bizonyos helyeken nem az előadáson leadott anyaggal megegyező, külső forrásból pótoltuk a nem érthető részeket.

A jegyzet tartalmaz(hat) hibákat, ezekért természetesen nem vállaljuk a felelősséget. Akinek nem tetszik a jegyzet, az ne használja, vagy a hibák jelentésével segítse javulását.

Ha módosítod/ kiegészíted / átformázod / átírod ezt az anyagot, akkor ne felejtsd el megosztani, hogy a jövő generációi is profitálhassanak a munkádból! Mindig készíts egy szerkeszthető (pl. doc) és egy kifejezetten nyomtatásra kitalált (legjobb a pdf) formát. Az eredeti szerzőket is említsd meg a módosított munkában!

Külön köszönjük Szűcs Gábornak, hogy átnézte a jegyzetet, és javította néhány tévedésünket.

A jegyzet főleg a következő arcoknak köszönhető:
azeroth, FBálint, Feri, Hidden, Misi, Nagy Bálint, ping-win

A jegyzet letölthető:
https://wiki.sch.bme.hu/bin/view/Infoszak/DontesTamogatasAMediainformatikaban

1. Előadás

Döntéshozás csoportosítása

Döntéshozó: aki a tényleges döntést meghozza
Probléma tulajdonos: akinek van valami "eldönteni valója"

Döntéshozó - probléma tulajdonos szempontból

	egyéni döntéshozó	csoportos döntéshozó
egyéni probléma tulajdonos	pl: felvegyek-e egy adott tárgyat	pl: amerikai bíróság
csoportos probléma tulajdonos	pl: cégnél a főnök dönt valamiről	pl: országgyűlés

Bizonytalansági szempontból

- Biztos: mindent pontosan tudunk, nincs bizonytalansági tényező
- Kockázatos: az egyes "esetek" bekövetkezési valószínűségeit tudjuk
- Bizonytalan: nincs semmilyen ismeretünk a valószínűségekről sem

Döntési tábla

Egy olyan táblázat, aminek az első oszlopa rögzíti az opciókat (azaz milyen döntést hozhatunk), a többi oszlop a döntés "eredményét" adja meg, az oszlop első sorában az adott "eredmény" bekövetkezésének valószínűségével.

Például: Van egy hírportálunk és el kell dönteni, hogy Sport vagy Bulvár hír legyen a főoldalon. Látogatottsági értékeket tartalmaz a táblázat, egy kutatás alapján pedig tudjuk, hogy milyen valószínűséggel kapjuk meg az adott cellában szereplő értéket. Jelenleg Bulvár híreket rakunk ki, ami 200k látogatót vonz.

	V1 =25%	V2=75%
Bulvár	200k	200k
Sport	100k	300k

Tehát, ha Sportra váltunk, akkor 25% val-el lesz 100k látogatónk, és 75% val-el lesz 300k. Bulvárnál az ismert 200k érték változatlan marad (mert ezt biztosan tudjuk). Megoldás szerint várható érték alapján a Sport mellett döntünk, mert 250k látogatót eredményez. - ez bizonytalansági szempontból: biztos (V1 + V2 ... szummában 1 kell legyen ebben az esetben, így tudunk számolni várható értéket.)

Döntési stratégiák bizonytalan esetben

Példa: Egy médiaarchívum el akarja dönteni, hogy min érdemes adatot tárolni, hogy az "utókor" le tudja majd olvasni 20 év múlva az adatokat. 120 archívummal számolunk.

	V1	V2	V3
DVD	0	60	120
BLU-RAY	30	60	90
Mágnesszalag	61	62	63
HDD	60	65	70

V1,V2,V3 bekövetkezésének valószínűségét nem ismerjük, ezért nem lehet várható értéket számolni.

sormax: egy adott sor legnagyobb eleme, kiszámítva minden sorra sormin: egy adott sor legkisebb eleme, kiszámítva minden sorra

Stratégiák

	leírás	döntési mechanizmus	döntés a példában
Wald	a legrosszabb bekövetkezésére számítunk, válasszuk azt, ami a "legrosszabb esetben a legjobb"	max(sormin) által meghatározott opció	Mágnesszalag (61 miatt)
Optimista	a legjobb bekövetkezésére számítunk, válasszuk azt, ami a "legjobb esetben a legjobb"	max(sormax) által meghatározott opció	DVD (120 miatt)
Hurwitcz	Wald és Optimista súlyozott változata, a „p” pesszimista paraméterrel számítva	max(sormin*p + (1-p)*sormax) által meghatározott opció	p=0.75 paraméterrel: HDD (62.5 miatt)
Laplace	úgy számolunk, mintha minden bekövetkezési eset azonos valószínűségű lenne	max(sorok értékeit összeadjuk, majd osztjuk a sor elemeinek számával = várható érték számítás sorra) által meghatározott opció	HDD (65 miatt)
Savage	minimális "megbánás" mellett döntünk -> megbánási mátrix	min(megbánási mátrix sormax)	BLU-RAY (magyarázat lent)

Megbánási mátrix (r)

Minden elemet kivonjuk az oszlop maximális eleméből, így kapjuk a "megbánást". Tehát:

	V1	V2	V3	sormax
DVD	61	5	0	61
BLU-RAY	31	5	30	**31**
Mágnesszalag	0	3	57	57
HDD	1	0	50	50

Döntési folyamat lépései

- alternatívák összegyűjtése
- információk begyűjtése az alternatívákról
- szempontrendszer kidolgozása (szempontok összemérhetővé tételével együtt)
- algoritmus konstruálása/kiválasztása a legjobb meghatározására
- döntés
- végrehajtás

Az első 4 a döntés előkészítés, az 5. a döntés, a 6. a végrehajtás fázisa.

MCDM (Multi-criteria decision analysis)

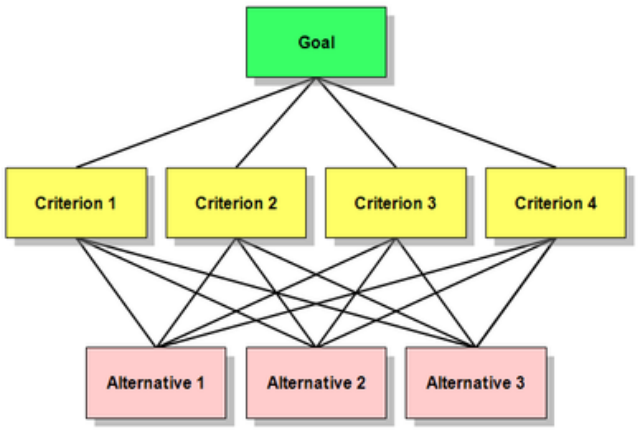
Többszempontú döntésanalízis, a döntéselméletnek az a része, mely több szempontból megvizsgált alternatívák közti választással foglalkozik.

MAUT (Multi-attribute utility theory)

Döntéstámogató folyamat, súlyozza az egyes szempontokat, így lehetőséget nyújt a fent említett algoritmusok használtára. Ehhez súlyfüggvényeket használ. (pl.: $f_1(S_1)$) Az egyes szempontok értékeit azonos „skálára” lehet hozni.

	Szempont 1	Szempont 2	Szumma „szempont”
Döntési lehetőség 1	$f_1(S_1)$	$f_2(S_1)$	$f_1(S_1)+ f_2(S_1)$
Döntési lehetőség 2	$f_1(S_2)$	$f_2(S_2)$	$f_1(S_2)+ f_2(S_2)$
…			

AHP (Analytic Hierarchy Process)



Az AHP egy hierarchikus model, ahol szempont párokat hasonlítanak össze a döntéshozók. (Pl.: szempont1/szempont2). Az értékelést egy egy skálán adják meg, legyen ez most 1-10.

	Szempont1	Szempont2
Szempont1	1	5
Szempont2	0.3	1

Ezekből a szempont párokból készül egy táblázat. Az A/B és B/A párok szorzatának 1-nek lennie, mert két érték összehasonlítása még KONZISZTENS kell, hogy legyen! 3 érték-nél már nem szükséges, pl. B kétszer olyan jó, mint A; C pedig ötször olyan jó, mint B; C pedig 7-szer olyan jó, mint A.

A példa szerint a Szempont1/Szempont2=5, tehát az első Szempontot 5-ször fontosabbnak ítélte a döntéshozó. A táblázat teljes kitöltése után egy sajátvektor számítással kideríthetőek a tényleges súlyok, ezek alapján pedig a MAUT módszerrel már meghozható a döntés.

Polano

A módszer lényege: a döntési mátrix sorai a szempontokat, oszlopai az alternatívákat tartalmazzák → sorok oszlopok metszetében az A_j alternatíva C_i szempont szerinti értéke (értékelése) áll. Minden szempontnál külön-külön meghatározzuk, hogy mely értékeket tartunk pl. jónak, közepesnek ill. rossznak. A döntési táblázat ezután az értékelések helyett színekkel jelzi a besorolásokat.

Kvázi az elkészített táblázatot beszínezzük, hogy lássuk a jó (zöld), közepes (sárga) és rossz (piros) értékeket. Ezek után már csak úgy érzésből ránézésre döntünk…

2. Előadás

Rangegyesítő módszerek:

Minden módszer alapja: felírjuk a döntéshozók által javasolt alternatívákat, az 1. a leginkább támogatott, az utolsó helyen a legkevésbé javasolt.

	1.	**2.**	**3.**	**4.**
$\mathbf{d_1}$	B	C	A	D
$\mathbf{d_2}$	C	D	B	A
$\mathbf{d_3}$	B	D	A	C

Borda:

Az 1. alternatíva N. pontot ér, az N. alternatíva pedig 1 pontot.

	1.	**2.**	**3.**	**4.**
$\mathbf{d_1}$	B	C	A	D
$\mathbf{d_2}$	C	D	B	A
$\mathbf{d_3}$	B	D	A	C
	4	**3**	**2**	**1**

A	2+1+2	5
B	4+4+2	10
C	4+3+1	8
D	3+3+1	7

Ennek alapján az alternatívák pontja a jobboldali táblázatban láthatók, a B nyert, utána C,D és A.

Bernardo

Első lépés a rang-pozíció mátrix felírása, nagyon hasonló az eredeti mátrixhoz, annak egy számosítása, itt az oszlopokban vannak az alternatívák, a cellákban pedig, hogy melyiket hányadik helyre sorolta az adott döntéshozó. (a_1 = A, a_2 = B, a_3 = C, a_4 = D)

RP	$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{a_3}$	$\mathbf{a_4}$
$\mathbf{d_1}$	3	1	2	4
$\mathbf{d_2}$	4	3	1	2
$\mathbf{d_3}$	3	1	4	2

EGY	$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{a_3}$	$\mathbf{a_4}$
1.	0	2	1	0
2.	0	0	1	2
3.	2	1	0	0
4.	1	0	1	1

A rang-pozíció mátrix alapján fel tudjuk írni egyszerűen az egyetértési mátrixot, azaz adott alternatíva adott pozíciójában hányan értettek egyet.

Ennek vesszük az összes permutációját (mint determináns számításkor) és a maximális összegű permutációt választjuk ki (valójában ránézésre ki lehet választani a legnagyobbat szerintem).

Hátrány: csak az egyetértést nézi, azt nem, hogy hányadik helyre sorolták be.

Eredmény: B, D, A, C

Cook Seiford

Bernardo módszerét javítja, az ellenzési mátrixot vezeti be. Az eredmény itt az egyetértésivel ellentétben a permutációk minimuma lesz. Az alábbi példának több

megoldása is van, mind a kettő 10 értékű: B, C, D, A / B, D, A, C . A kitöltést a rang-pozíció mátrix alapján végezzük:

ELL	a_1	a_2	a_3	a_4
1.	7	2	4	5
2.	4	3	3	2
3.	1	4	4	3
4.	2	7	5	4

A kitöltés menete: $\mathbf{a_{i|j} = \sum | j - a_{ij} |}$

$\mathbf{a_{1|1} = | 1 - 3 | + | 1 - 4 | + | 1 - 3 | = 7}$

$\mathbf{a_{1|2} = | 2 - 3 | + | 2 - 4 | + | 2 - 3 | = 4}$

$\mathbf{a_{1|3} = | 3 - 3 | + | 3 - 4 | + | 3 - 3 | = 1}$

$\mathbf{a_{1|4} = | 4 - 3 | + | 4 - 4 | + | 4 - 3 | = 2}$

Köhler

Outranking mátrix: melyik alternatíva hányszor előzi meg a másikat.

Kitöltése a rang-pozíció mátrix alapján, önmagával nem nézzük: x.

A megoldás a sorok minimumának a maximuma. Ez most az a_2 = B lett. Ezek után elhagyjuk a B-t, és újraszámoljuk az outranking mátrixot az $a_{1,3,4}$ -el, ebből megtudjuk ki a 2. stb.

OUT	a_1	a_2	a_3	a_4	Min
a_1	x	0	1	1	0
a_2	3	x	2	2	2
a_3	2	1	x	2	1
a_4	2	1	1	x	0

Reciprok rang

Helyezések reciprokának összegeinek a reciproka: $P_i = \frac{1}{\sum_i \frac{1}{Z_i}}$ Z_i =alternatíva helye Borda szerint

Min (Pi) = az 1. helyezett.

A: 1 / (1/3 + 1/3 + 1/4) = 1,09

B: 1 / (1/1 + 1/1 + 1/3) = 0,43

Példa

A legjobb 5 videót akarom bejelölni 10.000 közül, felírom a rang-pozíció mátrixot a_1 - $a_{10.000}$

Ahol szavaztak, ott beírom 1-5-ig a pontot, a többi 9995 érték a sorban pedig legyen egy olyan nagy szám, ami nem zavarja a mátrixszámolásokat, pl. N/2 = 5000.

A legkisebb összegű oszlop lesz a legjobb film.

Döntés automatizálás

Esetek: Spam / nem spam, stb. Két csoport:

- osztályozás: előre megadott csoportok (felügyelt tanulás)
- klaszterezés: az egymáshoz való hasonlóság és eltérés alapján alakulnak majd ki a csoportok

	ismert változó	**cél változó**
tanuló állomány	ismert	ismert
teszt állomány	ismert	?

Döntéstámogató rendszerek

- Általános: Decision Support System (DSS), Group DSS (GDSS)
- Felsővezetői: Management Information System (MIS), Executive IS (EIS)
- Expert System (ES)
- Data Mining (DM)

3. Előadás

Adatbányászat:
http://adatbanyaszat.tmit.bme.hu/~samirello/DTMI/DTMI_clickstream.pdf

4. Előadás

Döntési fák

Osztályozási és regressziós problémák

Az osztályozási problémák során folytonos és diszkrét változók értékei alapján akarunk egy korlátos diszkrét értéket – kategória típusú változót meghatározni, vagyis az ismert tulajdonságok alapján előre meghatározott osztályokba akarunk sorolni. Ilyen feladat például az, hogy filmeket előre meghatározott műfajokba soroljunk.

Regressziós problémák esetén az osztályozástól eltérően egy folytonos változó értékét szeretnénk megjósolni. Ilyen feladat mondjuk családi házak értékének megbecslése a telek fekvése, a ház kora, a ház állapota, az emeletek száma stb. alapján.

A változók felosztása a következő:

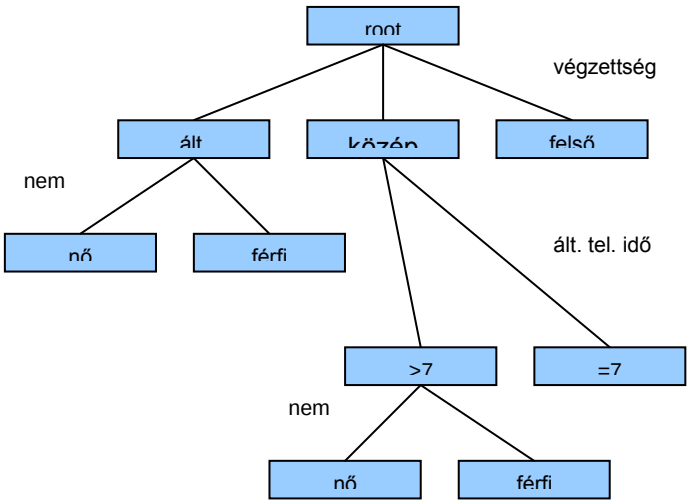
- **nominális**: más néven kategória jellegű változó, korlátos diszkrét értékeket vehet fel. Két ugyanolyan típusú nominális változó közt csak az ekvivalencia reláció értelmezett. Pl.: Színtípus:={piros, fehér, zöld, fekete, sárga, kék, hupilila}
- **ordinális**: a felvehető értékek közt egy sorrendezés értelmezett. Az ilyen jellegű változókra a medián értelmezett. Pl.: Lóversenyen a lovak célba érkezési sorrendje. Ez egy sorrend, de nem mond semmit a konkrét beérkezési időkről.
- **intervallum**: Ezeknél a változóknál adott egy mérték, és az egyes értékek közötti távolság mérhető. Itt értelmezett a számtani közép is. Pl.: hőmérséklet a Celsius skálán. Itt a mérték a víz légköri nyomás alatt mért forráspontja és a fagyáspontja közti különbség 1/100-ad része. A nulla pont választása önkényes, használhatók negatív értékek is. Az adott értékek közti arány nem hordoz jelentést (pont az önkényesen választott nulla pont miatt), de az értékek közti különbségek aránya már igen. Tehát ha eltolnánk a Celsius skálát, akkor a különbségek aránya ugyanúgy megmaradna. Minden olyan statisztikai jellemző értelmezett, melyhez különbségképzés vagy átlagolás használt csak.
- **arány**: a különbség az előző típussal, hogy itt ismert egy nem önkényesen választott nulla pont. Pl.: hőmérséklet mérése Kelvin skálán. Itt az abszolút nulla pont nem egy önkényes választás, hiszen 0K hőmérsékleten a részecskék kinetikus energiája 0. Erre a típusra minden statisztikai jellemzés értelmezett.

Az első kettő kvalitatív, a második kettő kvantitatív jellemzést ad.

A döntési fa egy döntéstámogató eszköz, mely fa struktúrában ábrázolja a döntéseket és a következményeiket. Pl.:

Nem	Ált. telefonálás hossza	Végzettség	Ügyfél
férfi	18	felső	jó
nő	27	közép	jó
férfi	7	ált.	rossz

Ez alapján egy döntési fa:



Döntési fa algoritmusok

A döntési fa algoritmusok az adatok alapján építenek egy döntési fát. Az algoritmusok bemenete tehát egy mérés, melyben az egyes mérések osztályozva vannak. Egyszerűbben: van egy táblánk, ahol az oszlopok az attribútumok, a sorok az egyes mérések és van egy oszlop, mely megmondja, hogy az adott mérést melyik osztályba soroltuk.

ID3 (http://en.wikipedia.org/wiki/ID3_algorithm)

Az ID3 úgy működik, hogy kiválasztja azt az attribútumot, ami a legjobban osztályoz és létrehoz neki egy csomópontot a döntési fában. A mérést szétbontja a kiválasztott attribútum szerint, alméréseket kapva (melyek oszlopai között tehát nem szerepel a kiválasztott attribútum), melyekre külön-külön eldönti, hogy melyik maradék attribútum osztályozza az adott almérést a legjobban.

A legjobban osztályozó attribútum az, amelynek legkisebb az entrópiája. Az entrópia: $I_e = -\sum_{i=1}^{n} p_i \log_2 p_i$, ahol n az összes lehetséges attribútum érték és osztály pár (tehát, ha A a felvehető értékek halmaza, C az osztályok halmaza, akkor $n = |A \times C|$), p_i az i-dik attribútum érték és osztály pár valószínűsége.

Az algoritmus megnézi az összes oszlopra ezt az értéket és kiválasztja azt az oszlopot, amelyiknél ez a legkisebb.

Pl.:

Zenészek száma	Felvétel helyszíne	Hangerő ingadozása	Letöltések száma	Zene típusa
<=2	stúdió	kicsi	kevés	könnyű
<=2	stúdió	kicsi	sok	könnyű
3..7	stúdió	kicsi	kevés	komolyzene

>7	koncertterem	kicsi	kevés	komolyzene
>7	szabadtér	nagy	kevés	komolyzene
>7	szabadtér	nagy	sok	könnyű
3..7	szabadtér	nagy	sok	komolyzene
<=2	koncertterem	kicsi	kevés	könnyű
<=2	szabadtér	nagy	kevés	komolyzene
>7	koncertterem	nagy	kevés	komolyzene
<=2	koncertterem	nagy	sok	komolyzene
3..7	koncertterem	kicsi	sok	komolyzene
3..7	stúdió	nagy	kevés	komolyzene
>7	koncertterem	kicsi	sok	könnyű

Letöltések száma kevés és könnyűzene:

2 db ilyen sor van, 8db „kevés” értékű sor van, így p=2/8=0.25, vagyis -0.25*log(0.25) = 0.5

Letöltések száma kevés és komolyzene:

6db ilyen sor van, 8db „kevés” értékű sor van, így p=6/8=0.75, vagyis -0.75*log(0.75) = 0.3113

Letöltések száma sok és könnyűzene:

- (3 ilyen sor / 6 „sok” értékű sor)*log(0.5) = 0.5

Letöltések száma sok és komolyzene:

- (3 ilyen sor / 6 „sok” értékű sor)*log(0.5) = 0.5

Letöltések száma attribútum entrópiája = (8 „kevés” értékű sor / 14 összesen)*(0.5 + 0.3113) + (9 „sok” értékű sor / 14)*(0.5 + 0.5) = 0.89217

Hasonlóan az összes többi attribútumra kiszámítva kijön, hogy a zenészek száma attribútum alapján érdemes először szétbontani.

C4.5 (http://en.wikipedia.org/wiki/C4.5_algorithm***)***

ID3 továbbfejlesztése, folytonos változók is megengedettek és a hiányzó értékeket is képes kezelni.

C5.0 (http://en.wikipedia.org/wiki/C4.5_algorithm, http://www.rulequest.com/see5-comparison.html)

A C4.5 továbbfejlesztése, gyorsabb, kevesebb memóriát igényel és kisebb fákat generál.

CART ()

CHAID (http://en.wikipedia.org/wiki/CHAID)

5. Előadás

Minták

Populáció minta: $P_1 = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x}^2)}{n}}$ és $P_2 = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x}^2)}{n-1}}$, Normális: N(γ, б)

Normális eloszlás mintavételi tétele: ha a populáció normális, akkor a minta is az:
$\gamma_S = \gamma$, $б_S = \frac{s}{\sqrt{nS}}$, n_S: minta számossága

Centrális határeloszlás tétel: n → ∞, a minta normális

Adatsor szétválasztása

Tanuló (2/3) és teszt (1/3) részre általában. Teszt állomány lehetőleg legyen véletlenszerű.

Szétválasztási metódusok:

Keresztvalidáció

Osszuk n (≈ 5, 10) részre, legyen a tanuló az 1...n-1., a teszt az n. rész, majd válasszuk a következőt a tesztállománynak 1…n-2,n, teszt: n-1. A hiba a hibák átlaga lesz.

Leave-One-Out

1 adat lesz a teszt a többi a tanuló. Csak ha kicsi az adatsor!

Bootstrap aggregating (Bagging)

$a_1...a_N$ húzunk véletlenül visszatevéssel (vagyis egy adatot többször is kihúzhatunk).
N-szer húzunk, amiket 1-szer se húztunk ki, azokból fog állni a teszt állomány.
Méret becslés: az, hogy egy körben nem húzzuk ki az i.-et: $p_i = 1 - \frac{1}{N}$ vagyis $(1- \frac{1}{N})^N = \frac{1}{e} \approx$ 27,8 % lesz a teszt. (ezért is hívják még „0,632 Bootstrap módszer”-nek).

Outlier adatok

Azok az adatok, amelyek „kilógnak”, valószínűsíthetőleg hiba miatt.

	V_1	…	V_j	…	V_N
a_1	3		19		
…	4		18		
a_i	452		21		
…	7		185		
a_N	2		13		

Trimmerelés

Végigmegyünk a táblázaton az 1. oszloptól, oszloponként és soronként, először az 1. oszlopból töröljük azokat a sorokat, amelyeknek az értéke kilóg. Veszély: lehet, hogy az N. oszlopnál már alig marad sorunk!

Windzorizálás

Kihúzás helyett csonkolunk: a kilógó érték helyére a megengedett max (vagy min) értéket írjuk.

Hiányzó értékek

Az érték nem hibás, hanem egyáltalán nincs megadva. A hiányzó helyre az adott oszlopban szereplő értékek átlagát írhatjuk. Amennyiben az értékek ordinálisak, pl: nyelvvizsga (A,B,C), vagy színek:

Gyakoriság alapján választunk. (Módusz)

Medián: az egyforma értékeket csoportosítva, valamilyen sorrendbe felírva, a középső.

Korrelált változóval való becslés: egy másik oszlop, amelyikkel korrelál, segíthet, pl. amikor ott alacsony az érték itt piros, mikor ott magas itt kék. Ez alapján ki tudunk találni egy átlagos értéket.

Hisztogram

Az adatokat megjeleníthetjük hisztogramon is. (Grafikonon, ahol az x tengelyen az értékek, az y tengelyen az adott értékű adatok száma található). Fontos, hogy jó felbontást találjunk, túl kicsi/túl nagy nem jó, pl: életkornál 2-5 éves felbontás.

6. Előadás

SEMMA

Az adatbányászati projekt elemzési szakaszának módszertanát hívjuk SEMMA-nak, ami az adatbányászati folyamat 5 fázisának angol nyelvű elnevezésének kezdőbetűiből alkotott betűszó: Sampling (mintavételezés), Exploration (feltárás), Manipulation (módosítás), Modelling (modellezés), Assessment (felmérés, kiértékelés). A piaci szolgáltatók adatbányászati eszközei a SEMMA-ban definiált lépések végrehajtására biztosít megoldásokat.

Mintavételezés (Sampling)

Az adatbányászathoz használandó adatbázisok mérete sokszor gigabyte nagyságrendű és folyamatosan növekszik. A szervezet különböző belső rendszereiben meglévő nagy mennyiségű adatot adatbányászathoz gyakran külső adatokkal egészítik ki, mint pl. demográfiai adatok, közvélemény kutatások, piackutatások adatai.

Az első kérdés, amit az adatbányászati ciklus folyamán fel kell tenni az, hogy szükséges-e a teljes adathalmazon dolgozni vagy esetleg elégséges egy mintán dolgozni. A legtöbb esetben nem szükséges a teljes adatállománnyal dolgozni, mivel a különböző fejlett mintavételezési eljárásoknak köszönhetően ugyanolyan pontos modelleket lehet létrehozni jelentősen rövidebb idő alatt, és lényegesen kisebb erőforrásokkal (hardverkapacitás). Az adatbányászati megoldások támogatják többek között az egyszerű (véletlen), a rétegzett és a csoportos mintavételezési eljárásokat.

Feltárás (Exploration)

A modellezést megelőzően szükséges az adatállomány feltárása, vizsgálata: vannak-e hiányzó vagy extrém értékek, amelyek az elemzést jelentősen befolyásolhatják. Amennyiben sok változónk van, érdemes megvizsgálni, hogy nem lehet-e összevonásokat alkalmazni, vagy redukálni a változók számát, illetve hogy a változóink megfelelnek-e a használni kívánt eszközök eloszlás kritériumának, stb. Ehhez a statisztikai modulokban különböző vizuális (pl.2-D,3-D grafikonok) és analitikus eszközök (klaszter elemzés, asszociáció, faktoranalízis stb.) állnak rendelkezésre. Itt kell még megemlíteni, hogy a klaszterelemzés, asszociáció a modellezés előkészítés mellett, sokszor lehet önmagában is az adatbányászat folyamat végeredménye, (amennyiben például a projekt célja ügyfélszegmentálás vagy keresztértékesítési lehetőségek felkutatása).

Módosítás (Modification)

Ebben szakaszban egyrészt a feltárás során szerzett tapasztalatokat felhasználva módosítjuk az adatbázisunkat: pótoljuk a hiányzó adatokat, kiszűrjük az extrém értékeket, összevonjuk vagy elhagyjuk a szükségtelen változókat, transzformáljuk a változóinkat, illetve különböző csoportosításokat, kategória besorolásokat végezhetünk a feltárás fázisban talált információk alapján (pl. fizetés alapján besoroljuk az ügyfeleinket alacsony, közepes, magas keresetű osztályokba).

Modellezés (Modelling)

Az adatbányászati folyamat legfontosabb fázisa a modellalkotás, melynek során azt a modellt keressük, amely a legjobban leírja az eredmény változó alakulását az input (magyarázó) változók függvényében. A modellezésre több lehetőséget biztosítanak a piaci

megoldások: pl. neurális hálózatok, döntési fák, idősor elemzési módszerek, statisztikai modellek (regresszió, diszkriminancia analízis). Az adatok függvényében egyik vagy másik modellezési eljárás tud pontosabb modellt alkotni, emiatt fontos, hogy az adatbányászati szoftver, mint mesterséges intelligencia, mint pedig hagyományos statisztikai eljárásokat ismerjen. Mind a neurális hálózatoknak, mind pedig a döntési fáknak meg van az az előnyük a hagyományos statisztikai (lineáris) modellezési algoritmusokkal szemben, hogy képesek nem lineáris összefüggéseket azonosítani. A neurális hálózatok általában azokban az esetekben tudják a legjobb modellt alkotni amikor az eredmény változó nagyon sok bemenő változó függvénye. A döntési fák akkor a legjobbak amikor a bemenő változók közül sok az irreveláns, és az eredmény változó viszonylag kis számú bemenő változó függvénye. A döntési fák egyik legfontosabb előnye, hogy az általuk alkotott modell könnyen értelmezhető.

Bizonyos esetekben előfordulhat, hogy a hagyományos statisztikai eszközök jobb becslést tudnak adni, mint a fent említett tanulóalgoritmusok, vagy ellenőrizhetik a mesterséges intelligencia eszközök által felépített modell helyességét stb., ezért fontos, hogy az adatbányászati termék a mesterséges intelligencia eszközökön túl hagyományos regressziós elemző eszközt (lineáris, logisztikus, nemlineáris - keresztszorzatok és magasabb fokú tagok beépítésével), illetve egyéb hagyományos statisztikai módszereket is biztosítson.

Felmérés, kiértékelés (Assessment)

Az adatbányászat célja a problémákra, kérdésekre való minél pontosabb válaszadás és ennek a válasznak minél magasabb szintű általánosítása. Az értékelési fázis, a SEMMA folyamat utolsó lépése egy keretet biztosít a különböző modellek összehasonlítására.

Az adatbányászat célja általában az adatokban megtalált rejtett információ általánosítása és felhasználása olyan adatokra, amelyek esetében csak a bemenő változók ismertek, a célváltozó nem. A modell felhasználása egy új adathalmazon a “scoring” nevet viseli. Az adatbányászati eszközök biztosítják az automatikus “scoring” lehetőségét.

Az adatbányászati eszközök lehetőséget biztosítanak a modellek összehasonlítására, a legjobb modell kiválasztására illetve modellek összekombinálására. Lehetőséget biztosítanak arra, hogy ugyanazon a projekten egyszerre több elemző dolgozzon. A nagy adatmennyiségen történő elemzéseknél különösen fontos grafikus megjelenítési lehetőségek legszélesebb skálájának biztosítása (2, 3 dimenziós grafikonok, diagramok).

Lineáris regresszió

A statisztikában a regressziószámítás (vagy regresszióanalízis) során két vagy több véletlen változó között fennálló kapcsolatot modellezzük. A regressziós modell tulajdonságai alapján megkülönböztethetünk lineáris és nemlineáris regressziót, az adataink alapján pedig **idősor**, **keresztmetszeti** és **panel** regresszióanalízist.

Alapfogalmak

A regressziós egyenletben a magyarázandó vagy függő más néven célváltozót (Y) a magyarázó változók vagy regresszorok (X) mint független változók segítségével magyarázzuk. A regressziós egyenletek fontos eleme a maradék (reziduum) vagy hibaváltozó (jelölése lehet: e, u, vagy gyakran ε), vagyis a modellünk által nem magyarázott rész. Ha a függő változónkat egy magyarázó változó segítségével modellezzük, akkor kétváltozós regresszióról, ha pedig több X változót is használunk, többváltozós regresszióról beszélünk. A

regresszió úgy mutatja meg két változó kapcsolatát, hogy egyben az egyik változó (függő változó) a másik változótól (független változó) való függésének mértékét is kifejezi.

Lineáris regresszió

A kétváltozós lineáris regressziós egyenlet általános formában: $Y_i = \beta_0 + \beta_1 X_1 + e_i$

Ahol a β karakter az együtthatókat vagy koefficienseket jelöli. A β0 együtthatót az egyenlet konstansának, vagy tengelymetszetének is szokták nevezni. Ez nem feltétlenül része az egyenletnek (például ha a függő változó várható értéke nulla). Bár az együtthatókat általában a görög ABC betűivel jelölik, ettől a gyakorlattól a szakirodalomban gyakran eltérnek. Elterjedt az y=a+bx+ε jelölésmód is. Ahol y: a függő változó, x: a független változó, a: az y tengely metszéspontja (intercept), b: az egyenes meredeksége (slope) –az a szám, amely megmutatja, hogy x egységnyi emelkedése hány egységnyi változást eredményez y értékében. Az egyenletet az y változó x változóra vonatkozó regressziós egyenletének nevezzük. (Hasonlóképpen leírható az x változó y-ra vonatkozó regressziója is, ekkor x fejezhető ki az y függvényében).

A többváltozós (k-változós) lineáris regressziós egyenlet általános formában: **Yi=, ahol, ha az egyenletnek van konstansa, X0=1, egyébként X0=0.**

Megjegyzés: Ha az x és y értékek közötti összefüggés nem lineáris, meg kell próbálni úgy transzformálni az értékeket, hogy az összefüggés lineárissá váljon, ha ez nem lehetséges, úgy nemlineáris regresszióval kell dolgozni.

A lineáris regresszió alkalmazásához csak annyit szükséges feltennünk, hogy a modellünk paramétereiben lineáris legyen. Ez nem követeli meg azonban az Y és X változók közötti kapcsolat linearitását. Példa: Az Yi=β0Xiβ1εi egyenlet ebben a formában paramétereiben nem lineáris, és lineáris regresszióval nem becsülhető. Loglineáris átalakítás után a következő egyenletet kapjuk: lnYi=lnβ0+β1lnXi +lnεi. Ez az egyenlet, bár nemlineáris kapcsolatot fejez ki Y és X között, paramétereiben lineáris, és így lineáris regresszióval becsülhető.

Multikollinearitás

A standard lineáris regressziós modell feltételezi, hogy a magyarázó változók egymástól lineárisan függetlenek. Ha azonban valamelyik magyarázó változó kifejezhető a többi tényező lineáris kombinációjaként (azaz függvényszerű kapcsolatban áll a többi magyarázó változóval) akkor multikollinearitásról beszélünk.

Ha a magyarázó változók lineárisan nem függetlenek:

- A becslés és az előrejelzés torzított marad
- A regressziós együtthatók standard hibái nőnek
- A becsléseink bizonytalanná válnak
- Az egyes magyarázó változók hatásainak elkülönítése nem lehetséges

Autokorreláció

A hibatagok lineárisan nem függetlenek.

Az autokorreláció különböző rendű lehet, attól függően, hogy a hibatag *i*-edik értéke melyik értékkel van kapcsolatban. Ha a hibatag *i*-edik értéke közvetlenül az előtte lévő értékkel áll korrelációs kapcsolatban, akkor *elsőrendű autokorreláció*-ról beszélünk. Az elsőrendű autokorreláció modellje: $\varepsilon_i = \rho \cdot \varepsilon_{i-1} + \lambda_i$

7. Előadás

Klaszterezés (felügyelet nélküli) <-> Osztályozás (felügyelt)

A klaszterezéstől az osztályozás feladatát az különbözteti meg, hogy az osztályozás felügyelt (supervised), a klaszterezés pedig felügyelet nélküli (unsupervised) csoportosítás. Osztályozás esetén a választható osztályok valamilyen ábrázolással, mintával vagy modellel leírva előre adottak. Klaszterezés esetén előzetesen megadott osztályok nincsenek, az adatok maguk alakítják ki a klasztereket, azok határait.

Egy klaszterezési feladat megoldásához ismernünk kell a fellelhető algoritmusok lényeges tulajdonságait, illetve szükségünk van az eredményként kapott klaszterezés kiértékelésére, jóságának mérésére.

Algoritmus tulajdonságok

- **robosztusság:**
  ne legyen zajos adatokra érzékeny
- **skálázhatóság:**
  az algoritmus tár és idő szükséglete agy adathalmazon, illetve magas dimenziós adatokon alkalmazva is kezelhető maradjon.
- **sorrend függetlenség:**
  ne függjön az adatpontok beolvasási sorrendjétől
- **adattípusok:**
  többfajta adattípusra is működjön (numerikus, bináris, kategorikus és ezek keverékei).

3 elvárható tulajdonság egy klaszterezőtől

- **konzisztencia:**
  Tekintsük egy adott adathalmazon alkalmazott klaszterez_o algoritmus eredményét. Ha ezután a pontok közötti távolságokat úgy változtatjuk meg, hogy tetszőleges,azonos csoportban lévő elempárok között a távolság nem nő, illetve különböző csoportban lévő elempárok közötti távolság nem csökken, akkor az újonnan kapott távolságokra alkalmazott algoritmus az eredetivel megegyező csoportosítást adja.
- **skálafüggetlenség:**
  Ha minden elempár távolsága helyett annak az α-szorosát vesszük alapul (ahol $\alpha > 0$), akkor a klaszterező eljárás eredménye ne változzon meg.
- **gazdagság:**
  Tetszőleges előre megadott klaszterezéshez létezzen olyan távolságmátrix, hogy a klaszterező eljárás az adott előre megadott partícióra vezessen.

Kleinberg tétel:

Nem létezik olyan klaszterező eljárás, ami egyszerre rendelkezik a skálafüggetlenség, gazdagság és konzisztencia tulajdonságokkal.

Távolság függvények

- eukledeszi: $d_{ij} = \sqrt{\sum_{k=0}^{n}(x_{ik} - x_{jk})^2}$
- Manhattan: $d_{ij} = \sum_{k=1}^{n} |x_{ik} - x_{jk}|$
- Minkowski metrika: $d_{ij} = (\sum_{k=1}^{n} |x_{ik} - x_{jk}|^p)^{1/p}$
- bináris:
  - $s_{ij} = \frac{|\{k|x_{ik}=x_{jk}\}|}{n}$
  - $s_{ij} = \frac{|\{k|x_{ik}\neq x_{jk}\}|}{n}$
  - d=1-s
  - Bináris (0-1) értékű attribútumok esetén, amikor a hasonlóság szempontjából csak az 1 értéket felvett attribútumok megegyezése lényeges.

Jaccard hasonlóság

Kettőnél több értékű kategorikus attribútumok átírhatók bináris attribútumokká úgy, hogy az attribútum minden lehetséges értékéhez hozzárendelünk egy bináris változót, amely 1 lesz, ha az attribútum felveszi az adott értéket, és 0 lesz különben.

Klaszterezés

k-közép

Feltesszük, hogy az adatpontok egy vektortérben helyezkednek el. Ekkor a klasztereket súlypontjukkal, középpontjukkal reprezentáljuk, azaz az adott klasztert az adatpontjaihoz tartozó vektorok átlagával reprezentáljuk. Az algoritmus olyan C klaszterezést keres, ahol az adatpontoknak a klaszterük $r(C_i)$ reprezentánsától mért

$$E(C) = \sum_{i=1}^{k} \sum_{u \in C_i} d(u, r(C_i))^2$$

négyzetes távolságösszege minimális. Egy adott C klaszterezés esetén az E(C) mennyiségre a későbbiekben a klaszterezés hibájaként fogunk hivatkozni.

Az algoritmus menete a következő. Legyen k a felhasználó által előre megadott klaszterszám. Ezután k darab tetszőleges elemet választunk az adatpontok vektorterében (célszerűen azok konvex burkában). Az iteráció első fázisában ezek a pontok reprezentálják a kezdeti k darab klasztert. Ezután minden adatpontot a legközelebbi reprezentánssal rendelkező klaszterhez sorolunk be. A besorolás során kialakult klaszterek új reprezentáns pontjai az új klaszterek középpontjai lesznek. A besorolás, középpont választás lépéseket addig iteráljuk, amíg a reprezentánsok rendszere változik. Akkor állunk meg, amikor a klaszterek elemei, illetve középpontjai már nem változnak meg az iterációs lépéstől.

Jelölje D az adatpontok halmazát egy euklideszi vektortérben. Legyen k az előre adott klaszterszám. A k-közép algoritmus pszeudokódja a következő:

$\{r(C_1), r(C_2), \ldots, r(C_k)\} \leftarrow$ reprezentánsok tetszőleges k elemű kezdeti halmaza
while az $r(C_i)$ reprezentánsok rendszere változik
  do for $i \leftarrow 1$ **to** k
    do $r(C_i) \leftarrow \frac{1}{|C_i|} \sum_{u \in C_i} u$
  for minden $u \in D$
    do u legyen az $\text{argmin}_i\, d(u, r(C_i))$ indexű klaszterben
  $C \leftarrow$ az új klaszterezés
return C

- hierarchikus
- sűrűség alapú
- spektrál
- grid alapú

Hierarchikus klaszterezés

A hierarchikus klaszterezési módszereket két csoportra bonthatjuk: a felhalmozó (egyesítő,**bottom-up**) és *lebontó* (felosztó, **top-down**) módszerek. A lentről felfelé építkező *felhalmozó* eljárásoknál kezdetben minden elem különálló klaszter, majd az eljárás a legközelebbi klasztereket egyesíti, a hierarchiában egy szinttel feljebb újabb klasztert alakítva ki. A fentről lefelé építkező lebontó módszerek fordítva működnek: egyetlen, minden adatpontot tartalmazó klaszterből indulnak ki, amit kisebb klaszterekre particionálnak, majd ezeket is tovább bontják.

- **Egyszerű összekapcsolás (single-link)**
  - Klaszterek közötti távolság: Két klaszter legközelebbi elempárosának távolsága
  - $D(C_i, C_j) = min(u, v), u \in C_i, v \in C_j$
- **Teljes kapcsolódás (complete-link)**
  - Klaszterek közötti távolság: Az elempárok közötti legnagyobb távolság
  - $D(C_i, C_j) = max(u, v), u \in C_i, v \in C_j$
  - Tehát megnézzük az összes klaszter közötti távolságot (távolság mérték fent definiált) és ezen távolságok minimuma „mentén” vonunk össze. Ez csak a távolság mértékének számításában tér el a single-linktől.
- **Súlypont módszer (centroid kapcsolódás)**
  - Klaszterek közötti távolság: A csoportbeli elemek súlypontjai közötti távolság
- **Medoid kapcsolódás**
  - A k-medoid módszer a klasztereket egy-egy, az adott klaszter elemeihez legközelebbi adatponttal, a medoiddal reprezentálja.
  - Súlypont módszertől eltérés: nem súlypontot számolunk, hanem az adott klaszteren belül lévő pontok közül választunk egyet, ami a többitől vett távolságokat szummázva a legkisebb értéket adja.
- **Ward kapcsolódás**
  - Úgy csoportosít, hogy a csoportok súlypontja és a csoport elemei közötti távolságösszeg minimális legyen
  - $D(C_i, C_j) = \sum_{u,v \in C_i, C_j} d^2(u, v) = \sum_{u,v \in C_i} d^2(u, v) - \sum_{u,v \in C_j} d^2(u, v)$

- **DENDOGRAM**
  - Hierarchikus klaszterezés eredményének bemutatását szolgáló fa, ahol nyomon követhető a klaszterek egymásba olvadása egészen a gyökérig (ahol az összes elem 1 klaszterbe kerül)

Egyszerű összekapcsolás (single-link)

$$D(C_i, C_j) = min(u, v), u \in C_i, v \in C_j$$

Bottom-up módszer, az összes távolság (A-B pontpárok) közül kiválasztja a legkisebbet, és ezek mentén hoz létre új klasztert. Legyen A,B,C,…Z pont, legyen A-B a legkisebb távolság az összes távolság közül (A-B,A-C,…B-C,…Y-Z), akkor A,B ből csináljunk egy klasztert. Iterálva csináljuk mindaddig, amíg az előre megadott klaszterszámot el nem érjük. **Fontos: AB egy klaszterbe sorolása után NEM számoljuk ki a súlyozott átlagukat (az a súlypont módszer).**

Többi hierarchikus ugyanígy megy, csak két klaszter távolságának „definíciója” különböző.

Sűrűség alapú klaszterezés

Egy sűrűség alapú klaszterező módszer a következő elemekből épül fel. Először az adatpontokon megadunk egy reflexív és szimmetrikus szomszédsági relációt. Ez a reláció meghatározza az adatpontok környezetét. A gyakorlatban a relációt úgy választjuk meg, hogy az minél jobban igazodjon a környezet esetleg előre adott intuitív fogalmához. Második lépésben lokális tulajdonságok alapján minden adatpontról eldöntjük, hogy az egy klaszter belső pontja lesz-e, azaz részt vesz-e a klaszter kialakításában, kohéziójában. A belső pontok meghatározása független lehet a korábban definiált szomszédsági relációtól.

A következő lépésben a belső pontokra bevezetjük a szomszédsági reláció tranzitív lezártját. Ez az új ekvivalencia reláció alakítja ki a belső pontok partícióját. A többi, nem belső adatpontot a szomszédsági reláció segítségével (nem feltétlenül egyértelműen és teljesen) hozzárendeljük a klaszterekhez, mint azok határpontjai. A kimaradt adatpontok adják a kívülállók halmazát.

Egy metrikus térbeli ponthalmaz esetén adott sugarú gömbbel értelmezhetjük a pontok környezetét, és egy küszöbnél nagyobb számosságú környezettel rendelkező pontokat tekinthetjük belső pontoknak. Az így kapott sűrűség alapú módszer előnye robusztussága az adatpontok (pontosabban távolságaik) perturbációjával szemben. Az algoritmus időkomplexitása lényegesen függ a környezetek generálásától, például az itt szubrutinként használható legközelebbi adatpontot kiválasztó operáció megvalósításától.

DBScan

Paraméterek: ϵ, Nmin.

$$szomszed(p) \Rightarrow d(d, d_i) \leq \epsilon$$

1. q sűrűség alapon **közvetlenül elérhető** a p-ből, ha $szomszed(p) \geq N_{min}$és $q \in szomszed(p)$
2. q sűrűség alapon **elérhető** a p-ből, ha p1,p2,...,pn ahol pi+1 közvetlenül összekötött pi-vel: p - - - - - - -> q

Tehát, fogjuk a kezdeti pontokat és két pontot „összekötünk” akkor, ha távolságuk ϵ-on belül van. Az „összekötött” részgráfok közül azokat nevezzük klaszternek, aminek legalább N_{min} eleme van. A maradék (N_{min}-nél kisebb elemszámú részgráfok) pontok egyenként lesznek klaszterek. *(Kiegészítés: „Megfogalmazás NEM elég pontos, félrevezető olyan értelemben, hogy 1 vonalon levő sűrű pontokat klaszternek tarthatja, holott nem azok.”)*

8. Előadás

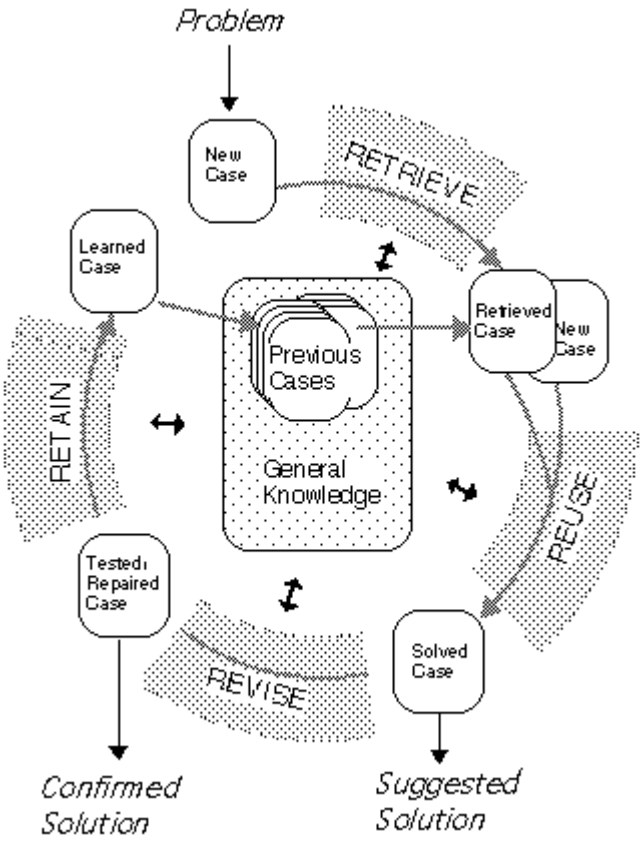
Konduktancia alapú mérték

http://www.cs.bme.hu/~bodon/magyar/adatbanyaszat/tanulmany/adatbanyaszat.pdf
(oldalszámozás alapján: 169.o – 171.o, tényleges oldalak a pdfben: 176.o – 178.o)

Case-based reasoning (CBR)

A CBR, vagyis az esetalapú következtetés, tulajdonképpen problémák a múltban átélt, tapasztalt hasonló problémák tanulsága alapján történő megoldását jelenti. A hasonlóságelemzési futtatások matematikai alapját ez a rendszermodell adja, így ennek ismertetésére Ezek a tapasztalatok, tanulságok 3 részre tagolhatóak: a problémára, amely inputként funkciónál, a megoldásra, amely az output szerepét tölti be és a megoldás megszületésének magyarázatára, amely tulajdonképpen rendszerelméleti szempontból a transzformáció.

A CBR ciklus



Tudományos körökben elfogadott nézet szerint a CBR folyamata 4 fő akcióból áll:

- **Retrieval - Visszanyerés:**
  A felmerülő problémához hasonló, releváns esetek előnyerését jelenti a memóriából
- **Reuse - Újrafelhasználás:**
  A memóriából előnyert eset leképezése az adott problémára, ami magában foglalhatja a megoldás átalakítását, amennyiben ez szükséges
- **Revise - Felülvizsgálat:**
  Az új megoldás szimulációja vagy tesztje után, a rendszer érzékenységétől függően, a megoldás elfogadása, vagy felülvizsgálatát, finomhangolását jelenti
- **Retain - Tárolás:**
  A teszt kimenetele által hordozott többletinformációk tárolása a memóriában a jövőbeli problémák megoldásához

A kinyerés

A kinyerés folyamata az esetet leíró paraméterek, jellemzők azonosításával kezdődik. A rendszer számára ismeretlen jellemzők figyelmen kívül hagyhatóak, vagy a használótól további magyarázatukra irányuló kérés lehet a megoldás. A leíró paraméterek körének lezárása után megkezdődhet a keresés az adatbázisban, amely tipikusan 2 részre tagolható. Első körben a rendszer szűkíti a lehetséges leginkább hasonló[1] esetek körét, bizonyos (előre beállított számra), majd ezekből kiválasztja a jelenlegi problémát leíró paramétereket legjobban megközelítő esetet.

[1] Hasonlóság alatt az esetek halmazokba történő rendezésekor

Az újrafelhasználás

A jelenlegi problémához leginkább hasonlító eset memóriából történő előhívása után két dolgot tehet a rendszer:

Az első esetben a jelenlegi és az visszanyert esetek különbözőségeit a rendszer absztraktnak tekinti, és így nem veszi figyelembe. Ezeknél az eseteknél a rendszer a visszanyert esetek megoldását egyszerűen átmásolja, beilleszti a megoldandó probléma kimenetének helyére.

A második esetben a jelenlegi és a visszanyert esetek különbözőségeit figyelembe veszi. A szakirodalom itt két fő lehetséges továbblépésről számol be: a transzformációs újrafelhasználásról, illetve a származtatott újrafelhasználásról. A transzformációs újrafelhasználáskor a visszanyert eset megoldását egy transzformációs operátor segítségével átalakítja a rendszer, hogy az megfeleljen a jelenlegi kihívásnak. A származtatott újrafelhasználás esetében nem a múltbéli eset megoldását veszi alapul a rendszer, hanem az ahhoz a megoldáshoz vezető folyamatot vizsgálja meg és az ott felhasznált, vagy éppen fel nem használt jellemzőkből indul ki, és ezek nyomán alakítja ki az adott problémára a megoldást.

Felülvizsgálat

A felülvizsgálati szakaszba érve a rendszer először tesztelésre bocsájtja a megoldást. Ez lehet valós életben elvégzett kísérlet, vagy egy szimulátor-program segítségével végezett próba. A próba vagy kísérlet elvégzése után a rendszer a következő, tárolási fázisába lép. A problémától függően időben hosszan is elnyúlhat, rendszer ilyenkor a részeredményeket eltárolja a esetek adatbázisában. A folyamat e szintjén kerül sor az esetleges rossz, nem várt megoldások javítására is, amely az esetek adatbázisának újabb átvizsgálását jelenti, tehát tulajdonképpen visszacsatol az újrafelhasználási fázisba.

Tárolás

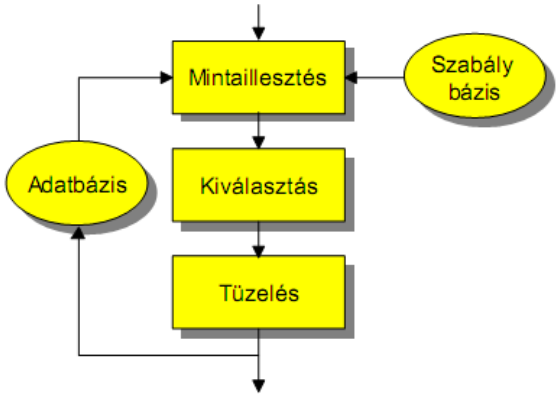
A tárolás a CBR ciklus talán legfontosabb eleme. Ebben a szakaszban ölt testet és kerül eltárolásra a memóriában az új megoldási mechanizmus. A tanulást vagy tárolást a kísérlet vagy teszt eredménye indukálja, és benne foglaltatnak az esetből megőrzendő információk kiválasztása, a tárolás formájának meghatározása, a tároláskor használt indexálás vagy kódolás életbe léptetése, valamint az új információk a meglévő memóriaszerkezetbe való integrálása. Az indexálás vagy kódolás az esetalapú következtetési rendszerek egyik kardinális kérdése, ugyanis az indexáláskor kell eldöntenie a rendszert tervezőnek, hogy milyen típusú jelzéseket használjon, a későbbi visszakeresések megkönnyítése érdekében, illetve az indexek visszakeresésének struktúráját is ki kell alakítania. A legkézenfekvőbb megoldás az összes bemeneti jel feltüntetése a jelzésekben, de léteznek ettől eltérő jelzésrendszerek is, amelyek gyorsabbá teszik az adatbázisban való keresést. A tárolási mechanizmus utolsó mozzanata az immár felcímkézett megoldás integrálása a memóriába. Amennyiben az indexáláskor nem egy új, hanem már meglévő jelzést kap az adott kimenet, abban az esetben az adatbázisban felülírásra kerül sor és a rendszer minőségi változáson megy keresztül, amennyiben viszont új jelzést kap a kimenet, abban az esetben mennyiségi a változás, így ugyanis eggyel több problémára lesz kimenet tárolva.

Szabály alapú következtetés (RBR)

A szabály-alapú következtetés (rule-based reasoning) nagyon közel áll az emberi gondolkodás bizonyos formáihoz. Gyakran döntünk *ha – akkor* típusú szabályok alapján. Ezt a gondolkodási folyamatot modellezi a szabály-alapú következtetés.

A szabály-alapú rendszerek tudásbázisa részben tényekből, részben szabályokból áll. A tények az általunk vizsgálni kívánt világ zárt leírására szolgálnak. A szabályok ezen világmodell elemeire vonatkoznak. A szabályok *feltétel-akció* párosok, ami azt jelenti: ha a feltétel kielégül akkor az akciót végre lehet hajtani. A feltétel azon körülmények kifejezésére szolgál, melyek között a szabály egyáltalán aktivizálódhat. Az akció oldal adja meg, hogy mi történjen a szabály aktivizálódása nyomán. A szabálynak kétféle hatása lehet: részben módosíthatja a tényeket, részben input/output tevékenységet generálhat.

A *következtetési ciklusnak* három fő lépése van.



A *mintaillesztés* kijelöli mindazon szabályokat, melyek feltételei illeszkednek a tényekre. Ekkor nem pusztán a feltételek logikai ellenőrzése, hanem a feltételek közt szereplő változók minden lehetséges módon való lekötése történik. Egy szabály többféle módon is illeszkedhet az adatbázisra, ezért valójában nem a szabályok tüzelnek, hanem azok példányai.

Választáskor el kell dönteni, hogy a mintaillesztés során kijelölt szabálypéldányok közül melyik tüzeljen. A feladat megoldására többféle vezérlési stratégiát alkalmazhatunk. A **leggyakrabban alkalmazott stratégiák** a következők:

- a legfrissebb tényekre illeszkedő;
- a legspecifikusabb, azaz a legtöbb feltételt alkalmazó;
- a legnagyobb prioritású, ha a szabályokhoz prioritás rendelhető, vagy
- egy véletlenszerűen kiválasztott szabály tüzel.

Egy szabály, lekötött, konkrét értékkel rendelkező változók mellett történő, végrehajtását *tüzelésnek* nevezzük. A tüzelés során módosulnak a tények, vagy input/output tevékenységek aktivizálódnak.

A **szabályok feldolgozása a következtető mechanizmusok alapján történheti:**

Szabály-alapú rendszerekben az alapvető következtetési mód az *előreláncolás* vagy adatvezérelt következtetés, melynek lényege a következő: ha a következtető mechanizmus talál olyan szabályt, melynek feltételeit a tények kielégítik, akkor végrehajthatja a szabályt, aminek hatására a tények módosulnak. Mindez ciklikusan addig folytatódik, míg van olyan szabály, amely "tüzelhet".

A következtetés másik, gyakran alkalmazott módja az ún. *hátraláncolt* vagy célvezérelt következtetés. A következtetés egy célból indul ki és megvizsgálja, hogy a rendelkezésre álló tények alapján a cél igazolható-e. A mechanizmus először a cél alapján kiválaszt egy szabályt, melynek akció oldalán a cél állítása szerepel. Ezután ellenőrzi, hogy a tények kielégítik-e ezen szabály feltételeit. Ha egy feltételre nincs illeszthető tényállítás, akkor azt az igazolandó célokhoz csatolja.

Ez a folyamat mindaddig tart, míg előáll egy megoldás, vagy kiderül, hogy az adott cél az adott tények esetén nem teljesíthető. A gyakorlatban gyakran alkalmazzák a vegyes, mind előre-, mind hátraláncoló következtetési módot. Ekkor, ha a mintaillesztés egy bizonyos tény hiánya miatt nem sikerül, akkor ezt a tényt célként lehet kitűzni és hátraláncoló szabályok segítségével meg lehet próbálni a többi tényből levezetni.

Összehasonlítás

	CBR	**RBR**
Szakterület függés	gyenge (hasonlósági fv)	erősen
Adathalmaz	nagy	kicsi
Szakértő emberi beavatkozás	nem kell	erős szakértő
Gyorsaság	illesztés: lassú következtetés: gyors	közepes
Pontosság	granuális	igen/nem
Rugalmasság	hiányzó adat mellett is tud dönteni	hiányzó adatnál nem tud dönteni

9. Előadás

Alkalmazási példák… ppt-t tessék átpörgetni, az annyi.