Írta:
FOGARASSYNÉ VATHY ÁGNES
STARKNÉ WERNER ÁGNES

INTELLIGENS ADATELEMZÉS

Egyetemi tananyag

TYPOTEX

2011

LEKTORÁLTA: Dr. Kiss Attila, Eötvös Loránd Tudományegyetem Informatikai Kar Információs Rendszerek Tanszék

TÁMOGATÁS:
Készült a TÁMOP-4.1.2-08/1/A-2009-0008 számú, „Tananyagfejlesztés mérnök informatikus, programtervező informatikus és gazdaságinformatikus képzésekhez” című projekt keretében.

ISBN 978 963 279 526 3

KÉSZÜLT: a Typotex Kiadó gondozásában

FELELŐS VEZETŐ: Votisky Zsuzsa

AZ ELEKTRONIKUS KIADÁST ELŐKÉSZÍTETTE: Gerner József

KULCSSZAVAK:

adatelemzés, egy- és többváltozós elemzés, adatvizualizáció, dimenziócsökkentés, adatbányászat, adattárházak

ÖSSZEFOGLALÁS:

Napjaink információs társadalmában a különféle célból rögzített adatok számos értékes információt rejtenek magukban. A rendelkezésre álló adatok elemzése azonban szerteágazó feladat, melynek diverzitása főként az adatok sokrétűségéből és a változatos elemzési célokból fakad. Ebből adódóan az adatokban rejlő információ feltárása rendkívül összetett folyamat, amely különféle elemzési módszereket foglal magában. Jelen jegyzet célja, hogy átfogó ismereteket nyújtson az adatelemzés főbb módszereiről és bemutassa azok alkalmazási feltételeit, lehetőségeit. Ennek megfelelően bemutatjuk az elemzést megelőző adatelőkészítési fázis fő lépéseit, szót ejtünk az alapvető matematikai és adatvizualizációs módszerekről, részletesen tárgyaljuk a leggyakrabban alkalmazott dimenziócsökkentési eljárásokat, átfogóan ismertetjük a főbb adatbányászati módszereket és kitérünk az adattárházak által nyújtott elemzési lehetőségekre. A jegyzet írása során mindvégig szem előtt tartottuk, hogy olyan tudást nyújtsunk az Olvasó számára, amely kidolgozott elméleti alapokon nyugszik, de nem nélkülözi a gyakorlati ismereteket sem.

Tartalomjegyzék

1. fejezet

Bevezetés

1.1. Motiváció

A mindennapjainkat átszövő informatikai szolgáltatások jelentős mennyiségű adatot halmoznak fel működésük során. Gondoljunk például a banki, telekommunikációs, egészségügyi, e-közigazgatási információs rendszerekre, melyek rengeteg fontos adatot tárolnak. Jóllehet az adatok tárolásának elsődleges célja a napi operatív feladatok ellátása, a működés során felgyülemlett adatok egyéb lehetőségeket is magukban rejtenek. Az adatok elemzése által olyan új ismeretekre tehetünk szert, amely információ birtokában új innovatív alkalmazásokat fejleszthetünk, megalapozott gazdasági döntéseket hozhatunk, vagy akár tudományos szempontból is hasznos felfedezéseket tehetünk.

A tárolt adatok mennyisége hatalmas, csupán azon elemzési módszereket kell megkeresnünk, amelyek elvezetnek minket a kívánt célhoz, vagyis segítségükkel az adatokat hasznos információvá alakíthatjuk. Márpedig ezen elemzési módszerek kiválasztása olykor nem is egyszerű feladat. Hiszen más elemzési módszert kell alkalmaznunk, amikor egy hipotézist szeretnénk igazolni, s más módszert kell választanunk akkor is, amikor olyan ismereteket keresünk, amelyekről nem rendelkezünk mi magunk sem előzetes információval. A megfelelő módszerek kiválasztását nehezíti továbbá a lehetőségek széles tárháza. Számos statisztikai, adatbányászati és egyéb elemzési módszer létezik, melyek megismerése szintén nem kis feladat. Jelen jegyzet elsődleges célul tűzte ki, hogy átfogó ismereteket nyújtson az adatelemzési technikákról, és betekintést adjon a fontosabbnak ítélt módszerek működésébe.

1.2. Az adatok struktúrája, típusai

Mielőtt rátérnénk a tudáskinyerés folyamatának részletes ismertetésére tekintsük át, hogy milyen jellegű adatok állnak az elemzők rendelkezésére. Mivel az elemzendő adatok számos forrásból származhatnak, ezért a megjelenési, tárolási formájuk is nagy mértékben eltérhet egymástól. Az adatok tárolása történhet strukturálatlan, félig-strukturált és strukturált módon is. Annak eldöntése, hogy az adatforrásban tárolt adatok strukturáltsága milyen szintű, nem egyszerű feladat. Egyrészt az adatokban megbújhat olyan belső struktúra, amely nincs formálisan definiálva, másrészt az első ránézésre strukturáltnak tűnő adathalmaz is tartalmazhat

strukturálatlan adatokat. Mindemellett, strukturált adatok esetében az is előfordulhat, hogy az alkalmazott struktúra valójában nem megfelelő, ezáltal nem hasznosítható az adatfeltárás folyamata során.

Strukturálatlan adatokon általában olyan elektronikus formában tárolt adatokat értünk, amelyekre nem illeszthető jól használható adatmodell. A strukturálatlan adatok legjellemzőbb előfordulási formái: videó (pl. film), audio (pl. rögzített telefonbeszélgetések), illetve hosszabb szöveges adatok (pl. blog, elektronikus könyv). Bár az adatelemzés szempontjából kétség kívül a legnehezebb feladat a strukturálatlan adatok elemzése, napjainkban már számos olyan technika létezik (pl. szövegbányászat), melyek alkalmazásával ezen adathalmazokból is hasznos információk nyerhetők ki.

A *félig-strukturált adatok* átmenetet jelentenek a strukturálatlan és a jól strukturált adatok között. Ezen adatok tárolási formájukat tekintve ugyan tartalmaznak olyan formális elemeket, amelyek elkülönítik az egyes tartalmi részeket egymástól, de ezen tagolás még távol áll a strukturált (pl. táblázatos) formában megadott reprezentációtól. Félig-strukturált adathalmaznak tekintünk például egy XML dokumentumot, ahol tagek határozzák meg a tartalom felosztását és hierarchiáját, illetve az e-maileket is, melyekben a feladó, a címzett, a tárgy, az elküldés dátuma strukturált formában kerül rögzítésre, azonban a tartalmi rész strukturálatlan formában tárolódik.

Strukturált adatok esetében az információ elemi szintű adatokra bomlik, s ezen elemi adatok kapcsolatrendszere modellek által definiált. A strukturált formában rögzített adatokon leggyakrabban táblázatokban tárolt adatokat értünk, ahol az egyes oszlopok az objektumokat leíró tulajdonságokat tartalmazzák, egy-egy sor pedig egy-egy objektumnak feleltethető meg.

Adatelemzési szempontból tekintve természetesen a strukturált adattárolási forma biztosítja az elemzési lehetőségek legszélesebb tárházát, és a legtöbb adatelemző algoritmus kizárólag strukturált formában tárolt adatokon képes dolgozni. Éppen ezért célul tűzhetjük ki, hogy az elemzésre szánt adatokat az elemzés megkezdése előtt minél strukturáltabb formára alakítsuk, konvertáljuk.

A strukturált formában (pl. relációs adatbázisban) tárolt objektumokat az őket jellemző tulajdonságok értékével definiáljuk. Az adatbázis-kezelésben gyakori elnevezéssel élve szokás ezen jellemző tulajdonságokat attribútumoknak, mezőknek, változóknak is nevezni, míg az általuk felvett értékeket pedig adatoknak hívjuk. Egy személyről tárolhatjuk például a nevét, nemét, születési dátumát, havi jövedelmét, beosztását, illetve azt, hogy rendelkezik-e gépkocsival. Ezen jellemző tulajdonságok különféle típusú és számosságú értékeket vehetnek fel az objektumok összességét tekintve. Így például amíg a havi jövedelem számos egymástól eltérő értéket vehet fel, addig annak meghatározására, hogy valaki rendelkezik-e gépkocsival két érték is elegendő.

Adatelemzési szempontból fontos momentum, hogy egy-egy attribútum milyen értékeket vehet fel, és ezen értékek hogyan viszonyulnak egymáshoz. Az objektumokat leíró tulajdonságok a következő két fő csoportba sorolhatók:

- *Folytonos változók*: A folytonos változók egy adott skálán tetszőleges értékeket vehetnek fel. A skála bármely két értéke között végtelen sok újabb érték helyezkedik el, s a folytonos típusú változók ezen értékek bármelyikét felvehetik. Folytonos típusú változónak tekintjük például a földrajzi hely hosszúsági és szélességi koordinátáit, a

testsúlyt, vagy akár a hőmérsékletet is, hiszen a pontosság mértéke tetszőlegesen növelhető.

- *Kategorikus (diszkrét) adatok*: A kategorikus változók a folytonos változókkal ellentétben a mérési skálán nem vehetnek fel tetszőleges értéket, a felvehető értékek jól elkülönülnek egymástól, köztük „rés" van. Matematikai szempontból azt mondhatjuk, hogy egy változót akkor nevezünk kategorikus változónak, ha a természetes számok egy részhalmaza és a változó által felvehető értékek között megadható egy egyértelmű leképezés. Kategorikus változónak tekintjük a személyek esetében például a nem, a foglalkozás és a lakóhely attribútumokat.

Az objektumokat leíró változókat azonban nem csak a fenti szempont szerint csoportosíthatjuk. Az objektumváltozók azon szempont szerint is különbözhetnek egymástól, hogy a tulajdonságot leíró mérték egyes értékei hogyan viszonyulnak egymáshoz. Ezen szempont alapján a következő típusú változókat különböztethetjük meg [34]:

- *Felsorolás típusú változók*: A felsorolás típusú változók olyan diszkrét értékeket vehetnek fel, mely értékek között sorrendiség nem adható meg. Egy felsorolás típusú változó két értékére vonatkozóan csupán azt állapíthatjuk meg, hogy a két érték egyenlő-e, egyéb matematikai művelet ezen változók esetében nem értelmezhető. A felsorolás típusú változók speciális esete a *bináris (logikai) változó*, mely esetében a felvehető értékek száma pontosan kettő. Felsorolás típusú változónak tekintjük például a szemszín tulajdonságot, vagy azt, hogy valaki rendelkezik-e bankkártyával vagy sem.

- *Rendezett típusú változók*: A rendezett típusú változók szintén kategorikus értékeket vehetnek fel, azonban a változók egyes értékei között sorrendiség is definiálható. Ilyen változónak tekintjük például az iskolai végzettség szintjét, amely a következő értékeket veheti fel: „nincs", „általános iskola", „középiskola", „BSc (főiskola)", „MSc (egyetem)". Látható, hogy ezen értékek között definiálható egy sorrendi viszony, amely alapján az is megadható, hogy két személy közül ki rendelkezik magasabb szintű iskolai végzettséggel.

- *Intervallumskálázott változók*: Az intervallumskálázott változók értékei között már nem csupán sorrendiség értelmezhető, hanem a változó két értékének különbsége is meghatározható. A változótípus további jellemző tulajdonsága, hogy a 0 pont megválasztása tetszőleges. Az intervallumskálázott változók tipikus példája a hőmérséklet Celsius-fokban történő megadása, mely skála pontosan mutatja be a 0 pont megválasztásának önkényességét, illetve azt, hogy a Celsius-fokban megadott hőmérsékletek különbsége egyértelműen értelmezhető és informatív.

- *Arányskálázott változók*: Az arányskálázott változók esetében a felvett értékeknek nem csupán a különbsége, hanem az egymáshoz viszonyított aránya és mérvadó. Ilyen típusú adatok esetében a 0 érték is kitüntetett szereppel bír, ez az adott tulajdonság hiányát jelöli. Arányskálázott változónak tekintjük például a személyek testsúlyát, fizetését, illetve az áramerősséget, vagy a sebességet rögzítő változókat.

Fontos kiemelni, hogy a változók típusai nem csupán a tárolt adatok jellegét határozzák meg, hanem azt is, hogy milyen műveleteket, statisztikai függvényeket értelmezhetünk rajtuk, s ezáltal meghatározzák a rajtuk futtatható algoritmusok körét is.

1.3. Adatelemzési módszerek, az adatelemzés folyamata

Számos adatelemzési módszer létezik, mellyel eljuthatunk a kívánt célhoz, vagyis meghatározhatjuk az elemzendő adatok fő jellemzőit, eloszlását, és modelleket alkotva leírhatjuk a karakterisztikájukat. Az elemzéshez használt módszerek kiválasztását több tényező is befolyásolhatja. Az alkalmazott módszereket nagy mértékben meghatározza az elemzendő adatok típusa, az elemzési cél, az elemzendő adatokra vonatkozó *a priori* ismeretek megléte, vagy hiánya, illetve, hogy milyen elemzési módszereket szokás alkalmazni az adott tudományterületen belül. Érdemes azonban megemlítenünk egy olyan filozófiát, melynek követése nagy mértékben növelheti az elemzés hatékonyságát. Ez az adatelemzési megközelítés a feltáró adatelemzés filozófiája.

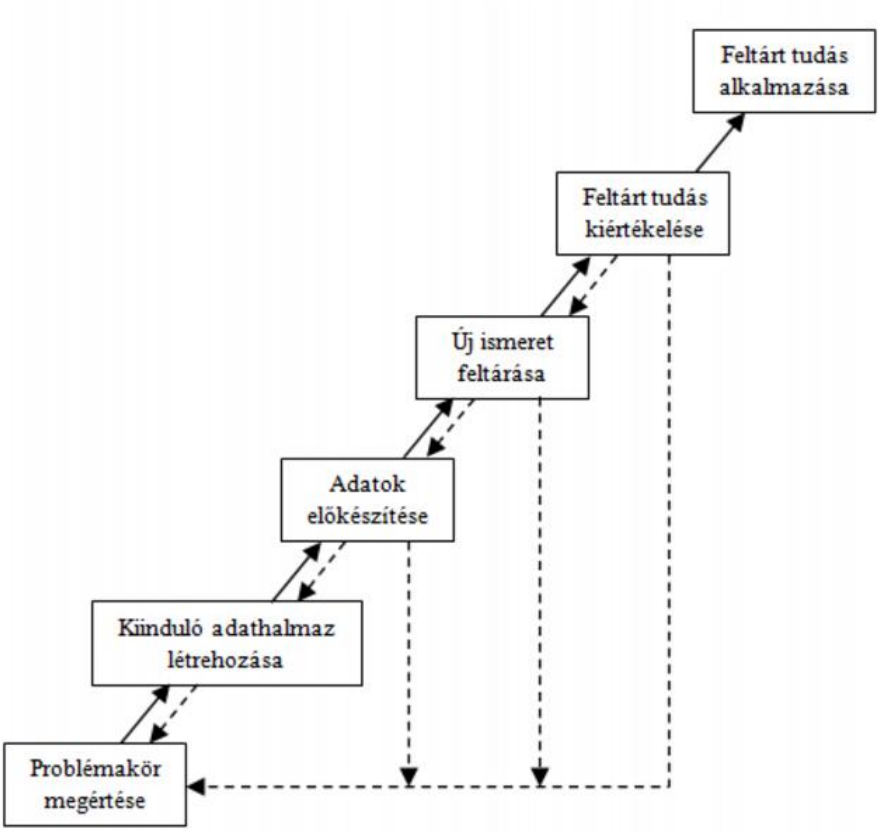
A *feltáró adatelemzés* [38] egy adatelemzési filozófia, megközelítés, mely számos különféle technikát alkalmaz azon célból, hogy:

- maximális rálátást biztosítson az adathalmaz adataira,
- feltárja az adatokban rejlő meghatározó struktúrákat,
- meghatározza az adatokra leginkább jellemző tulajdonságokat,
- rámutasson a többi adattól nagy mértékben eltérő adatokra,
- feltételezéseket teszteljen,
- és adatleíró modelleket hozzon létre.

A feltáró adatelemzés az adatokból indul ki, s gyakran alkalmaz olyan adatvizualizációs módszereket (pl. grafikonok), melyek az adatok grafikus megjelenítését szolgálják. A grafikai megjelenítés célja azon emberi képesség kiaknázása, hogy amennyiben a szó szoros értelmében betekintést nyerünk az adatokba, azok egymáshoz viszonyított elhelyezkedésébe, az őket leíró tulajdonságok értékeinek eloszlásába, akkor könnyebben felismerjük az adatokban megbúvó rejtett struktúrákat, váratlan összefüggéseket. A grafikai módszerek alkalmazása tehát jelentős előnyt nyújt azon adatelemzési módszerekhez képest, melyek ennek hiányában végzik el az ismeretek feltárását. Jelen jegyzet az imént említett filozófiát követve igyekszik minél több olyan adatvizualizációs módszert bemutatni, amelyek hatékony segítséget nyújtanak az elemzők számára a rendelkezésre álló adatok jellemző tulajdonságainak feltárásához, modelljeik megalkotásához.

Az adatok elemzését azonban nem szabad egy különálló, önálló tevékenységként elképzelni. Az adatelemzés egy olyan összetett folyamat része, melyben az adatok összegyűjtése, előkészítése, elemzése, és az elemzés kiértékelése hasonlóan fontos szerephez jut. Ez a folyamat a *tudásfeltárás folyamata*, amely a következő fő lépésekből áll:

1. a problémakör megértése,
2. a kiinduló adathalmaz létrehozása,
3. az adatok előkészítése,
4. új ismeretek feltárása,
5. a feltárt tudás kiértékelése,
6. a feltárt tudás alkalmazása.



1.1. ábra. A tudásfeltárás folyamata

Az elemzési módszertől függetlenített tudásfeltárási folyamatot az 1.1 ábra szemlélteti. Mint láthatjuk, a tudásfeltárás tevékenysége nem csupán egymást követő lineáris lépések sorozataként képzelendő el, hanem egy iteratív folyamat. Az egyes fázisok során visszacsatolások alakulhatnak ki, melyek például az elemzendő adatok kiegészítését, vagy az egyes fogalmak újradefiniálását, pontosabb megértését célozzák. A következőkben tekintsük át, hogy az egyes fázisok milyen főbb tevékenységeket foglalnak magukban.

A problémakör megértése: Az adatok elemzését, beleértve a teljes tudásfeltárás folyamatát, általában az úgynevezett tudásmérnökök végzik el. Ők azok a szakemberek, akik azon

matematikai és informatikai ismeretekkel rendelkeznek, amelyek nélkülözhetetlen feltételei a sikeres tudásfeltárásnak, azonban ezek a szakemberek az elemzendő szakterület tudását csak ritka esetben tudhatják magukénak. Gondoljunk csak arra, hogy amennyiben például ipari, vagy egészségügyi adatok elemzése a cél, akkor már maga a szaknyelv elsajátítása, a probléma megértése is nehézségeket okozhat azoknak, akik ezzel a szakterülettel korábban nem foglalkoztak. A tudásfeltárás első fázisa során az elemzést végző tudásmérnökök és az elemzendő szakterület szakértői megbeszélések során tisztázzák az alapvető fogalmakat, az adatokat jellemző tulajdonságok közti összefüggéseket, és meghatározzák az elemzés célját.

A kiinduló adathalmaz létrehozása: A második fázisban meg kell határozni, hogy milyen adatok érhetők el, használhatók fel a tudásfeltárás folyamán, az elemzési célok ismeretében milyen esetleges további adatok beszerzése szükséges, illetve melyek azok a rendelkezésre álló adatok, amelyek az elemzésben nem vesznek részt. Általánosságban elmondhatjuk, hogy az adatelemzés nem feltétlenül egyetlen adatbázis adatainak elemzését jelenti, hanem gyakran van szükség több adatforrás migrálására, összefésülésére, illetve egyéb hiányzó adatok pótlására.

Az adatok előkészítése: Az adatfeltárás folyamatának harmadik lépése meghatározó jelentőségű az eredményül kapott tudás minőségének szempontjából. Amennyiben hibás, rossz adatokon hajtjuk végre az analízist, akkor nagy valószínűséggel a levont következtetések is hibásak lesznek. Az adatelőkészítési fázis fő feladata a rendelkezésre álló adatok megtisztítása, a redundanciák megszüntetése, az adatokban megbúvó ellentmondások, inkonzisztenciák feloldása, s amennyiben lehetséges, akkor a hiányzó értékek pótlása. Az adatelőkészítés fő feladatait részletesebben az 1.4 fejezetben mutatjuk be.

Az új ismeretek feltárása: Gyakran szokás ezt a fázist elemzési fázisnak is nevezni. Az alkalmazandó elemzési módszer kiválasztását nagy mértékben meghatározza a rendelkezésre álló adatok mennyisége, minősége, a problématerület jellege és az elemzési cél. A leginkább elterjedt adatelemzési technikák statisztikai, adatbányászati, adattárház-elemző módszereket, illetve ezek együttes alkalmazását foglalják magukban. Mint már korábban említettük, ezen elemzési technikákat adatvizualizációs módszerekkel kiegészítve még hatékonyabban alkalmazhatjuk a rendelkezésünkre álló eszközöket.

A feltárt tudás kiértékelése: A tudásmérnökök által feltárt ismeretek értékelése a szakterület szakértőinek a feladata. A feltárt tudás szakértők felé történő prezentálásában újfent jelentős szerephez juthatnak a különféle adatvizualizációs eszközök. A szakértők feladata meghatározni, hogy a feltárt ismeretek, újak-e, hasznosak-e, nem tartalmaznak-e trivialitásokat, illetve ellentmondásokat, valamint hogyan illeszthetők be a korábbi ismeretek rendszerébe.

A feltárt tudás alkalmazása: Amennyiben a feltárt tudás valóban új ismereteket tartalmaz, s ezek hasznosak, akkor a továbblépés első fázisa ezen ismeretek alkalmazási területeinek feltárása, majd beépítése a mindennapi gyakorlatba.

Mint korábban említettük, az adatelőkészítési fázis kiemelt szereppel bír az elemzési eredmény minőségének tekintetében. A következő fejezet ezen fázis fő tevékenységeit mutatja be.

1.4. Az adatok előkészítése

A rendelkezésre álló adatok előkészítése a tudásfeltárás folyamatának egy rendkívül fontos lépése. Gondoljunk csak arra, hogy például egy mérőeszköz meghibásodása hibás adatrögzítést eredményezhet, s amennyiben az elemző algoritmusok a hibás adatokat dolgozzák fel, akkor az eredményül kapott következtetések is nagy valószínűséggel hibásak lesznek. Az adatok előkészítése azonban nem csak a hibás adatok javítását jelenti, hanem számos egyéb feladatot is magába foglal. Miután tapasztalati tény, hogy az adatelőkészítési fázis a tudásfeltárás teljes folyamatának időben akár 60-70%-át is kiteheti, ezért vessünk mi is egy rövid pillantást a megoldandó problémák körére.

Az adatelőkészítési fázis a következő két fő gondolatkört foglalja magában: (1) az adatok megtisztítása azon célból, hogy ne tartalmazzanak hibás, téves értékeket, illetve (2) az adatok átalakítása az elemzési szempontok és algoritmusok figyelembe vételével. Ezen második problémakör megoldása feltételezi az elemzési célok pontos megfogalmazását, illetve azt, hogy az elemzést végző szakember már részben döntést hozzon az alkalmazandó elemzési módszerekre és algoritmusokra vonatkozóan, hiszen csupán ezen ismeretek birtokában tudja meghatározni, hogy a rendelkezésre álló adatokat milyen formára kell transzformálni.

Az adatelőkészítési fázis fő feladatai a következőképpen foglalhatók össze:

- *Adatintegráció*: Az elemzéshez használt adatok számos forrásból származhatnak (pl. különféle információs rendszerek, flat fileok, Excel táblázatok). Az adatintegráció célja ezen adatok egységes rendszerbe (általában egy adatbázisba) történő összegyűjtése, integrálása. Az adatok egyesítése során azonban különféle gondok merülhetnek fel: (1) Gyakori probléma, hogy az egyesítendő adatforrások különféle sémában tárolják az adatokat. Ekkor az elemző feladata, hogy ezen adatsémákat összefésülje, és kialakítson egy egységes sémát (pl. relációs adatbázisrendszert), amely az összes rendelkezésre álló adat tárolására alkalmas, majd az adatokat ezen sémába importálja. (2) A különféle rendszerekben tárolt adatok a tárolt információk tekintetében számos ellentmondást tartalmazhatnak. Előfordulhat például, hogy a hőmérséklet adatok az egyik adatforrásban Celsius-fokban, míg a másikban Kelvin-fokban kerültek tárolásra. A migráció feladata ezen adattárolási konfliktusok detektálása és feloldása. (3) Amennyiben az adatok több forrásból származnak, akkor gyakran előfordul, hogy ugyanazon adat mindkét adatforrásban tárolásra került. Az adatintegráció során az elemző feladata a redundáns adattárolás megszüntetése, különös tekintettel a redundánsan tárolt, de egymásnak ellentmondó értékek problémájának kezelésére. Ilyen probléma lehet például, ha egy személyre vonatkozóan az egyik adatbázisból az olvasható ki, hogy a gyermekeinek száma 1, míg a másikban ez a jellemző tulajdonság 2-es értéket tartalmaz. Az ilyen jellegű ellentmondások feloldása gyakran nagyon időigényes, hiszen további utánajárást igényel. Egyszerűbb megoldást jelenthet ezen értékek együttes törlése, ez azonban adatvesztést eredményez.

- *Adattisztítás*: Az adattisztítás célja a hibás, inkonzisztens adatok javítása, a kiugró értékek azonosítása és szükség szerinti javítása, illetve a hiányzó értékek pótlása. Az adathibák leggyakoribb forrása az emberi tévesztés, illetve a rögzítő eszköz hibás működése. Adathiány általában a rögzítő eszköz működési zavarából, törlésből, illetve azon

okból alakulhat ki, hogy az adott adat a rögzítés során nem tűnt fontosnak (vagy például nem volt olvasható), ezért nem került rögzítésre. Az adathibákat legkönnyebben az adatbázison futtatott lekérdezések által, illetve statisztikai módszerekkel tárhatjuk fel. Megnézhetjük például az adott változó értékeinek eloszlását, s ennek ismeretében a gyanúsnak minősülő (például értékében nagyon kiugró) adatokat manuálisan ellenőrizhetjük. A hiányzó adatok esetében megoldást jelenthet az adatsor törlése (bár ez csökkenti a rendelkezésre álló adatok számosságát), az adatok manuális pótlása, globális konstansok bevezetése (pl. „ismeretlen”), illetve az értékek kitöltése valamely formula alapján. Ez utóbbi módszer kivitelezhető például adott minta középértékének beírásával, vagy következtetés alapú formula (pl. származtatott attribútum, döntési fa, regresszió) használatával.

- *Adattranszformáció*: Az adattranszformáció rendkívül sokrétű feladat, mely számos célt takarhat. Az átalakítások során az adatokat olyan módon transzformáljuk, hogy azok megfelelőek legyenek az alkalmazandó algoritmusok számára, és hatékony adatelemzést tegyenek lehetővé. Gyakori adattranszformációs megoldás például az új változók bevezetése, a meglévő változók normalizálása, illetve a folytonos értékek kategorikus adatokká történő konvertálása. A változók normalizálása kiemelendő feladat, hiszen azáltal, hogy az eltérő tulajdonságokat leíró változókat azonos terjedelmű értéktartományra konvertáljuk elkerülhetjük azt, hogy az eredetileg nagyobb skálán mozgó adatok nagyobb befolyással rendelkezzenek bizonyos adatelemzési módszerek esetén (pl. csoportosítási feladatok).

- *Adatredukció*: Az adatredukció célja olyan kisebb adathalmaz létrehozása, amely ugyanahhoz az elemzési eredményhez vezet. Az adatredukció igénye származhat például a rendelkezésre álló adatok túl nagy méretéből adódóan, melynek elemzése redukció nélkül túlságosan időigényes lenne. A vizsgált objektumok számosságának csökkentéséhez például a különféle mintavételezési technikák, vagy csoportosítási algoritmusok alkalmazása nyújthat segítséget. Az adatredukciós eljárások másik fő típusa az objektumokat leíró jellemző tulajdonságok számosságának csökkentése. Ez történhet oly módon, hogy a kevésbé fontos tulajdonságokat elhagyjuk, illetve oly módon is, hogy a rendelkezésünkre álló tulajdonságok összességéből újabb, kevesebb számú jellemző tulajdonságokat hozunk létre. Ezen leggyakrabban alkalmazott dimenziócsökkentési eljárások részletes ismertetése a 3. fejezetben található.

Láthatjuk tehát, hogy az adatelőkészítés rendkívül szerteágazó feladatkör. A feladat fontosságából adódóan számos adatelemzésre használt programcsomag tartalmaz adatelőkészítést támogató eljárásokat, algoritmusokat. Miután jelen jegyzetnek nem célja az adatelőkészítési technikák részletes bemutatása, ezért a fentiekben csupán vázoltuk a főbb feladatokat. Az adatelőkészítés során alkalmazott gyakoribb algoritmusokról bővebb ismereteket a [13] irodalomban talál a kedves Olvasó.

Miután áttekintettük a rendelkezésre álló adatok típusait és az ismeretfeltárás folyamatát, a továbbiakban az elemzési technikák részletes bemutatása következik. A 2. fejezetben bemutatjuk az adatbázisok elemzése során leggyakrabban alkalmazott alapvető statisztikai

és adatvizualizációs módszereket, a 3. fejezet pedig a főbb dimenziócsökkentési eljárások ismertetését tartalmazza. Az adatbányászat fő területeinek ismertetése és a leggyakrabban alkalmazott algoritmusok bemutatása a 4. fejezetben található. Az 5. fejezetben egy speciális adatelemzési módszert, az adattárházak alkalmazását mutatjuk be.

2. fejezet

Alapvető matematikai és adatvizualizációs módszerek

A tudásfeltárás folyamatában az adatok előkészítése és a tényleges elemzési fázis nem határolható el diszkréten egymástól. Mindamellett, hogy a két lépés között folyamatos a visszacsatolás, az adatelőkészítési fázisnak már önmagában is része bizonyos adatfeltáró, elemző tevékenység. Ezen elemzések által az elemzést végző szakemberek részletesebb rálátást nyernek az elemzendő adatok jellemző tulajdonságaira, illetve ezeknek az ismereteknek a birtokában készítik elő az adatokat az alkalmazandó algoritmusok futtatásához.

Jelen fejezet célja azon statisztikai és adatvizualizációs eszközök bemutatása, amelyek gyakran használatosak a strukturált formában tárolt adatok vizsgálata során. Ezek a módszerek hatékony segítséget nyújtanak a tudásmérnökök számára az elemzendő adatok főbb karakterisztikájának megállapításában, és nélkülözhetetlenek az adatok előkészítési fázisában. A fejezetben a továbbiakban feltételezzük, hogy a vizsgált adatok relációs adatbázisban állnak az elemzők rendelkezésére. Az egyes módszerek bemutatásakor itt és a továbbiakban is gyakran fogjuk segítségül hívni az adatbányászatban közismert „iris adathalmazt”. Ez az adathalmaz 150 db iris virág 4 jellemző tulajdonságát tartalmazza, melyek a következők: csészelevél hossza, csészelevél szélessége, sziromlevél hossza, sziromlevél szélessége. A 4 jellemző tulajdonság mellett mind a 150 virágról ismert az alfaja is (Iris Setosa, Iris Versicolour, Iris Virginica). Az adathalmaz a mellékletben `iris.txt` néven érhető el.

2.1. Egyváltozós elemzés

Az egyváltozós vizsgálatok során az elemzés célja valamely kiválasztott változó (attribútum, jellemző) vizsgálata függetlenül a többi változó értékétől. Az egyváltozós elemzés jellemzően az első lépések egyike, amely a rendelkezésre álló adatok karakterisztikájának feltárásához vezet.

2.1.1. Szélső- és középértékek, szórás

Egy adott attribútum által felvett értékek vizsgálatakor az első lépés annak megállapítása, hogy az adott attribútum értékei megfelelnek-e az attribútumra előzetesen definiált korláto-

zásoknak (pl. felvehető értékek korlátozása, karakterek maximális száma), és milyen terjedelemben mozognak. Relációs adatbázisban tárolt adatok esetén ezen kérdések SQL lekérdezések segítségével könnyen megválaszolhatóak. Az adathalmaz terjedelmére vonatkozó kérdés azonban csupán rendezett, intervallumskálázott és arányskálázott attribútumok esetén vizsgálható, mivel a felsorolás típusú adatok estén az értékek között nem értelmezhető sorrendiség.

A változó terjedelmének vizsgálatához tekintsünk egy x attribútumot, mely N db értéket vesz fel. Az x attribútum által felvett értékek a következők: $x_1, x_2, \ldots, x_N$. Az *attribútum minimális* és *maximális értékét* a 2.1 és 2.2 képletek definiálják:

$$x_{min} = x_i, \text{ahol } x_i \leq x_k, \forall i,k \in 1,2,\ldots,N \tag{2.1}$$

$$x_{max} = x_j, \text{ahol } x_j \geq x_l, \forall j,l \in 1,2,\ldots,N \tag{2.2}$$

Az *attribútum terjedelme* a minimális és maximális értékek ismeretében a következőképpen határozható meg:

$$T_x = x_{max} - x_{min} \tag{2.3}$$

A minimális és maximális értékek, illetve az attribútum terjedelmének kiszámítása relációs adatbázisban könnyen elvégezhető az SQL nyelv beépített függvényei segítségével. Az alábbi példa a *dolgozo* táblában tárolt alkalmazottak minimum és maximum *fizetését*, illetve ezen tulajdonság terjedelmét számolja ki:

```
SELECT min(fizetes) AS minimum, max(fizetes) AS maximum,
max(fizetes)-min(fizetes) AS terjedelem
FROM dolgozo;
```

Míg a minimum és a maximum értékek fontos adathibákra (pl. tizedesjegyek téves megadása) hívhatják fel az elemzők figyelmét, addig az attribútum terjedelme önmagában még nehezen értelmezhető, nagysága kevésbé informatív. Azt azonban kijelenthetjük, hogyha a változó terjedelme 0, akkor az azt jelenti, hogy az attribútum a teljes adathalmaz esetében ugyanazt az értéket veszi fel, tehát a további elemzések során ezen változót biztosan kihagyhatjuk az elemzésből. Megjegyezzük, hogy hasonló következtetést vonhatunk le abban az esetben is, ha az attribútum által felvett különböző értékek számosságát vizsgáljuk meg (`SELECT DISTINCT`). Ha ez az érték 1, akkor az attribútumot a további elemzések során nem kell figyelembe vennünk. Ez utóbbi módszer szélesebb körben alkalmazható, mint a terjedelem vizsgálata, hiszen felsorolás és rendezett típusú attribútumok esetén szintén értelmezhető.

Ahhoz, hogy kissé több információt nyerjünk a vizsgált változóra vonatkozóan, érdemes a változó által felvett értékek középértékét, vagyis az *átlagát* kiszámítani. Az adatok átlaga felsorolás és rendezett típusú változók esetén nem értelmezhető. Folytonos értékeket felvevő változók esetén az adatelemzések során a változó átlaga alatt a változó által felvett értékek számtani átlagát értjük, melyet a 2.4 képlet definiál. A folytonos típusú attribútum átlaga az SQL nyelv beépített `AVG` függvénye segítségével szintén könnyen kiszámítható.

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{2.4}$$

Az attribútum minimumának, maximumának és átlagának ismeretében a terjedelem is informatívabbá válik. Amennyiben az attribútum terjedelme nagy, és az átlag értéke valamely szélsőértékhez (minimum, maximum) közel esik, akkor érdemes figyelmet fordítani a másik szélsőérték és a hozzá közel eső adatok vizsgálatára. Miután a terjedelem érzékeny az úgynevezett outlier adatokra, vagyis azokra az adatokra amelyek nagy mértékben eltérnek a többi adattól, ezért ezekben az esetekben a vizsgált attribútum nagy valószínűséggel outlier értéket is tartalmaz. Az ilyen outlier adatok származhatnak akár hibás adatrögzítésből is, azonban amennyiben ténylegesen valós adatot takarnak, akkor érdekes esetekre hívhatják fel az elemzők figyelmét. Meg kell azonban jegyeznünk, hogy egyetlen érték kiugrása önmagában nem feltétlen jelent az átlagostól eltérő esetet, hiszen egy objektumot általában több attribútum együttesen jellemez. Az attribútumérték ilyen jellegű eltérése önmagában csupán figyelemfelhívó szereppel bír, pontosabb elemzési lehetőséget ezen kérdésben az adatok csoportosítása nyújthat.

Az attribútum értékeinek átlaga mellett további információt adhat az elemző számára az értékek mediánjának és móduszának kiszámítása. A *medián (Me)* olyan helyzeti középérték, amely értéknél ugyanannyi kisebb és ugyanannyi nagyobb értéket vesz fel az attribútum. Úgy is mondhatjuk, hogy a medián az attribútum értékeinek felezőpontja, a nála nagyobb és nála kisebb értékek gyakorisága azonos. Mivel a medián kiszámítása ugyancsak feltételezi a változó értékei között értelmezhető sorrendiség meglétét, ezért ezen érték rendezett, intervallum- és arányskálán mért értékek esetén adható meg. A medián, ellentétben az átlaggal, nem érzékeny az outlier adatokra, ezért kiszámítása elsősorban aszimmetrikus eloszlások esetében hasznos. Az attribútumok mediánjának kiszámításához számos SQL implementáció (Pl. Oracle 10g) tartalmaz beépített függvényt (`MEDIAN`).

Egy adott *változó módusza* a változó által leggyakrabban felvett értéket definiálja. Ezen mérőszám már értelmezhető felsorolás típusú attribútumok esetén is, s jellemzően kategorikus változók jellemzésére használatos. Amennyiben a mintában minden érték azonos gyakorisággal fordul elő, akkor a módusz értékét nem lehet meghatározni. A módusz értéke egyéb esetekben sem feltétlenül egyértelmű, mivel több különböző attribútumérték is előfordulhat ugyanolyan maximális gyakorisággal.

Az attribútum által felvett értékek minimuma és maximuma mintegy keretbe foglalja az adatokat, a medián pedig elfelezi őket. Részletesebb rálátást nyerhetünk az adatokra oly módon, hogyha az attribútum által felvett értékeket nem csupán 2 tartományra (minimum-medián és medián-maximum) osztjuk, hanem több kisebb, egyenlő számosságú csoportot határozunk meg. A *kvantilis értékek* a vizsgált adatok azon pontjai, amelyek az értékeket egyenlő számosságú részhalmazokra osztják fel. A kvantilis értékek meghatározása oly módon történik, hogy az adatokat sorba rendezzük, majd k db egyenlő számosságú részhalmazra osztjuk fel őket. A halmaz i. k-ad rendű kvantilise az a szám, amelynél az adatok i/k-ad része kisebb és $(1-i/k)$-ad része nagyobb. A gyakorlatban használt nevezetes kvantilis értékek a következők:

- *Medián*(Me): $k=2$ estén az adatokat 2 részre osztó kvantilis, amely érték alatt és felett ugyanannyi adat helyezkedik el.

- *Kvartilisek*: $k=4$ esetén az adathalmazt 4 egyenlő részre osztjuk. Az adatok 25%-a kisebb, mint az alsó kvartilis (Q_1). A második kvartilis a medián (Q_2), melynek értéke

alatt az adatok 50%-a helyezkedik el. A harmadik kvartilis a felső kvartilis (Q_3), mely érték alatt az adatok 75%-a, felette pedig az adatok 25%-a található.

- *Kvintilisek*: A $k = 5$ eset kvantilisei ($Q_1 - Q_4$), melyek az adatokat 5 egyenlő részhalmazra osztják.
- *Decilisek*: A $k = 10$ eset kvantilisei ($Q_1 - Q_9$), melyek az adathalmazt 10 részre osztják.
- *Percentilisek*: Ez a felosztás megfelel a hagyományos százalékos felosztásnak, ahol az adathalmazt a percentilisek 100 egyenlő számosságú részre tagolják ($k = 100$).

Az adatok vizsgálata során az attribútumértékek csoportosulása mellett az adatok egymástól való eltérésének vizsgálata is fontos szerephez jut. Az attribútumértékek egymástól való eltéréseit, szóródását a különféle szórásmutatókkal vizsgáljuk. A statisztikában különféle mérőszámok használatosak az adatok varianciájának vizsgálatára, melyek közül leggyakrabban a *szórás* és ennek négyzete, a *szórásnégyzet* használatos. A *tapasztalati (empirikus) szórásnégyzet* az adatok átlagtól vett eltérésnégyzetének átlagát adja meg, melyet a következő képlet definiál:

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(\bar{x} - x_i)^2 \tag{2.5}$$

A 2.5 egyenletben definiált empirikus szórásnégyzet azonban a minta nem torzítatlan becslése, ezért helyette gyakran használatos a *korrigált tapasztalati szórásnégyzet*, ahol a nevezőben N helyett $N-1$ szerepel.

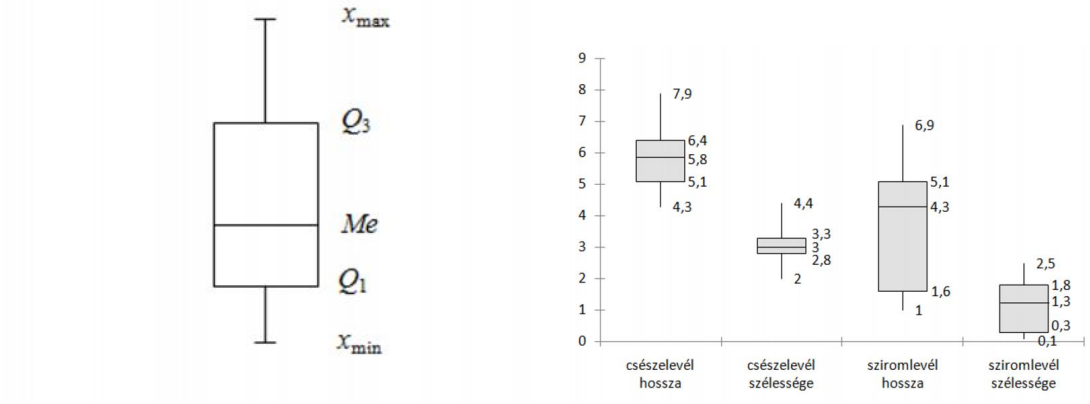
Az empirikus szórásnégyzet az adatok mérésére szolgáló skála mértékében fejezi ki az adatok átlagos eltérését. Amennyiben különféle mértékegységű adatok szórását szeretnénk összehasonlítani, akkor erre egy skálafüggetlen mértékegységet kell használni. A *variációs együttható* egy mértékegység-független mutató, amely a szórás átlaghoz viszonyított mértékét fejezi ki százalékos formában. A variációs együttható a következőképpen számítható ki:

$$V_x = \frac{\sigma_x}{\bar{x}} \tag{2.6}$$

A grafikus szemléltetés a számokat értelmezhetőbbé teszi. A fentiekben említett jellemző mérőszámok tömör, grafikus ábrázolási módja a *boxplot* diagram (szokás még „box and whiskers” ábrázolásnak is nevezni). A boxplot diagram a vizsgált változó 5 nevezetes mérőszámát (minimum, maximum, kvartilisek) egy egyenesen helyezi el oly módon, hogy Q_1, Me és Q_3 által az adatok 50%-át dobozba zárva tünteti fel. A 2.1(a) ábra a boxplot diagram általános felépítését, a 2.1(b) ábra pedig az iris adatsor adatainak boxplot ábrázolását szemlélteti. Speciális esetekben szokás a boxplot diagram különböző módosított formáit is alkalmazni, melyekben az előbb említett 5 jellemző mérőszám helyett egyéb mérőszámok (pl. átlag, átlag±szórás és átlag±szórás konstansszorosa) kerülnek ábrázolásra.

2.1.2. Gyakorisági eloszlás

Nagy adathalmaz esetén, ahhoz, hogy megfelelő rálátással rendelkezzünk az attribútum által felvett értékek elhelyezkedésére vonatkozóan, meg kell vizsgálni az értékek eloszlását.



(a) A boxplot diagram adatai

(b) Az iris adatok boxplot diagramja

2.1. ábra. Boxplot diagramok

Diszkrét és folytonos változók eloszlásának meghatározása során az elemzőknek más és más módszereket kell alkalmazniuk.

Folytonos értékeket tartalmazó attribútumok eloszlásának feltérképezéséhez az attribútum terjedelmét osztályközökre kell osztani, majd meg kell határozni az egyes osztályok elemeinek relatív gyakoriságát (a minta egészéhez viszonyítva). Általában jellemző, hogy az osztályok hossza (terjedelme) azonos, ettől csak ritka esetben, illetve a későbbi elemzések során szokás eltérni. Az osztályok számának meghatározására nincsenek egzakt szabályok. Általánosságban azt mondhatjuk, hogy eleinte célszerű több osztályt kialakítani, s amennyiben az osztályok száma túl nagy, akkor azok összevonásával ez a számosság csökkenthető. Tapasztalati alapokon kiindulva a következő két formula nyújthat segítséget az osztályok számának megállapításában:

$$2^{c_0} > N \tag{2.7}$$

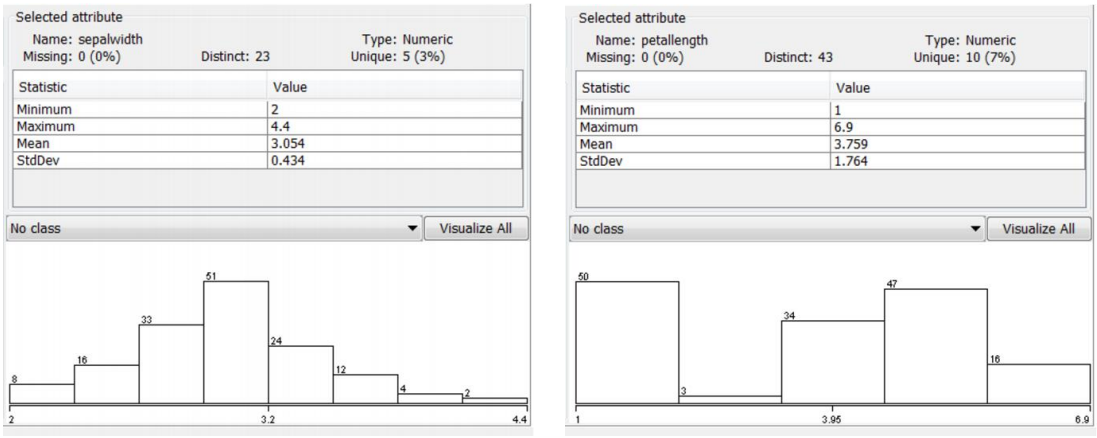
$$c_0 = 1 + 3,3 \times lgN, \tag{2.8}$$

ahol c_0 jelöli a minimálisan kialakítandó osztályok számát, N pedig az attribútum számossága.

A gyakorisági eloszlások szemléltetése *hisztogramon* történik. A hisztogram a gyakorisági eloszlás oszlopos formában történő ábrázolása, ahol a téglalapok magassága a gyakoriságot, a szélessége pedig az osztályközt jeleníti meg.

Konkrét példát tekintve, vizsgáljuk meg az iris adathalmaz csészelevél szélességének és sziromlevél hosszúságának az eloszlását, melyet a 2.2 ábra szemléltet. Az adatok teljesebb értelmezése végett ezen ábra egyéb statisztikai értékeket is tartalmaz a vizsgált adathalmazra vonatkozóan. A 2.2(a) ábra hisztogramján látható, hogy az adatbázisban tárolt csészelevél

szélességi adatok normál eloszlást mutatnak. A sziromlevél hosszúságáról tárolt adatok esetében feltűnik, hogy az értéktartomány gyakorlatilag két részre oszlik. Felmerülhet a kérdés, hogy vajon azon iris virágok, melyek sziromlevelének hossza egyértelműen rövidebb, mint a többi vizsgált virág sziromlevele, nem alkotnak-e egy önálló alfajt. Amennyiben részletesebben szemügyre vesszük az adathalmazt, akkor azt találjuk, hogy valóban, ezen virágok egy külön alfajt alkotnak, ez az alfaj pedig az Iris Setosa.



(a) A csészelevél szélességének hisztogramja *(b) A sziromlevél hosszúságának hisztogramja*

2.2. ábra. Folytonos adatok gyakorisági eloszlása

Diszkrét értékű attribútumok esetén a gyakorisági eloszlás hasonlóan alakul a folytonos értékű attribútumok esetéhez, azonban az értéktartomány előbb ismertetett k egyenlő részre történő felosztása nem lehetséges. Az attribútum által felvett diszkrét értékek gyakorlatilag már elvégzik az értéktartomány felosztását, így az elemzést végző szakembereknek csupán arról kell dönteniük, hogy ezen felosztás alapján határozzák-e meg a gyakorisági eloszlásokat, vagy esetleg (például túl sok diszkrét érték esetén) bizonyos attribútumértékek egy csoportba történő összevonásával új csoportokat hoznak-e létre. Az összevonás alapja mindig valamilyen hasonlóság kell hogy legyen, így például a 1.2 fejezetben említett iskolai végzettség attribútum esetében a „BSc (főiskola)” és az „MSc (egyetem)” kategóriák összevonhatók egy „felsőfokú végzettség” kategóriába. A gyakorisági eloszlás számítása ezt követően analóg módon folytatódik, vagyis minden egyes csoportra meg kell határozni a csoport számosságának relatív gyakoriságát, majd az így kapott értékek grafikonon ábrázolhatók. Diszkrét értékek gyakorisági eloszlása esetén a grafikon x tengelye nem folytonos, hanem diszkrét skála, amely a kialakított kategóriák értékeit tartalmazza.

2.2. Többváltozós elemzés

Az adatbázisban tárolt adatok elemzése jellemzően számos változó együttes vizsgálatát jelenti. Érdemes tehát megvizsgálni, hogy ezen változók között van-e kapcsolat, illetve ha igen,

akkor a változók hogyan befolyásolják egymás értékeit. A változók közti kapcsolat erősségét matematikai eszközökkel a korrelációszámítás alkalmazásával, a feltárt kapcsolatok leírását pedig a regressziószámítás segítségével végezhetjük el.

2.2.1. Lineáris korreláció

Amennyiben két attribútum közti kapcsolat meglétét és erősségét szeretnénk vizsgálni, akkor a leglátványosabb megoldás, ha felrajzoljuk a két változó pont-pont diagramját (felhődiagram, scatterplot). Ha a két változó között lineáris kapcsolat áll fenn, akkor a diagramon az adatpontok egy egyenes mentén helyezkednek el. Minél erősebb a kapcsolat a két vizsgált változó között, a pontok annál jobban rásimulnak az egyenesre. Pozitív lineáris korreláció esetén az egyik változó értékének növekedése a másik változó értékének növekedését vonja maga után. Negatív lineáris korreláció esetén amennyiben az egyik változó értéke nő, akkor a másik változó értéke csökken. Ha a két változó korrelálatlan, akkor a pontok „összevissza" szétszórtan helyezkednek el a síkban. A változók között természetesen létezhet egyéb, nem lineáris kapcsolat is, ebben az esetben a pontok egy tetszőleges görbe alakját mintázzák.

A *korrelációszámítás* a vizsgált változók közti lineáris kapcsolat erősségét vizsgálja, és írja le oly módon, hogy a kapcsolat erősségét számszerűen fejezi ki. A vizsgált kapcsolat erősségét a *korrelációs együttható* adja meg. A lineáris korreláció Pearson-féle korrelációs együtthatója a következőképpen számítható ki:

$$r = \frac{\sum_{i=1}^{N} (x_i - \bar{x}) \times (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N} (x_i - \bar{x})^2 \times \sum_{i=1}^{N} (y_i - \bar{y})^2}} \tag{2.9}$$

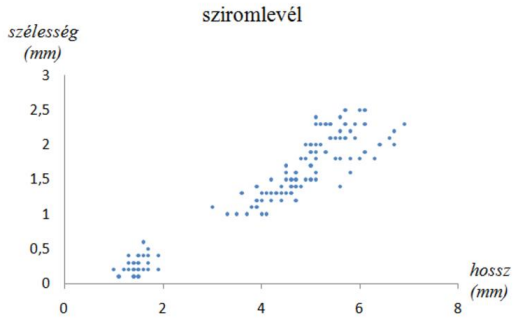
ahol x_i és y_i a vizsgált változók értékeit jelölik, $\bar{x}$ és $\bar{y}$ a változók számtani átlaga, N pedig x és y számossága. Az r dimenzió nélküli mérőszám, értéke a $[-1,1]$ intervallumba esik. $r = 1$ esetén maximális pozitív lineáris korreláció áll fenn a vizsgált két értéksor között. Az $r = -1$ maximális negatív lineáris korrelációt fejezi ki, az $r = 0$ érték pedig azt jelzi, hogy a két változó korrelálatlan. Minél közelebb esik r értéke a -1, vagy 1 értékhez, annál erősebb a lineáris korreláció a vizsgált adatok között. Általában az $r \leq -0,7$ és $r \geq 0,7$ értékekre szokás azt mondani, hogy erős korrelációs kapcsolatot fejeznek ki, de ennek megítélése a vizsgált változók függvényében változhat.

Mint láthatjuk, a korrelációs együttható páronként írja le a változók közti kapcsolat erősségét. Egy adatbázisban természetesen számos változópár közti kapcsolatot kell ellenőrizni. Az így adódó páronkénti korrelációs együtthatók tömör tárolási formája a *korrelációs mátrix* (táblázat), melyre a 2.3 ábra (a) részábrája mutat példát.

A 2.3 ábra a korrelációszámítás eredményét mutatja be egy konkrét példán keresztül. A 2.3(a) részábra az iris adathalmaz csésze- és sziromlevél mért hosszúsági és szélességi értékeinek Pearson-féle korrelációs együtthatóit foglalja össze. Miután a korrelációs együttható definíciója alapján szimmetrikus, ezért elegendő a korrelációs mátrix egyik felét megadni. Az értékekből kiolvasható, hogy a legerősebb korreláció (természetesen nem számítva a változó önmagával való korrelációját) a sziromlevél hossza és szélessége között áll fenn. Ezen

r	csészelevél hossza	csészelevél szélessége	sziromlevél hossza	sziromlevél szélessége
csészelevél hossza	1			
csészelevél szélessége	-0,1094	1		
sziromlevél hossza	0,8718	-0,4205	1	
sziromlevél szélessége	0,8180	-0,3565	0,9628	1

(a) Az iris adathalmaz korrelációs táblája



(b) A sziromlevél hosszúságának és szélességének felhődiagramja

2.3. ábra. Az iris adathalmaz korrelációs együtthatói és ábrázolása

két attribútum felhődiagramját a 2.3 ábra (b) része szemlélteti. Az előzetes elgondolásoknak megfelelően láthatjuk, hogy az adatpontok egy emelkedő egyenes mentén helyezkednek el.

A lineáris korrelációs együttható kiszámíthatóságának vannak azonban feltételei is. A lineáris korrelációs együttható csak folytonos értékű attribútumok esetén számítható ki, illetve az attribútum értékeinek normál eloszlást mutató populációból kell származniuk. A vizsgált változók mérésének egymástól függetlenül kell történnie, és ugyancsak teljesülnie kell, hogy a változók ugyanazon objektumok megfigyeléséből származnak, tehát olyan összehasonlításokat nem érdemes végezni, amelyek során a változók olyan két különböző adatbázisrendszerből származnak, amelyek más és más objektumok, populációk tulajdonságainak rögzítését végzik el.

Többváltozós korreláció

A valós világban azonban egy változó (eredményváltozó, függő változó) értékét jellemzően több másik változó (tényezőváltozó) is befolyásolja. A *parciális korrelációs együttható* az mutatja meg, hogy milyen szoros a kapcsolat valamelyik kiválasztott tényező és a függő változó között, ha a többi tényezőváltozó hatását mind a vizsgált tényezőváltozóból, mind az eredményváltozóból kiszűrjük. Kiindulásként tekintsük a korrelációs mátrix általános formáját oly módon, hogy a mátrix első sora, illetve első oszlopa az eredményváltozó és az egyes tényezőváltozók közötti kapcsolat szorosságát mérő lineáris korrelációs együtthatókat tartalmazza, a mátrix többi eleme pedig a tényezőváltozók egymás közötti korrelációját adja meg. A korrelációs mátrix általános alakja tehát:

$$R = \begin{bmatrix} 1 & r_{yx_1} & r_{yx_2} & \dots & r_{yx_p} \\ r_{x_1y} & 1 & r_{x_1x_2} & \dots & r_{x_1x_p} \\ r_{x_2y} & r_{x_2x_1} & 1 & \dots & r_{x_2x_p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r_{x_py} & r_{x_px_1} & r_{x_px_2} & \dots & 1 \end{bmatrix}$$

ahol p a változók számát jelöli, y a függő változó, x-ek pedig a tényezőváltozók. Háromváltozós modellben az y és az x_1 közti parciális korrelációs együttható (függetlenítve az x_2 változótól) a következőképpen határozható meg:

$$r_{yx_1.x_2} = \frac{r_{yx_1} - r_{yx_2} * r_{x_1x_2}}{\sqrt{\left(1 - r_{yx_2}^2\right) * \left(1 - r_{x_1x_2}^2\right)}} \tag{2.10}$$

A parciális korrelációs együttható szokványos jelölése szerint az indexben először azon változókat soroljuk fel, amelyeket vizsgálunk, majd egy ponttal elválasztva következnek azon változók, amelyek hatását kiszűrjük. Az $r_{yx_2.x_1}$ és $r_{x_1x_2.y}$ értéke analóg kiszámítható. A parciális korrelációs együttható értéke szintén a $[-1,1]$ intervallumból vesz fel értékeket.

A páronkénti parciális korrelációs érték háromnál több változó esetén is kiszámítható, azonban ekkor a számításhoz a korrelációs mátrix inverzét kell alapul vennünk, amely legyen a következő:

$$R^{-1} = \begin{bmatrix} q_{yy} & q_{yx_1} & \cdots & q_{yx_j} & \cdots & q_{yx_p} \\ q_{x_1y} & q_{x_1x_1} & \cdots & q_{x_1x_j} & \cdots & q_{x_1x_p} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{x_jy} & q_{x_jx_1} & \cdots & q_{x_jx_j} & \cdots & q_{x_jx_p} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{x_py} & q_{x_px_1} & \cdots & q_{x_px_j} & \cdots & q_{x_px_p} \end{bmatrix}$$

Ezen mátrix alapján az y és x_j változók parciális korrelációs együtthatója a következőképpen számítható ki:

$$r_{yx_j.x_1,x_2,\ldots,x_{j-1},x_{j+1},\ldots,x_p} = \frac{-q_{yx_j}}{\sqrt{q_{yy} * q_{x_jx_j}}} \tag{2.11}$$

Többváltozós modellben amennyiben az y változó $x_1,\ldots,x_p$ változóktól történő együttes függését kívánjuk meghatározni, akkor a változók *többszörös korrelációs együtthatóját* kell meghatározni. A többszörös korrelációs együttható speciális háromváltozós modellben a 2.12 képlet alapján, általános többváltozós modellben pedig a 2.13 alapján számítható ki:

$$r_{y.x_1,x_2} = \sqrt{\frac{r_{yx_1}^2 + r_{yx_2}^2 - 2r_{yx_1}r_{yx_2}r_{x_1x_2}}{1 - r_{x_1x_2}^2}} \tag{2.12}$$

$$r_{y.x_1,x_2,\ldots,x_p} = \sqrt{1 - \frac{1}{q_{yy}}} \tag{2.13}$$

Kategorikus változók függetlenségének vizsgálatára a fentebb említett módszerek nem alkalmasak. Amennyiben a vizsgált attribútumok kategorikus értékeket vesznek fel, akkor ezen változók függetlenségének vizsgálatát a χ^2-*próba* segítségével végezhetjük el. A χ^2-próba tulajdonképpen abból a nullhipotézisből indul ki, hogy a vizsgált változók függetlenek, s összehasonlítja a valódi gyakorisági táblázatot azzal az elméleti gyakorisági táblázattal, amely a függetlenség esetén állna fenn. A próba alkalmazhatósági feltétele, hogy az elméleti gyakorisági táblázatban cellánként legalább 2 elem legyen, és legfeljebb a cellák 20%-ában lehet 5-nél kevesebb elem. A gyakorlatban a χ^2-próba széles körben elterjedt, mivel nem

tartalmaz megkötést a változók eloszlására vonatkozóan. Ezen jellemzőjéből adódóan nem normál eloszlású folytonos változók esetén is alkalmazható oly módon, hogy a folytonos változókat kategorizáljuk.

Természetesen léteznek további korrelációszámítási módszerek is. Így például gyakran használatos még a Spearman-féle korrelációs együttható, amely rendezett, vagy nem normál eloszlást mutató folytonos adatok közti korreláció számítása során nyújt hasznos segítséget.

2.2.2. Regresszió

A fejezet bevezetőjében említettük, hogy a korrelációszámítás eredménye a lineáris kapcsolat erősségét fejezi ki, a változók közti kapcsolat matematikai leírását pedig a *regressziószámítás* segítségével adhatjuk meg. Amennyiben y és x_1 változók között *lineáris kapcsolatot* feltételezünk, akkor a kapcsolat leírásához keressük azt az $y = \beta_1 x_1 + \beta_0$ egyenest, amely legjobban közelíti a ponthalmazt. Több változó esetén a lineáris regressziós modellben a függő változó a független változók lineáris kombinációjaként a 2.14 egyenlettel írható fel (többszörös lineáris regresszió).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon \tag{2.14}$$

A fenti egyenlet egy hipersíkot definiál, ahol ε a regressziós hipersík hibatagja (reziduálja). Cél a $\beta_0, \beta_1, \ldots, \beta_p$ tényezők meghatározása oly módon, hogy az ε hibatagot minimalizáljuk, vagyis az egyenlet által becsült és valós y értékek a legkevésbé térjenek el egymástól. Erre a célra a leggyakrabban alkalmazott módszer az eltérések négyzetösszegének minimalizálása. Az eltérések négyzetösszege a következőképpen számítható:

$$\sum_{i=1}^{N} e^2 = \sum (y - \widehat{y})^2, \tag{2.15}$$

ahol N az adatpontok száma, $\widehat{y}$ a becsült érték, ami a következőképpen adódik:

$$\widehat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p \tag{2.16}$$

A keresett egyenlet $\beta_0, \beta_1, \ldots, \beta_p$ paramétereinek értéke a 2.15 egyenlet $\beta_0, \beta_1, \ldots, \beta_p$ szerinti parciális deriváltjainak meghatározásával állítható elő.

Természetesen nem csak lineáris kapcsolat állhat fenn a változók között, hanem egyéb nemlineáris kapcsolat is. Ezen nemlineáris kapcsolatok számos esetben visszavezethetőek lineáris esetre, mint például a függő és független változók közt fennálló *exponenciális kapcsolat* is, amelyet a 2.17 egyenlet ír le. Ezen egyenlet a 2.18 egyenlet formájában visszavezethető lineáris alakra, ahol a lineáris kapcsolat nem y és x változók értékei között, hanem a $\log y$ és x értékek között áll fenn, tehát nincs más dolgunk, mint az eredeti y változók logaritmusát képezni, és $\log y$-ok és x-ek között keresni a lineáris kapcsolatot.

$$y = ab^x \tag{2.17}$$

$$\log y = \log a + x \log b \tag{2.18}$$

Hasonló a helyzet *hatványfüggvénnyel leírható kapcsolat* esetén is (2.19), melynek lineáris alakra visszavezetett formáját a 2.20 egyenlet mutatja. Láthatjuk, hogy a lineáris kapcsolat

a $\log y$-ok és $\log x$-ek között áll fenn, tehát első lépésben az eredeti változók logaritmusát kell képezni, s ezek között kell keresni a lineáris összefüggést.

$$y = ax^b \tag{2.19}$$

$$\log y = \log a + b \log x \tag{2.20}$$

Természetesen léteznek ettől eltérő regressziós modellek is, így például a *polinomiális regresszió* (2.21 egyenlet), amelyet tipikusan olyankor alkalmazunk, amikor a várt görbének minimuma vagy maximuma van. Polinomiális regresszió esetében célszerű minél alacsonyabb fokszámra törekedni, mivel magas fokszám esetén a paraméterek értelmezése szinte lehetetlen.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_p x^p \tag{2.21}$$

Mint látható számos matematikai, statisztikai eszköz létezik az adatbázisban tárolt változók összefüggéseinek vizsgálatára vonatkozóan. Ezen eszközök alkalmazásával az elemzést végző szakemberek feltárhatják az egyes attribútumok közti kapcsolatokat, melyek fontos ismereteket szolgáltatnak a későbbi elemzésekhez. Az [1] irodalomban további regresszió-számítási módszerekről és a regresszió eredményének értékeléséről találunk leírást. A korrelációszámítás rejtelmeibe és egyéb statisztikai módszerekbe a [12, 25, 26, 28] irodalmak nyújtanak részletes betekintést.

3. fejezet

A dimenzionalitás csökkentése

3.1. A dimenziócsökkentés célja, főbb módszerei

Az adatbázisok jellemzően témaspecifikusak, tehát egy vizsgált terület adatait gyűjtik össze. Ezek az adatok az adott témakör objektumaira jellemző tulajdonságok. Egy-egy objektumot számos tulajdonsággal jellemezhetünk, s az objektumokat, mint adatpontokat ezen tulajdonságok vektorterében képzelhetjük el. Ahhoz, hogy a D db tulajdonsággal jellemzett objektumok egymáshoz való viszonyát, csoportosulásait megállapíthassuk, ezen D-dimenziós adattérbe kell belátást nyernünk. $D = 1, 2, 3$ esetén ez nem okoz gondot, azonban magasabb dimenziószám mellett az emberi érzékelés korlátaiba ütközünk. De mivel tudjuk, hogy egy kép, egy ábra gyakran többet ér számos leírásnál, kiszámított értéknél, ezért az adatelemzési folyamat során hathatós segítséget nyújtanak azon eszközök, melyek a sok tulajdonsággal jellemzett objektumok egymáshoz viszonyított kapcsolatait az emberi szem számára is láthatóvá teszik.

Első ránézésre azt gondolhatnánk, hogy a részletes leírásnak csupán pozitív hatásai vannak az elemzések szempontjából. Richard Bellman cikke [3] azonban rámutat arra a tényre is, hogy a magas dimenzionalitásnak hátrányai is vannak. Bellman által a *dimenzionalitás átkának* nevezett matematikai jelenség szerint ahhoz, hogy a vizsgált objektumhalmaz megfelelő leírása megadható legyen, a dimenzió számának növekedésével a vizsgált mintaobjektumok számának exponenciálisan kell növekednie. Tehát minél több tulajdonság jellemzi a vizsgált objektumhalmazt, annál több minta szükséges annak kellő pontosságú jellemzéséhez.

Láthatjuk tehát, hogy az objektumok dimenziócsökkentésének több haszna is van. Egyrészt a vizsgált objektumokat, mint adatpontokat láthatóvá tehetjük az emberi szem számára is, illetve az objektumokra jellemző tulajdonságok számát csökkentve kisebb számosságú adathalmaz esetén is pontosabb elemzést adhatunk meg.

Matematikai szempontból tekintve, a dimenzionalitás csökkentésének célja a vizsgált magas dimenzionalitású (D-dimenziós) adathalmaz olyan alacsony dimenzionalitású (d-dimenziós) reprezentációja, amely leginkább megőrzi az adathalmazban rejlő információkat, s annak struktúráját. Formálisan, adott az $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ objektumhalmaz, ahol $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]$ az i. objektum, melyet D tulajdonság jellemez. A dimenzionalitást csökkentő eljárások a vizsgált $\mathbf{X}$ objektumhalmazt egy új, d ($d \ll D$) dimenziós $\mathbf{Y}$ ($\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$, $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{id}]$) objektumhalmazba képezik le.

A dimenzionalitás csökkentése a következő két módon valósulhat meg:

- *a jellemzők szelektálásával*: amely bizonyos jellemző tulajdonságok elhagyását jelenti, illetve
- *az objektumtér transzformálásával*: amely a meglévő tulajdonságokból kiindulva új, de kevesebb számú tulajdonságot hoz létre.

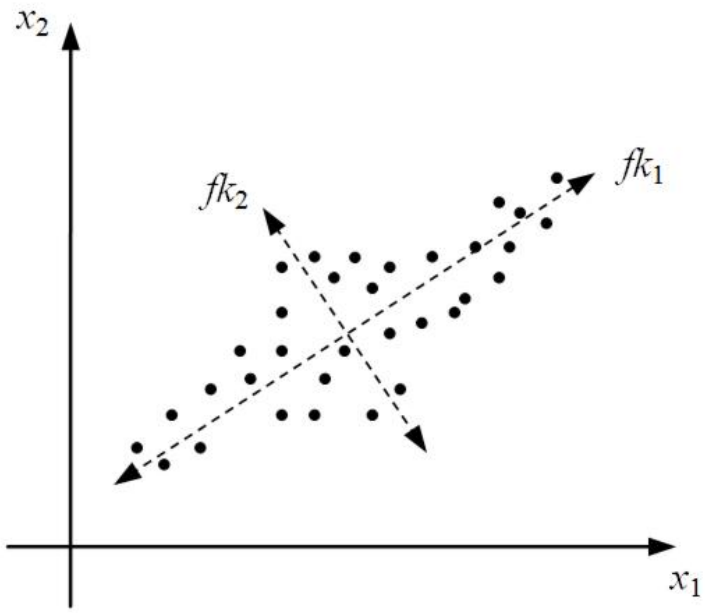
A *jellemzők szelektálásának* alapját az a tény adja, hogy az elemzésekből kihagyhatóak azok a nem releváns attribútumok, amelyek nem járulnak jelentősen hozzá az információtartalom növeléséhez. A *szekvenciális tulajdonságkiválasztás* szemlélete szerint a jellemzők szelektálása az erre a célra kiválasztott attribútumok hozzáadásával, illetve elvételével iteratív módon történik. Ezek a gyakorta használatos szekvenciális módszerek két kategóriába sorolhatók az alapján, hogy üres halmazból indulnak-e ki, s ehhez vesznek-e hozzá fokozatosan újabb és újabb attribútumokat (Sequential Forward Selection (SFS) algoritmusok és Gerneralized Sequential Forward Selection (GSFS) algoritmusok), illetve a teljes tulajdonsághalmazból indulnak-e ki, melyből az attribútumok lépésről-lépésre történő törlése által érik el a kívánt eredményt (Sequential Backward Selection (SBS) algoritmusok és Generalized Sequential Backward Selection (GSBS) algoritmusok). Ezen algoritmusok átfogó ismertetése a [22] irodalomban található meg. Miután azonban ezek az algoritmusok nem vizsgálják meg az összes attribútum-részhalmazt, mint lehetséges megoldást, ezért futásuk nem feltétlenül eredményez optimális megoldást. Továbbfejlesztett változataikban a jellemzők hozzáadása és törlése már egy lépésen belül is megvalósítható (Seqvential Floating Forward Selection (SFFS) algoritmus [29] és Seqvential Floating Backward Selection (SFBS) algoritmus [29]). A jellemzők kiválasztása különféle módon történhet, melyek közül az információnyereség elvét érdemes kiemelni. Ezen elv részletes ismertetése a 4.3.3 fejezetben található meg.

A *objektumtér transzformálása* során a magas dimenzionalitású adatoknak egy alacsonyabb dimenzionalitású adattérbe történő transzformálása történik oly módon, hogy az adatokat leíró tulajdonságok felhasználásával új, de kevesebb számú jellemző tulajdonság jön létre. Ez a transzformáció lehet lineáris, illetve nemlineáris jellegű. *Lineáris objektumtér transzformáció* esetén az objektumokra jellemző új tulajdonságok az eredeti attribútumok lineáris kombinációiként jönnek létre. Az objektumtér lineáris transzformációjára a *főkomponens analízist*, a *független komponens analízist*, illetve az osztályozási feladatok esetében alkalmazható *lineáris diszkriminancia analízist* használják leggyakrabban. Napjainkban azonban egyre inkább tért hódítanak a *nemlineáris objektumtér transzformációs eljárások*, melyek szakítanak a lineáris kombináció szemléletével. Ezen algoritmusok alkalmazása által az olyan adatstruktúrák is nagyobb valószínűséggel felismerhetővé válnak, ahol a magas dimenzionalitású térben az objektumok egy beágyazott, nem lineáris sokaság mentén helyezkednek el. A 3.6(a) ábrán látható úgynevezett „Swiss roll” adathalmaz is egy ilyen tipikus belső rejtett struktúrát mutat be, ahol a 3-dimenziós térben egy 2-dimenziós sík nemlineáris beágyazása figyelhető meg. A nemlineáris objektumtér transzformációs módszerek közül a leggyakrabban alkalmazott eljárások közé tartoznak a *többdimenziós skálázás* módszerei, a *Kohonen-féle önszerveződő hálózat*, illetve a *lokálisan lineáris beágyazás* módszere.

A fejezet további részeiben a leggyakrabban alkalmazott objektumtér transzformációs eljárásokat mutatjuk be.

3.2. Főkomponens analízis

A *főkomponens analízis* (Principal Component Analysis, PCA) [14, 18] az egyik leggyakrabban alkalmazott lineáris objektumtér transzformációs módszer. A módszert gyakorta szokás Hotelling, vagy Karhunen-Loève transzformációnak is nevezni. A főkomponens analízis célja, hogy a nagy számú, egymással korreláló változókból kevesebb, egymással korrelálatlan változókat hozzon létre, s ezen változókkal írja le a vizsgált objektumokat. Geometria megközelítéssel élve a PCA egy olyan ortogonális lineáris transzformáció, amely az adathalmazt egy új koordináta-rendszerbe transzformálja oly módon, hogy az a komponens, amelynek legnagyobb a varianciája az első koordinátatengely (első főkomponens) mentén helyezkedjen el, a második legnagyobb varianciájú komponens a második koordináta (második főkomponens) mentén helyezkedjen el, és így tovább. A PCA algoritmus tehát úgy forgatja be az adathalmazt egy kisebb dimenzionalitású hipersíkba, hogy az adatok varianciája minél jobban látható legyen. Miután az objektumfelhő a transzformáció során kisebb dimenzionalitású hipersíkba kerül, ezért az eredeti objektumok ettől a hipersíktól a valóságban el is térhetnek. Ez az eltérés adja a leképezés hibáját. A 3.1 ábra a koordináta-rendszer transzformációját szemlélteti egy egyszerű esetben, ahol a kiindulási és eredménytér dimenzionalitása megegyezik. Az ábrán x_1 és x_2 az eredeti koordinátatengelyt, fk_1 és fk_2 pedig az új főkomponenseket jelölik.



3.1. ábra. A főkomponensek geometriája

A PCA algoritmus tulajdonképpen az objektumok kovariancia mátrixának sajátérték felbontásán alapul. Bemeneti paraméterként adott az $\mathbf{X}$ $D \times N$ dimenziós adatmátrix, ahol az egyes sorok a változóknak, az oszlopok pedig az egyes objektumoknak feleltethetők meg (a

szokásos adattárolási mátrix transzponáltja). Az algoritmus működése a következő lépésekben összegezhető:

1. lépés: Az algoritmus első lépésben minden dimenzió mentén kivonja az adott dimenzió átlagát a dimenzióadatok értékéből. Ezáltal előáll egy olyan adathalmaz ($\widehat{\mathbf{X}}$), melynek átlaga 0.

2. lépés: Az objektumok kovariancia mátrixának kiszámítása. Ez egy D-dimenziós objektumhalmaz esetén egy $D \times D$ dimenziós szimmetrikus mátrixot eredményez, amely a következőképpen adódik:

$$\mathbf{C} = \frac{1}{N-1}\widehat{\mathbf{X}}\widehat{\mathbf{X}}^T, \tag{3.1}$$

ahol $\widehat{\mathbf{X}}$ a normalizált adatok $D \times N$ méretű mátrixa. A $\mathbf{C}$ mátrix c_{ij} komponense az i. és j. változók kovarianciáját, a c_{ii} pedig a i. változó varianciáját jelöli. Ha az i. és j. komponensek nem korrelálnak, akkor kovarianciájuk 0 ($c_{ij} = c_{ji} = 0$).

3. lépés Az ortogonális bázis kiszámítása a kovariancia mátrix sajátvektorai és a sajátértékei alapján. Ehhez meg kell oldani a mátrix sajátérték egyenletét, amely a következő:

$$\mathbf{C}\mathbf{e}_i = \lambda_i \mathbf{e}_i, i = 1, 2, \ldots, D \tag{3.2}$$

ahol $\mathbf{e}_i$ a sajátvektorokat, λ_i pedig a sajátértékeket jelöli. Az egyenlet megoldásához feltételezzük, hogy λ_i sajátértékek különbözőek.

Ezután az algoritmus az eredményül kapott sajátvektorokat sorba rendezi a sajátértékek szerinti csökkenő sorrendbe. A keresett ortogonális bázist a sorbarendezés szerinti első d sajátvektor adja.

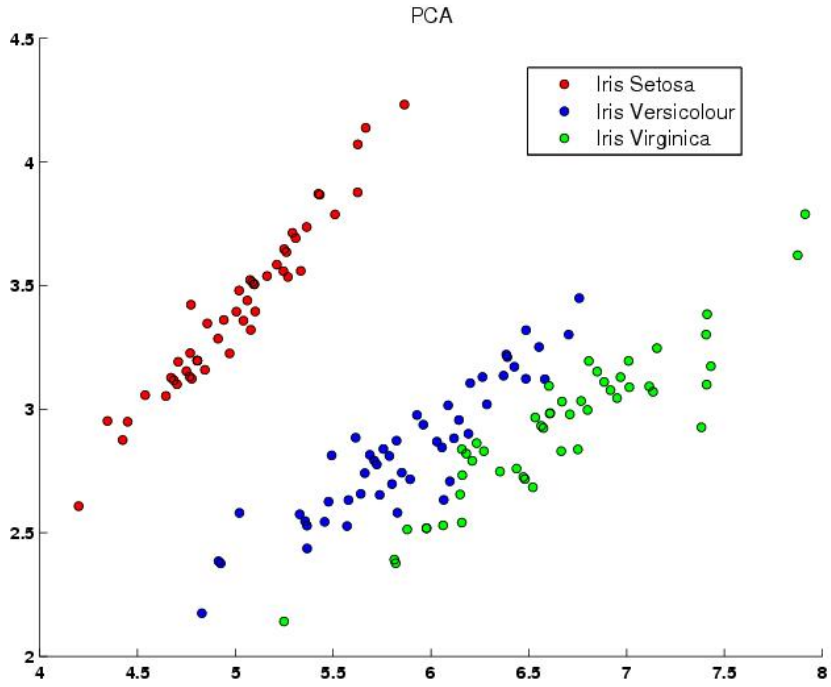
4. lépés: Az új adathalmaz létrehozása az eredeti objektumhalmazból a következő módon:

$$\mathbf{Y} = \mathbf{W}\widehat{\mathbf{X}}, \tag{3.3}$$

ahol $\mathbf{W}$ egy $d \times D$ dimenziós mátrix, amely a kovarianciamátrix legfontosabb sajátvektorait (d db) tartalmazza sorvektorokként oly módon, hogy legfelül a legnagyobb sajátértékkel rendelkező sajátvektor helyezkedik el, alatta a második legnagyobb sajátértékű sajátvektor, és így tovább lefelé. $\widehat{\mathbf{X}}$ az eredeti objektumhalmaznak az attribútumok átlagaival korrigált reprezentációja.

Az algoritmus bemutatása után tekintsünk meg egy valós leképezési eredményt is. A korábbiakban bemutatott iris adathalmaz 4-dimenziós leírásából a főkomponens analízis által létrehozott 2-dimenziós reprezentáció a 3.2 ábrán tekinthető meg. Az ábrán az egyes alfajok egyedeit eltérő színekkel tüntettük fel. Mint láthatjuk, az Iris Setosa egyedei jól elkülönülnek a másik két alfaj egyedeitől, melyek azonban egymáshoz közel, kevésbé szeparáltan helyezkednek el.

A *független komponens analízis* (Independent Component Analysis, ICA) a főkomponens analízishez szorosan kapcsolódó, annak továbbgondolásaként született dimenziócsökkentési



3.2. ábra. Az iris virágok 2-dimenziós PCA reprezentációja

módszer. Amíg a PCA célja a tömörített adatok leképezési hibájának a minimalizálása, addig az ICA algoritmus célja az objektumokat leíró komponensek statisztikai függetlenségének maximalizálása. Ellentétben a PCA algoritmussal, az ICA algoritmusok eredményeképpen létrejött komponensek nem feltétlenül ortogonálisak. Mivel a független komponens analízis célja az egymástól független komponensek feltárása, ezért elsősorban osztályozási, csoportosítási problémák (pl. arcfelismerés) esetén alkalmazott módszer. A független komponens analízis részletes bemutatása az A. Hyvärinen, J. Karhunen és E. Oja szerzők által jegyzett könyvben található meg [15].

3.3. Többdimenziós skálázás

A *többdimenziós skálázás* (Multidimensional scaling, MDS) a legszélesebb körben alkalmazott nemlineáris dimenziócsökkentési eljárás. A többdimenziós skálázás elnevezés valójában gyűjtőfogalom, amely számos, hasonló filozófián alapuló dimenziócsökkentési eljárás összefoglaló neve. Ezen dimenzióredukciós eljárások célja a vizsgált objektumhalmaznak kis dimenziószámú térben történő szemléltetése oly módon, hogy azok az objektumok, amelyek az eredeti magas dimenzionalitású hipertérben közel helyezkednek el egymáshoz, azok az eredményül kapott kis dimenzionalitású térben is közel legyenek egymáshoz, illetve azok az objektumok amik az eredeti magas dimenzionalitású térben távol helyezkednek el egymástól, azok a leképezés eredményterében is távol legyenek egymástól. Mint látható, az MDS algoritmusok során az objektumok közötti távolság meghatározó jelentőséggel bír. Azonban két objektum távolságának meghatározása sok esetben nem is olyan egyszerű feladat, hiszen

az objektumokat leíró tulajdonságok lehetnek felsorolás, rendezett, intervallum-, vagy arányskálázott típusú attribútumok is. Az objektumok közötti távolságok meghatározásának főbb módszereit a 4.4.2 fejezetben mutatjuk be.

A többdimenziós skálázás algoritmusai az alapján, hogy hogyan kezelik az objektumok közötti távolságokat, illetve ezen távolságoknak mely jellemzőjét emelik ki a következő két fő csoportba sorolhatók:

- *metrikus többdimenziós skálázás*, és
- *nemmetrikus többdimenziós skálázás*.

A *metrikus (klasszikus) MDS* algoritmusok a leképzés során az objektumok közötti távolságok minél pontosabb megőrzésére törekednek. Ezen algoritmusok alapja azon stresszfüggvény (illeszkedési mutató) minimalizálása, amely az eredeti objektumok közötti távolságok és a leképzés eredményeképpen adódó alacsony dimenzionalitású objektumok távolságainak eltérését határozza meg. A leggyakrabban alkalmazott stresszfüggvény az *s-stress* hibafüggvény, amely az egymásnak megfeleltethető valódi és leképzett távolságok eltérésének négyzetösszegét viszonyítja az eredeti távolságok négyzetösszegéhez. Az s-tress matematikailag a következőképpen definiálható:

$$E_{s\text{-}stress} = \frac{1}{\sum_{i<j}^{N} d_{ij}^{*2}} \sum_{i<j}^{N} \left(d_{ij}^{*} - d_{ij}\right)^2, \tag{3.4}$$

ahol d_{ij}^{*} a magas dimenzionalitású $\mathbf{x}_i$ és $\mathbf{x}_j$ objektumok közti távolságot, d_{ij} pedig a leképzés eredményeképpen létrejött $\mathbf{y}_i$ és $\mathbf{y}_j$ objektumok közti távolságot jelöli. Amennyiben a leképzés eredményeképpen létrejött alacsony dimenzionalitású prezentációban az eredményobjektumok távolsága tökéletesen illeszkedik a kiindulási objektumok távolságára, akkor a fenti stresszfüggvény értéke pontosan 0. Az s-stress függvény 0-tól eltérő értékeinek értelmezéséhez a 3.1 nyújthat segítséget.

s-stress	**Értékelése**
[0,00 – 0,05)	Kiváló, valószínűleg minden releváns információt tartalmazó leképezés.
[0,05 – 0,10)	Jó.
[0,10 – 0,20)	Elfogadható.
[0,20 – 1,00)	Az alkalmazott dimenziószám esetén nagy az információveszteség, meg kell próbálni eggyel nagyobb dimenziószámú modellt alkalmazni.

3.1. táblázat. Az s-stress értelmezése

A metrikus többdimenziós skálázási módszer alkalmazhatósága azonban bizonyos mértékben korlátozott. A legnagyobb korlátozást az jelenti, hogy az algoritmus feltételezi, hogy az objektumokat arány- és/vagy intervallumskálázott attribútumok írják le, hiszen a távolság

fogalma csak ezen adattípusok esetében értelmezhető. Az alkalmazhatósági kört tovább szűkíti az olyan attribútumok jelenléte, melyek értékeinek meghatározása emberi szubjektivitáson alapul. Ilyenek lehetnek például a kérdőívekben gyakran előforduló százalékos értékelést váró kérdések eredményei (pl: Hány százalékban ért egyet a ....? típusú kérdések). Az emberek ugyanis hajlamosak ilyen esetekben a széleken nagyobb különbséget meghatározni, mint a skála közepén. Ezen limitációkra vonatkozóan nyújt lehetséges megoldást a nemmetrikus többdimenziós skálázás módszere.

A *nemmetrikus többdimenziós skálázási módszer* a dimenzionalitás csökkentése során az objektumok különbözőségének sorrendiségét, vagyis a különbözőségek rangját kívánja megőrizni. Ezen algoritmusok bemeneti paramétere az objektumok különbözőségi mátrixa. A nemmetrikus MDS célja az, hogy a pontoknak az alacsony dimenzionalitású térben egy olyan konfigurációját adja meg, ahol az objektumpáronként értelmezett euklideszi távolságok sorrendisége (rangja) megegyezik az eredeti objektumok között mért különbözőségek rangjával. Másképpen kifejezve azt is mondhatjuk, hogy a nemmetrikus MDS az objektumoknak egy olyan konfigurációját hozza létre az eredménytérben, ahol az objektumok közti euklideszi távolságok az objektumok különbözőségének monoton transzformációját közelítik.

A nemmetrikus MDS algoritmusok működése során iteratív módon szintén egy stresszfüggvény (*stress 1*, *Kruskal stress*) értékelődik ki, amely a következőképpen adható meg:

$$E_{nemmetrikus\_MDS} = \sqrt{\sum_{i<j}^{N} \left(\widehat{d}_{ij} - d_{ij}\right)^2 / \sum_{i<j}^{N} d_{ij}^2}, \tag{3.5}$$

ahol d_{ij} a leképzés eredményeképpen létrejött $\mathbf{y}_i$ és $\mathbf{y}_j$ pontok távolsága, $\widehat{d}_{ij}$ pedig az úgynevezett pszeudo-távolság, amely d_{ij}-ből származtatható Kruskal monoton regressziós eljárása alapján.

A nemmetrikus MDS algoritmus (Kruskal algoritmusa) egy iteratív folyamat, amely a következőképpen adható meg:

1. lépés: Az algoritmus kezdetben az eredménytérben tetszőlegesen választott objektumkonfigurációból indul ki, majd kiszámítja ezen objektumok páronkénti távolságát. Az iterációk számának kezdeti beállítása: t=0.

2. lépés: Az algoritmus a második, úgynevezett *nemmetrikus fázisban* meghatározza a $\widehat{d}_{ij}^{(t)}$ értékeket a $d_{ij}^{(t)}$ értékekből a $d_{ij}^{(t)}$ értékek és az eredeti objektumok között mért különbözőségi értékek (δ_{ij}) között létrehozott monoton regressziós kapcsolat alapján, azon korlátozás mellett, hogy ha $\delta_{ij} < \delta_{rs}$, akkor $\widehat{d}_{ij}^{(t)} < \widehat{d}_{rs}^{(t)}$ -nek is teljesülnie kell.

3. lépés: A *metrikus fázisban* az eredménytér konfigurációjának módosítása történik oly módon, hogy az újonnan kiszámolt $d_{ij}^{(t+1)}$ értékek közelebb kerüljenek az előző lépésben kiszámított $\widehat{d}_{ij}^{(t)}$ értékekhez.

4. lépés: Az algoritmus negyedik lépésében az eredmény kiértékelése történik, amelyben a $d_{ij}^{(t+1)}$ távolságok és $\widehat{d}_{ij}^{(t)}$ pszeudo-távolságok illeszkedésének vizsgálata történik. Ha az eredmény nem kielégítő, akkor az algoritmus futása a 2. lépéstől iteratív módon folytatódik tovább az iteráció számának $t = t + 1$ növelése mellett.

Látható, hogy az algoritmus nemmetrikus fázisában a $\widehat{d}_{ij}$ értékek az iteratív eljárás során minden iterációban újraszámítódnak oly módon, hogy a sorrendjük megfeleljen az eredeti objektum közti különbözőségek rangjának, és amennyire csak lehet közel legyenek a megfelelő d_{ij} értékhez.

A többdimenziós skálázás további rejtelmeibe a [7] irodalom nyújt részletes betekintést.

3.4. Sammon-leképezés

A *Sammon-leképezés* [31] az egyik legelső nemlineáris dimenziócsökkentési eljárás, amely tekinthető a metrikus többdimenziós skálázás speciális esetének is. A Sammon-leképezés során használt Sammon-stresszfüggvény nagy mértékben hasonlít az s-stress-hez, csupán abban tér el tőle, hogy a távolságmegőrzés hibája az eredeti távolságokkal normalizálva van. A *Sammon-stresszfüggvény* matematikailag a következőképpen definiálható:

$$E_{Sammon} = \frac{1}{\sum\limits_{i<j}^{N} d_{ij}^{*2}} \sum_{i<j}^{N} \frac{\left(d_{ij}^{*} - d_{ij}\right)^2}{d_{ij}^{*}}, \tag{3.6}$$

A fenti normalizációból fakadóan a Sammon-leképezés az s-stress alkalmazásához képest jobban hangsúlyozza a kisebb távolságok pontosabb megőrzését, s ezáltal alkalmasabbá válik az objektumhalmaz rejtett belső struktúrájának megőrzésére. A Sammon-stress kiértékelése során Kruskal javaslata alapján [24] a következő határokat emelhetjük ki:

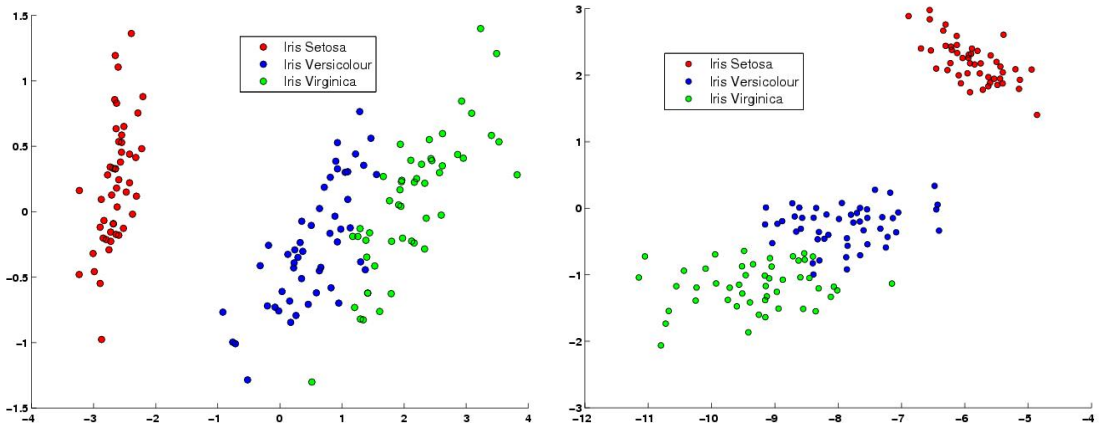
Sammon-stress értéke:	0,3	0,2	0,1	0,025	0,0
Az illeszkedés jósága:	szegényes	elégséges	jó	kiváló	tökéletes

3.2. táblázat. A Sammon-stress kiértékelése Kruskal alapján

Az érdekesség kedvéért tekintsünk meg újfent egy valós leképezési eredményt. A 3.3 ábra az iris adathalmaznak a metrikus többdimenziós skálázás (alkalmazott stresszfüggvény: s-stress) és a Sammon-leképezés által létrehozott 2-dimenziós reprezentációit mutatja. A 2-dimenziós eredménytérben létrejött reprezentációk hasonlítanak egymáshoz, mint ahogy ezt el is várjuk, de az eltérő stresszfüggvényekből adódóan némi megjelenésbeli különbséget is mutatnak.

3.5. Kohonen-féle önszerveződő hálózat

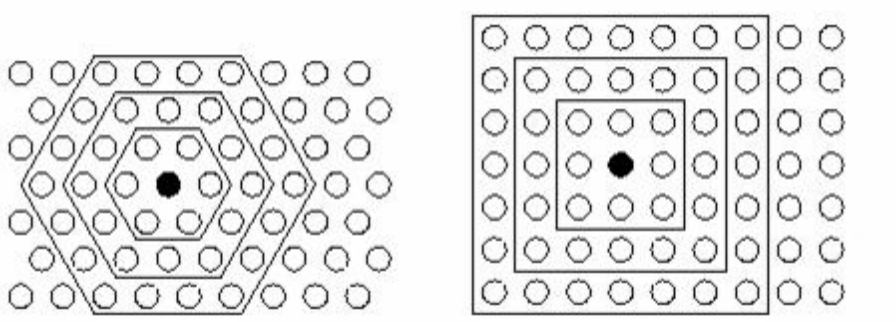
A *Kohonen-féle önszerveződő hálózat* (önszerveződő térkép, Self Organizing Map, SOM) [23] az eddig ismertetett eljárásoktól merőben különböző dimenzióredukciós eljárás. Napjainkban már számos dimenziócsökkentési eljárás alapját adják a neurális hálózatok. A SOM szintén egy ilyen mesterséges neurális hálózat, amely abban különbözik más mesterséges neurális hálózatoktól, hogy a bemeneti tér topológiájának megőrzése céljából szomszédossági függvényt is alkalmaz.



(a) metrikus MDS *(b) Sammon-leképezés*

3.3. ábra. Az iris adathalmaz 2-dimenziós reprezentációja metrikus MDS és Sammon-leképezés eredményeképpen

A SOM neuronjai topológiailag rendezett rácsban helyezkednek el, melyek közül legelterjedtebb a 3.4 ábrán szemléltetett négyzetes és a hexagonális struktúra. A hálózat topológiáját a neuronok között definiált szomszédsági függvény határozza meg. A hálózatban található neuronokhoz tartozik továbbá egy-egy D-dimenziós, tehát a vizsgált objektumtér dimenziójával azonos dimenziójú *referenciavektor* (*súlyvektor*) is. A SOM működési filozófiájának alapját az adja, hogy az algoritmus a rácsban közel elhelyezkedő neuronoknak olyan objektumokat feleltet meg, amelyek a vizsgált magas dimenzionalitású térben közel vannak egymáshoz.



3.4. ábra. Hexagonális és négyzetes SOM struktúra

A Kohonen-féle önszerveződő hálózat alkalmazása két fő fázisra különíthető el. Az első fázisban a hálózat inicializálása és tanítása történik, majd a második fázisban a betanított hálózat felhasználása következik.

A SOM tanítási fázisának megelőző lépése a *hálózat inicializálása*. Ez a kiindulási hálózat kialakítását jelenti, amely a neuronok számának meghatározását és a topológia kiválasz-

tását, valamint a kezdeti súlyvektorok inicializálását foglalja magában. A neuronok számát érdemes viszonylag nagynak választani, hogy a reprezentáció minél részletesebb legyen, de arról sem szabad elfeledkezni, hogy a neuronok számának növelésével a számítási költség (amely elsősorban a tanítási fázisban jelentős) szintén nő. A súlyvektorok inicializálása lehet véletlenszerű, illetve alapulhat a vizsgált objektumhalmazból történő véletlen mintakiválasztáson is. Minél jobb inicializálást sikerül megvalósítani, annál hamarabb konvergál a kívánt eredményhez a hálózat a tanítási fázis során.

A *hálózat tanítása* egy iteratív folyamat, melynek minden egyes ciklusában kiválasztunk a bemeneti objektumhalmazból véletlenszerűen egy objektumot, majd megkeressük a neuronháló azon neuronját, melynek távolsága a kiválasztott objektumhoz képest az uklideszi távolságfüggvény alapján a legkisebb. Ez a neuron lesz a legjobban illeszkedő neuron, vagyis a BMU (Best Matching Unit). Ezután a BMU és topológia szomszédjainak súlyvektorait oly módon módosítjuk, hogy azok még közelebb kerüljenek a véletlenszerűen kiválasztott objektumhoz. A változás kiterjesztettsége és mértéke időben csökkenő tendenciát mutat. A BMU és a szomszédos neuronok módosítása a következő formula alapján történik:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + h_{BMU,i}(t)\left[\mathbf{x}(t) - \mathbf{w}_i(t)\right], \tag{3.7}$$

ahol t az idő, $\mathbf{w}_i$ az i. neuron a rácsban, $\mathbf{x}(t)$ a t időpillanatban a bemeneti objektumhalmazból véletlenszerűen kiválasztott objektum, $h_{BMU,i}(t)$ a BMU körüli szomszédossági függvény, amely a BMU szomszédságát és az ahhoz rendelt tanulási rátát határozza meg a t időpillanatban. Ahogy haladunk az iterációk sorában előre a neuronokat érintő változás egyre kisebb, hiszen azok egyre jobban idomulnak bemeneti objektumhalmazhoz. A tanítási folyamat aktuális fázisának minőségi értékelése például a következő függvény alapján határozható meg:

$$E_{SOM} = \frac{1}{N}\sum_{i=1}^{N} \parallel \mathbf{x}_i - \mathbf{w}^i_{BMU} \parallel, \tag{3.8}$$

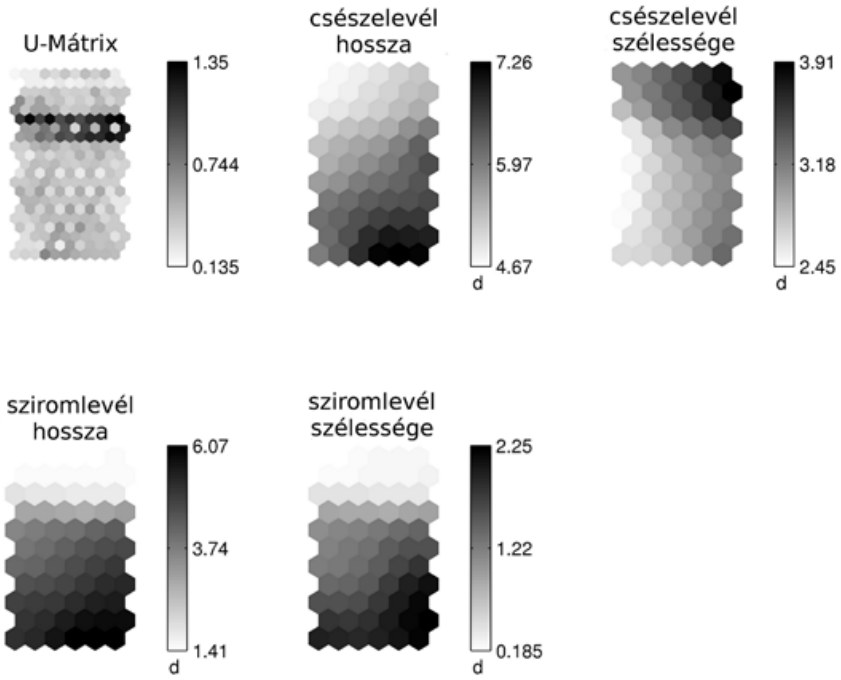
ahol N a leképezendő objektumok száma és $\mathbf{w}^i_{BMU}$ az $\mathbf{x}_i$ objektumhoz legjobban illeszkedő neuron.

A mellékletben található `som_demo.avi` fájl szemléletesen mutatja be, hogy a SOM neuronhálója a tanítási fázis során a kezdeti szabályos struktúrából kiindulva hogyan veszi fel a mintaobjektumok struktúráját. A demonstrációs fájlban láthatjuk, hogy jelen esetben Magyarország települései képezik a bemeneti objektumhalmazt, a SOM inicializálásakor pedig négyzetes topológiai struktúrát határoztunk meg.

A neuronhálózat a tanítását, edzését követően többféleképpen használható. A betanított neuronháló alkalmas például új példaobjektumok gyors osztályozására a hozzá legközelebb eső BMU alapján.

A Kohonen-féle önszerveződő hálózatok értelmezésében nagy szerephez jutnak a SOM-hoz kapcsolódó különféle vizualizációs eszközök. Legelterjedtebb ábrázolási mód az úgynevezett *U-mátrix* (Unified distance matrix), amely a többdimenziós adatok SOM-on alapuló 2D-s ábrázolása. Az U-mátrix az egyes neuronoknak a szomszédos neuronoktól számított távolságának egységre normált átlagát jeleníti meg szürkeárnyalatos módon. A konvencionális jelölés szerint ha az átlagos távolság kicsi, akkor világos, ha az átlagos távolság nagy, akkor sötétebb árnyalatot szokás használni. Ezáltal az U-mátrix alkalmas az egymástól távol álló objektumcsoportok megkülönböztetésére oly módon, hogy a világos foltok magukat

az objektumcsoportokat reprezentálják, a sötét vonalak pedig ezen objektumcsoportokat különítik el egymástól. A szürkeárnyalatos színezés mellett használatos még a magassághoz kapcsolódó ábrázolás is, ahol az átlagos távolság értékét nem színezve, hanem egy harmadik dimenzióban magasságként ábrázolják. További ábrázolási lehetőség a *komponenstérképek* használata, amely ábrázolási forma során kiválasztunk egy változót (attribútumot), s a SOM neuronjain ezen attribútum értékét jelenítjük meg színezéssel, vagy magasság formájában. E két gyakran együtt használt ábrázolási formát mutatja be a 3.5 ábra. Természetesen léteznek egyéb ábrázolási módok is, így például az önszerveződő hálózatot gyakran ábrázolják főkomponens analízis, vagy Sammon-leképezés segítségével is, ezáltal részletesebb betekintést nyerve a magas dimenzionalitású adatok struktúrájába.



3.5. ábra. Az iris adathalmaz U-mátrixa és komponenstérképei

3.6. Topológia alapú eljárások

A következőkben bemutatásra kerülő topológia alapú dimenziócsökkentési eljárások közös jellemzője, hogy elsődleges céljuk a magas dimenzionalitású adathalmaz topológiájának feltárása, s az alacsony dimenzionalitású prezentáció oly módon történő megadása, hogy ez a topológia a leképezés során megmaradjon.

A topológia alapú algoritmusok az alacsony szintű prezentáció kialakítása során gyakran támaszkodnak az objektumokkal szomszédos objektumokra. Ebben a felfogásban kardinális

kérdés, hogy mit értünk a szomszédosság fogalma alatt. A szomszédosság meghatározására leginkább elterjedt megközelítési módok a *k-legközelebbi szomszéd* és az ε *sugarú környezet* elvén alapulnak:

- *k-legközelebbi szomszéd*: Ahhoz, hogy egy objektum k-legközelebbi szomszédjait megadhassuk, ki kell számolni a páronkénti távolságokat az összes objektumpárra vonatkozóan. Egy $\mathbf{x}_i$ objektum k-legközelebbi szomszédjainak meghatározásához rendezzük az összes többi objektumot az $\mathbf{x}_i$-hez kiszámított távolságok alapján növekvő sorrendbe, majd válasszuk a lista első k objektumát. Ők lesznek az $\mathbf{x}_i$ k-legközelebbi szomszédjai.

- ε-*szomszédosság*: Az ε-szomszédossági elv alapján azon $\mathbf{x}_j$ objektumokat tekintjük szomszédosnak $\mathbf{x}_i$ objektumhoz viszonyítva, amelyek távolsága $\mathbf{x}_i$-től kisebb, mint egy szabadon választott tetszőlegesen kis ε, vagyis $d(\mathbf{x}_i - \mathbf{x}_j) < \varepsilon$, ahol d általában az euklideszi távolságot jelöli.

Mint a következőkben bemutatásra kerülő algoritmusok esetén is látni fogjuk, a szomszédsági kapcsolatok különféleképpen járulhatnak hozzá a magas dimenzionalitású objektumok alacsony dimenzióban történő szemléltetéséhez.

3.6.1. Isomap

A Tenenbaum által 2000-ben publikált Isomap algoritmus az objektumpárok *geodéziai távolságán* alapszik. Egy tetszőleges struktúrát képező adatsokaság két pontja között a geodéziai távolságon azon legrövidebb útnak a hosszát értjük, amely út „nem lép” ki a sokaságból. Szemléltetésképpen tekintsük a 3.6(a) ábrán látható Swiss roll struktúrájú adathalmazt. Az ábrán látható objektumhalmazból válasszunk tetszőlegesen egy kék és egy piros színnel jelölt objektumot. Látható, hogy a kék és piros színnel jelölt objektumok ugyan a 3-dimenziós térben közel helyezkednek el egymáshoz, azonban ha a rejtett struktúrát szemléljük, és „kiterítjük” azt a síkba, akkor azt vesszük észre, hogy ezen objektumok valójában nagyon is távol találhatók egymástól. A geodéziai távolság alkalmazása tehát az objektumhalmaz rejtett struktúrájának pontosabb felismeréséhez járul hozzá.

Az *Isomap algoritmus* [36] egy nemlineáris dimenziócsökkentési eljárás, amely azon alapfeltevésből indul ki, hogy a D-dimenziós térben elhelyezkedő adathalmaz objektumai egy kisebb, d dimenzionalitású struktúra mentén helyezkednek el ($d \ll D$). Az algoritmus célja ezen belső struktúra megőrzése és megjelenítése a d-dimenziós térben. Ezen cél megvalósításához az Isomap algoritmus az objektumpárok távolságát az objektumok geodéziai távolságaként határozza meg, majd a többdimenziós skálázás módszerét hívja segítségül az alacsony dimenzionalitású vizualizáció létrehozásához. Az Isomap algoritmus a következő 3 fő lépésre tagolható:

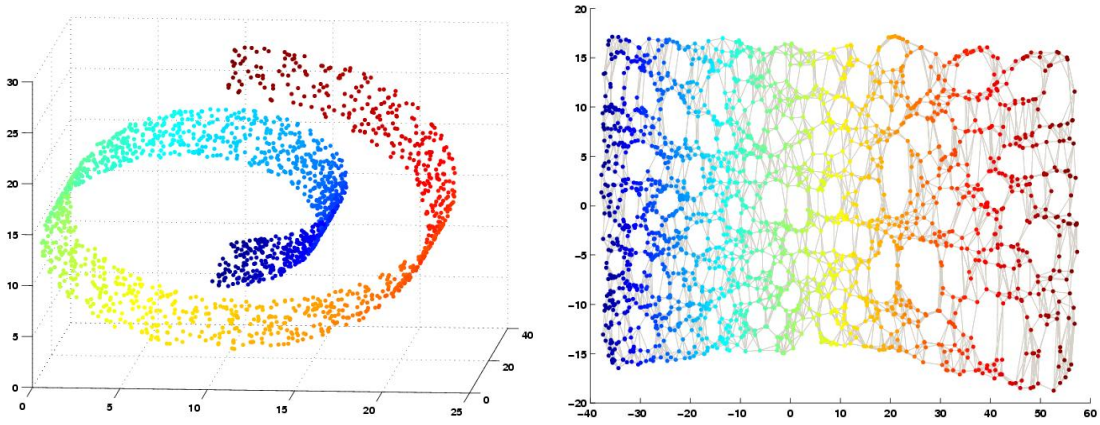
1. lépés: Az objektumok szomszédsági gráfjának kialakítása a k-legközelebbi szomszéd (vagy az ε környezet elve) alapján.

2. lépés: A geodéziai távolság kiszámítása minden objektumpárra az előző lépésben kialakított gráf alapján.

3. lépés: A d-dimenziós megfeleltetés kialakítása az MDS algoritmus segítségével.

Az algoritmus első lépésének eredményeképpen egy súlyozott gráf jön létre, ahol az élek az objektumok közti szomszédsági relációkat adják meg, az élek súlya pedig megegyezik azon csomópontok euklideszi távolságával, melyeket összeköt. A második lépésben a geodéziai távolságok kiszámítása történik oly módon, hogy két objektum geodéziai távolsága egyenlő az őket összekötő legrövidebb út hosszával, azaz a gráfban az objektumokat reprezentáló csomópontokat összekötő legrövidebb út éleihez tartozó súlyok összegével. Az algoritmus a harmadik lépésben a d-dimenziós reprezentáció kialakításhoz a metrikus MDS algoritmust használja oly módon, hogy az MDS algoritmus nem az objektumok euklideszi távolságát veszi alapul, hanem a 2. lépésben kiszámolt geodéziai távolságokat. Az algoritmus feltételezi, hogy az objektumok egyetlen, összefüggő adatsokaságot alkotnak. Amennyiben azonban az objektumok diszkrét csoportokat képeznek, akkor az MDS nem alkalmazható, hiszen az 1. lépés eredményeképpen több, egymással nem kapcsolódó részgráf is létrejöhet. A probléma megoldásaként a Wu és Chan által javasolt [40] részgráf-összekötési módszert alkalmazhatjuk.

Az algoritmus futási eredményének szemléltetéséhez tekintsünk egy $N = 2000$ adatpontból álló Swiss roll objektumhalmazt, melyet a 3.6 ábra (a) része szemléltet. Az ábrán az objektumok struktúrán belüli elhelyezkedését színezéssel szemléltettük. A 3.6 ábra (b) részén az Isomap leképezés 2-dimenziós eredményét tekinthetjük meg $k = 7$ legközelebbi szomszéd választása esetén. Az ábrán az egyes objektumokat összekötő szürke vonalak a k-legközelebbi szomszédsági kapcsolatokat jelölik. Az eredményképpen létrejött alacsony dimenzionalitású reprezentációban látható, hogy $k = 7$ választás esetén az algoritmus felismerte az objektumhalmaz belső struktúráját.



(a) Swiss roll struktúrájú objektumhalmaz *(b) Isomap leképezés eredménye ($k = 7$)*

3.6. ábra. Egy Swiss roll struktúrájú adathalmaz és 2-dimenziós reprezentációja az Isomap leképzés eredményeképpen $k = 7$ esetén

Az Isomap algoritmus gyenge pontja a k db legközelebbi szomszéd megválasztásának kérdése. Az imént bemutatott 2000 objektumot tartalmazó Swiss roll struktúra leképezéséhez a legközelebbi szomszédok számának $k = 7$ választása egy optimális választást jelent,

azonban amennyiben az algoritmus futása során a legközelebbi szomszédok számát ennél kisebbre választjuk, akkor a leképzés eredménye már nem mutatja ilyen egyértelműen a feltárandó struktúrát. Az Isomap algoritmus kapcsán további alkalmazási nehézséget jelent a zajos, szennyezett adatok jelenléte. Ennek oka az, hogy a nem megfelelően választott k bemeneti paraméter esetén egy-egy, a sokaságból kiugró adat is számos geodéziai távolság értékét módosíthatja oly módon, hogy az eredményül kapott d-dimenziós reprezentáció jelentős torzulást szenved.

3.6.2. Lokálisan lineáris beágyazás

A *lokálisan lineáris beágyazás* (Locally Linear Embedding, LLE) [30] módszerét Roweis és Saul az Isomap algoritmussal megközelítőleg azonos időben publikálták (2000), s számos esetben jobb eredményt nyújt mint az Isomap algoritmus. Eltérően az Isomap algoritmustól, az LLE nem igényli az egymástól távol eső adatpontok távolságának meghatározását, hanem minden egyes objektumot csak a maga kis környezetében vizsgál. A lokálisan lineáris beágyazás alapgondolata az, hogy a magas dimenzionalitású térben nem lineárisan beágyazott adatsokaság kis szeletét (foltját) nézve lokálisan lineárisnak tekinthető. Az eljárás a magas dimenzionalitású térben minden egyes objektumot a szomszédok lineáris kombinációja segítségével határoz meg, majd ezen lokális lineáris leírásokat használja fel a kisebb dimenzionalitású eredménytér kialakításához.

Az LLE algoritmus bemeneti paramétere az objektumokat leíró $N \times D$ dimenziós $\mathbf{X}$ adatmátrix, a leképezés eredményterének d dimenziója, s a szomszédok számát meghatározó k érték, melynek a kimeneti dimenziónál legalább eggyel nagyobb értéknek kell lennie ($k \geq d+1$). Az LLE algoritmus a következő három fő lépésre tagolható:

1. lépés: Keressük meg minden egyes $\mathbf{x}_i$ vektornak a k db legközelebbi szomszédját.

2. lépés: Keressük meg azt a $\mathbf{W}$ súlymátrixot, amely által legjobban rekonstruálhatók az $\mathbf{x}_i$ objektumok a szomszédjaik által, vagyis az eredeti és a rekonstruált objektumok távolságának négyzetes eltérése minimális. A rekonstrukciós hiba a következőképpen definiálható:

$$E(\mathbf{W}) = \sum_{i=1}^{N} \| \mathbf{x}_i - \sum_{j \neq i} w_{ij} \mathbf{x}_j \|^2, \tag{3.9}$$

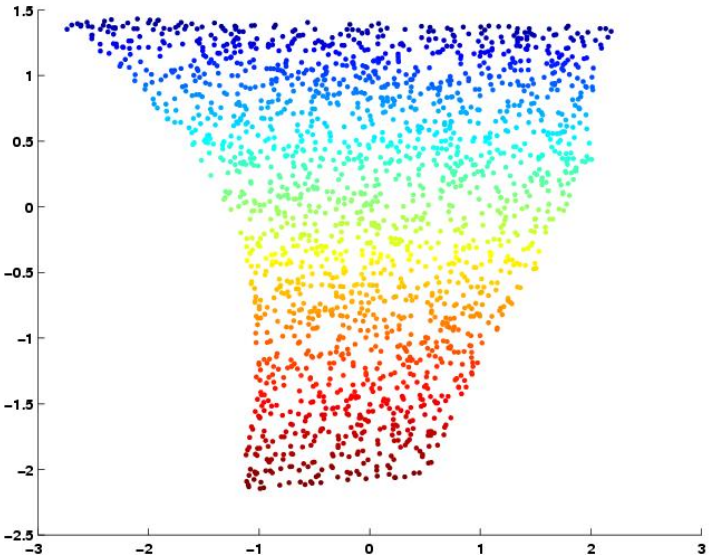
ahol w_{ij} értéke csak az $\mathbf{x}_i$ k-legközelebbi szomszédjai esetén nem 0, és minden i-re $\sum_j w_{ij} = 1$.

3. lépés: Keressük azokat az $\mathbf{y}_i$ d-dimenziós objektumokat ($i = 1, 2, \ldots, N$), amelyek minimalizálják az előző lépésben kiszámolt súlyok rekonstrukciós hibáját:

$$\Phi(\mathbf{Y}) = \sum_{i=1}^{N} \| \mathbf{y}_i - \sum_{j \neq i} w_{ij} \mathbf{y}_j \|^2, \tag{3.10}$$

Látható, hogy az algoritmus második és harmadik lépésében adott költségfüggvények hasonlóak, azonban míg a második lépésben a megfelelő súlyokat keressük és az objektumvektorok ismertek, addig a 3. lépésben a súlyvektorok adottak és az optimalizációs probléma

a megfelelő alacsony dimenzionalitású adatpontok keresésére van kihegyezve. Mivel ezen optimalizáció tisztán algebrai úton, egy $N \times N$-es ritka mátrix sajátérték problémaként megoldható, ezért gyorsabb futást eredményez mint az Isomap algoritmus alkalmazása. Azonban mivel az LLE csak a lokális struktúrákra helyezi a hangsúlyt, ezért a leképezésben globálisan tekintve váratlan torzulások fordulhatnak elő. Ilyen torzulást figyelhetünk meg a 3.7 ábrán is, amely a 3.6(a) ábrán bemutatott Swiss roll struktúrájú adathalmaznak az LLE algoritmus által létrehozott 2-dimenziós vizualizációja $k = 12$ legközelebbi szomszéd alkalmazása esetén. Kevesebb k-legközelebbi szomszéd választása esetén az alacsony dimenziójú reprezentáció még nagyobb torzulásokat szenved.



3.7. ábra. Egy Swiss roll struktúrájú adathalmaz ($N = 2000$) 2-dimenziós LLE vizualizációja ($k = 12$)

4. fejezet

Adatbányászat

4.1. Az adatbányászat fogalma, feladatköre

Az adatbányászat viszonylag rövid, pár évtizedes történeti múltra tekint vissza. A relációs adatmodell 70-es évektől kezdődő térhódítása, az információs eszközök minél nagyobb mértékű elterjedése és a tárolókapacitások növekedése révén a 80-as, 90-es évekre az adatbázisrendszerek jelentős teret nyertek a hétköznapi élet szinte minden területén (pl. kereskedelem, telekommunikáció, pénzügy, egészségügy, ipar). Ennek eredményeképpen az adatbázisokban hatalmas mennyiségű adat halmozódott fel, melyek számos értékes információt rejtettek magukban. Ezen adatok elemzése a hagyományos adatbázis-kezelő rendszerekbe beépített egyszerűbb statisztikai függvények által már nem bizonyult kielégítőnek. A felhalmozódott adatmennyiség jogosan vetette fel a szakértők azon igényét, hogy a sok adatból mélyebb, árnyaltabb összefüggéseket is feltárjanak. Ennek eredményeképpen a 80-as évek végén kialakult egy új tudományág, az adatbányászat, mely számos olyan adatelemzési módszert rejt magában, melyek célja a nagy mennyiségű adathalmazban megbúvó rejtett információk feltárása.

Az *adatbányászat* fogalmának meghatározása nem egyszerű feladat, tekintve, hogy az általa felölelt technológiák rendkívül szerteágazóak. A fogalom definiálása éppen ezért elsősorban az adatbányászati célok megfogalmazása mentén lehetséges. Legszélesebb körben talán az a meghatározás terjed el, mely szerint az adatbányászat nem más, mint olyan nem triviális, érvényes, korábban ismeretlen információk kinyerése a nagy adathalmazokból, melyek várhatóan hasznosnak bizonyulnak. Lényeges kiemelni, hogy a hagyományos statisztikai elemzésekkel, adatbázis lekérdezésekkel, illetve aggregációkon alapuló elemzésekkel végzett információkinyerés nem tartozik az adatbányászat témakörébe. Az adatbányászat ennél összetettebb, automatikus és fél-automatikus elemzői módszereket foglal magában. Az összetett jellege abból származik, hogy az adatbányászat egy multidiszciplináris tudományterület, mely az adatbázis-technológia, a gépi tanulás, a statisztika, a vizualizáció és egyéb diszciplínák mentén alakult ki.

Mivel az adatbányászati módszerekkel elemezhető adatok típusa széleskörű, ezért az adatbányászati módszerek felhasználási területe is rendkívül szerteágazó. Míg az adatbányászati módszerek kialakulásában elsősorban az üzleti élet kihívásai és a tudományos kérdések megválaszolásának igénye játszott kiemelkedő szerepet, addig napjainkra a beépített adat-

bányászati algoritmusok elengedhetetlen részévé váltak számos mindennapi informatikai alkalmazásnak (pl. pénzügyi szféra, telekommunikáció, biztonsági rendszerek), s emellett az alapkutatások terén is jelentős szerephez jutottak. Csak pár példát említve a lehetséges alkalmazási területekről, adatbányászati módszereket használnak például a direkt a marketing területén, azon célból, hogy az ügyfeleket a számukra leginkább illeszkedő szolgáltatásokkal kereshessék fel, a kereskedelemben az áruk értékesítésével kapcsolatos összefüggések feltárásának céljából, a csalások detektálásában (pl. biztosítási csalások, térfigyelő kamerák), a web tartalmának szűrése, keresése, és elemzése során, illetve számos orvostudományi kutatás (pl. génkutatás) terén is.

A szerteágazó feladatkörből és adattípusokból adódóan az adatbányászati algoritmusok is széles skálán mozognak. A történeti fejlődés során azonban kialakultak azok a főbb problémakörök, melyek megoldása leginkább jellemző az adatbányászati alkalmazásokra. Ezen főbb adatbányászati feladatcsoportok a következők:

- *Gyakori elemhalmazok és asszociációs szabályok feltárása*: A gyakori elemhalmazok feltárása során azokat az elemeket keressük, amelyek gyakran fordulnak elő együtt. A gyakori előfordulások alapján olyan asszociációs szabályok alkothatók, melyek az együttes előfordulások szabályszerűségét, s annak valószínűségét írják le. Ezen adatbányászati problémakör tipikus példája az úgynevezett bevásárlókosár analízis, mely elemzés során arra keressük a választ, hogy a vevők mely termékeket vásárolják gyakran együtt. A termékek együttes vásárlása számos információt szolgáltat a kereskedőknek arra vonatkozóan, hogy hogyan tehetik az egyes termékeiket még vonzóbbá, s hogyan növelhetik a bevételeiket.

- *Osztályozás és előrejelzés*: Az *osztályozás* folyamata egy irányított tanulás, ahol a megfigyelt objektumokat egymástól elkülönülő osztályokba soroljuk, majd arra keressük a választ, hogy az ismert tulajdonságok mentén hogyan lehet az objektumok osztályba való tartozását meghatározni. Az így megalkotott leírások a későbbiekben a felhasználókat az új objektumok osztályokba történő besorolásában segíthetik. Amíg az osztályozás az objektumoknak kategóriákba történő besorolását teszi lehetővé, addig az *előrejelzés* folytonos értékek prognózisát jelenti. Az osztályozásnak számos tipikus alkalmazási lehetősége van, így például gyakran alkalmazott módszer a különféle ügyfélcsoportok jellemzésére, majd új ügyfelek csoporthoz való tartozásának becslésére.

- *Csoportosítás*: A csoportosítás egy irányítás nélküli tanulási forma, melynek célja, hogy a vizsgált objektumhalmazban olyan csoportokat találjon, ahol egy adott csoporton belül az objektumok nagyon hasonlóak egymáshoz, viszont más csoport objektumaitól nagy mértékben különböznek. A csoportosítás tipikus példája a piacszegmentálás, amikor a szolgáltatók az ügyfelek homogén csoportjait keresik azon célból, hogy számukra testreszabott szolgáltatásokat nyújthassanak. A csoportosításnak egy speciális alkalmazási területe a csoportoktól jelentősen eltérő objektumok feltárása. Ezen objektumok például eszközök hibás működésére, csalásokra hívhatják fel az elemzők figyelmét.

A felsorolás természetesen nem lehet teljes, hiszen számos esetben olyan testreszabott adatbányász tevékenységet kell végezni, amely egyik csoportba sem sorolható.

A továbbiakban az imént felsorolt leggyakoribb feladatköröket tekintjük át elemzői szemszögből kissé részletesebben. Miután minden egyes problémakörről önállóan is akár egy-egy könyvet lehetne írni, ezért az a főbb elemzési kérdések áttekintése mellett csak a leginkább elterjedt, leggyakrabban alkalmazott algoritmusok rövid ismertetésére szorítkozunk. Az adatbányászatról részletesebb ismeretekre vágyó Olvasó figyelmébe az alábbi irodalmakat ajánljuk: [1], [6], [13].

4.2. Gyakori elemhalmazok és asszociációs szabályok feltárása

A gyakori elemhalmazok feltárásának igénye először a kereskedelemben fogalmazódott meg. Miután a különféle napi feladatokat ellátó alkalmazások (pl. vonalkódleolvasók) rendre rögzítik, hogy az egyes vásárlások alkalmával mit tartalmaznak a vásárlói kosarak, ezért jogosan merül fel az a kérdés, hogy mik azok a termékek, amiket a vevők gyakran együtt vásárolnak. Ezt hívják az adatbányászatban bevásárlói kosár analízisnek. Az együtt vásárolt termékek ismeretében a kereskedők különféle célirányos akciókkal, a katalógusok ilyen szempontokat is figyelembe vevő összeállításával, illetve a termékek bolton belüli elhelyezésének tudatos megválasztásával hatékonyan növelhetik bevételeiket. Napjainkra a gyakori elemhalmazok keresése már nem csupán a bevásárlói kosarak elemzése során használatos módszer, hanem számos egyéb alkalmazása létezik. Így például hasonló elemzéseket végzünk akkor is, amikor a gyakorta együtt meglátogatott weboldalak kapcsolatát kívánjuk feltárni.

4.2.1. Gyakori halmazok, asszociációs szabályok

A feladatunk tehát megkeresni a gyakorta együtt előforduló elemek halmazát. De mit is jelent ez pontosan? Adott $I = \{i_1, i_2, \ldots, i_m\}$ az elemek halmaza és $T = \{t_1, t_2, \ldots, t_n\}$ a tranzakciók halmaza, ahol minden $t_i \subseteq I$. Legyen $X \subseteq I$ egy elemhalmaz, s azt mondjuk, hogy a $t_i \in T$ tranzakció tartalmazza X-et, ha $X \subseteq t_i$. Egy adott *elemhalmaz (X) gyakorisága* a T tranzakciók egészére vonatkozóan a következőképpen határozható meg:

$$gyakoriság(X) = \frac{|\{t_i | X \subseteq t_i, t_i \in T\}|}{|T|} \tag{4.1}$$

Egy tetszőleges X *elemhalmaz gyakori*, ha gyakorisága a T tranzakciók tekintetében nagyobb, mint egy, a felhasználó által meghatározott minimális gyakoriság. Jelöljük ezt a felhasználó által választott minimális gyakoriságot σ-val. Ekkor X gyakori, ha

$$gyakoriság(X) \geq \sigma \tag{4.2}$$

A gyakori elemhalmazokra vonatkozóan megállapítható egy nagyon fontos tulajdonság, miszerint a gyakori elemhalmaz bármely részhalmaza szintén gyakori elemhalmaz. Ezen elvet nevezzük *Apriori elvnek*.

A gyakori elemhalmazok ismeretében már megalkothatjuk azon szabályokat, melyek az elemek egy halmazának korrelációját írják le az elemek egy másik halmazával. Az ilyen

szabályokat *asszociációs szabályoknak* nevezzük. Az asszociációs szabályok formálisan $X \Longrightarrow Y$ alakban adhatók meg, ahol $X \subset I$, $Y \subset I$ és $X \cap Y = \emptyset$. X-et a *szabály törzsének*, vagy *előzményének*, Y-t pedig *fejnek*, vagy *következménynek* nevezzük. X és Y lehetnek összetett kifejezések is. Attól függően, hogy egy, vagy több attribútumra vonatkozóan fogalmaznak-e meg leírásokat, *egy- vagy többdimenziós asszociációs szabályoknak* nevezzük őket. Tekintsük a következő két példát:

$$vásárol(\text{„}pelenka\text{”}) \Longrightarrow vásárol(\text{„}sör\text{”})[s = 0,5\%, c = 60\%] \tag{4.3}$$

$$kor(\text{„}40-49\text{”}) \wedge jövedelem(\text{„}400e-500e\text{”}) \Longrightarrow vásárol(\text{„}VW Passat\text{”}) \quad [s = 3\%, c = 70\%], \tag{4.4}$$

A 4.3 asszociációs szabály egydimenziós, mivel csak a vásárlás attribútum értékeit vizsgálja, a 4.4 asszociációs szabály pedig többdimenziós, mivel a kor, a jövedelem és a vásárlás attribútumok mentén fogalmazza meg az összefüggéseket.

Mint a fenti két példából is láthatjuk, az *asszociációs szabályok erősségét* két százalékos formában megadott értékkel szokás jellemezni. Az elsőként megadott érték a szabály támogatottságát (támaszát), a második a megbízhatóságát, vagyis a szabály bizonyosságát (konfidenciáját) jellemzi. A *szabály támogatottsága* annak a valószínűsége, hogy egy tranzakció tartalmazza $X \cup Y$-t. A *szabály bizonyossága* feltételes valószínűsége annak, hogy azon tranzakciók, amik tartalmazzák X-et tartalmazzák Y-t is. Formálisan tehát a támogatottság (s) és a bizonyosság (c) a következőképpen definiálható:

$$s(X \Longrightarrow Y) = gyakoriság(X \cup Y) \tag{4.5}$$

$$c(X \Longrightarrow Y) = P(Y|X) = \frac{gyakoriság(X \cup Y)}{gyakoriság(X)} \tag{4.6}$$

Felmerül a kérdés, hogy milyen asszociációs szabályok érdekesek az elemzők és a szakértők számára? Elsősorban azon asszociációs szabályok, melyek erősek, tehát támogatottságuk és bizonyosságuk is meghalad egy-egy, a felhasználó által definiált *minimális támogatottsági* és *minimális bizonyossági küszöböt*. Az ilyen szabályokat nevezzük *érvényes (erős) asszociációs szabályoknak*.

Az érvényes asszociációs szabályok feltárása kétlépcsős folyamat. Első lépésben a gyakori elemhalmazok megtalálása a cél, majd az érvényes asszociációs szabályok a gyakori elemhalmazokból az Apriori elv alapján a következőképpen hozhatók létre. Miután tudjuk, hogy a gyakori elemhalmaz bármely részhalmaza gyakori, ezért minden gyakori X elemhalmazra generáljuk az összes valódi nem üres részhalmazát ($X_i, X_i \subset X$). X mindegyik nem üres részhalmazára alkossuk meg a $X_i \Longrightarrow (X - X_i)$ szabályt, majd vizsgáljuk meg, hogy bizonyossága nagyobb-e a minimális bizonyossági küszöbnél. Ha ez teljesül, akkor a vizsgált szabály érvényes.

Az asszociációs szabályok generálásának kulcsmozzanata tehát a gyakori elemhalmazok feltárása. Az erre a célra leggyakrabban alkalmazott algoritmusok rövid ismertetése a 4.2.3 fejezetben található. Mielőtt azonban rátérnénk az algoritmusok ismertetésére tekintsük át a szabályok kiértékelésére vonatkozó főbb alapelveket.

4.2.2. Asszociációs szabályok kiértékelése

Az asszociációs szabályok kiértékelése nem egyszerű feladat és számos buktatót rejt magában. Először is ki kell emelnünk, hogy az asszociációs szabály nem jelent oksági kapcsolatot, tehát nem jelenthetjük ki egyértelműen, hogy a szabály törzs része okozza a szabály fej részét. Az elemek együttes előfordulásának gyakorta egyéb külső oka van, így például a *cipőben alvás* $\Longrightarrow$ *fejfájás* esetén a fejfájást valószínűleg nem a cipő, hanem egyéb tényező (pl. másnaposság) okozza. Továbbá, a következmény rész megléte, vagy hiánya szintén nem befolyásolja az előzményt, hiszen ha egy *vásárol(„cipő")* $\Longrightarrow$ *vásárol(„cipőtisztító")* implikáció esetén a kereskedő megszűnteti a cipőtisztítók forgalmazását, az nem jelenti azt, hogy ezután cipőket sem fog tudni eladni.

Az asszociációs szabályok generálása során a megfelelő minimális bizonyossági és támogatottsági küszöbszintek megválasztása fontos kérdés. Amennyiben túl magasra tesszük ezeket a küszöbértékeket, akkor lehet, hogy érdekes szabályokat veszítünk el, ellenkező esetben pedig számos érdektelen szabályt is eredményül kaphatunk. Mindemellett fontos kiemelni, hogy nem minden érvényes asszociációs szabály érdekes. Gondoljunk csak arra például, hogy az asszociációs szabályok az elemek kapcsolatát különféle általánossági szinten írhatják le. Példaként tekintsük a következő két szabályt:

$$vásárol(„tej") \Longrightarrow vásárol(„kenyér")[s = 8\%, c = 7\%] \tag{4.7}$$

$$vásárol(„2,8\%\text{-}os\ tej") \Longrightarrow vásárol(„fehér\ kenyér")[s = 2,7\%, c = 7,1\%] \tag{4.8}$$

Mint látjuk, a 4.7 és a 4.8 szabályok eltérő részletezettségi szinten írják le az elemek kapcsolatát. Ha mindezek mellé még azt is tudjuk, hogy a 2,8%-os tej eladások körülbelül egyharmadát teszik ki az összes tej eladásának, akkor láthatjuk, hogy a második szabály nem hordoz új információt az elemzők számára.

Az asszociációs szabályokra meghatározott bizonyossági érték és a támogatottság ismerete nem elegendő a szabály értékességének meghatározásához, és félrevezető is lehet. Példaként tekintsük a következőt: egy felmérés szerint az emberek 80%-a fogyaszt kávét, 20%-a teát, s 15%-uk mindkettőt. Ez alapján a *fogyaszt(„tea")* $\Longrightarrow$ *fogyaszt(„kávé")* szabály bizonyossága 75%. A magas bizonyossági szint értékes szabályt sugall. A probléma csupán az, hogy nem feledkezhetünk meg arról, hogy a kávéfogyasztók aránya eleve 80%. Összevetve a két adatot az a sejtésünk támadhat, hogy a tea fogyasztása negatívan befolyásolja a kávéfogyasztást. Ez így is van, s a probléma forrása abból fakad, hogy a bizonyossági mérték nem veszi figyelembe a következmény gyakoriságát. Ezt a hiányosságot kiküszöbölendő az adatbányász alkalmazások a feltárt szabályok esetében gyakorta megadnak egy harmadik mértéket is, az úgynevezett *Lift mutatót*. A Lift mutató már figyelembe veszi a következmény rész gyakoriságát is, s a következőképpen számítható ki:

$$Lift(X \Longrightarrow Y) = \frac{gyakoriság(X \cup Y)}{gyakoriság(X)gyakoriság(Y)} \tag{4.9}$$

A Lift értéke 0 és végtelen közötti értékeket vehet fel. Ha a Lift értéke nagyobb, mint 1, akkor az azt jelenti, hogy a szabály törzse és feje gyakrabban fordulnak elő együtt, mint az várható lenne. Ebben az esetben a szabály törzse pozitív hatással van a szabály fejének előfordulására.

Ha a Lift értéke kisebb mint 1, akkor az azt jelenti, hogy az előzmény és a következmény ritkábban fordulnak elő, mint az várható lenne, tehát az előzmény negatívan befolyásolja a következményt. Ha a Lift értéke 1 (illetve ahhoz nagyon közeli), akkor az a szabály fejének és törzsének függetlenségét jelenti, vagyis az előzmény résznek nincsen hatása a következmény részre. Ha az előző példában kiszámoljuk a Lift értékét, akkor $0,9375$-öt kapunk, ami a negatív hatást sejtő gondolatunkat hivatott alátámasztani.

Az asszociációs szabályok érdekességének kiértékelése egy rendkívül szerteágazó feladat, s ennek megfelelően az imént bemutatott érdekességi mutatók mellett számos egyéb érdekességi mutató is használatos. Ezen mérőszámokról részletesebb leírást az [1] és [6] irodalmakban találhatunk.

4.2.3. Gyakori algoritmusok

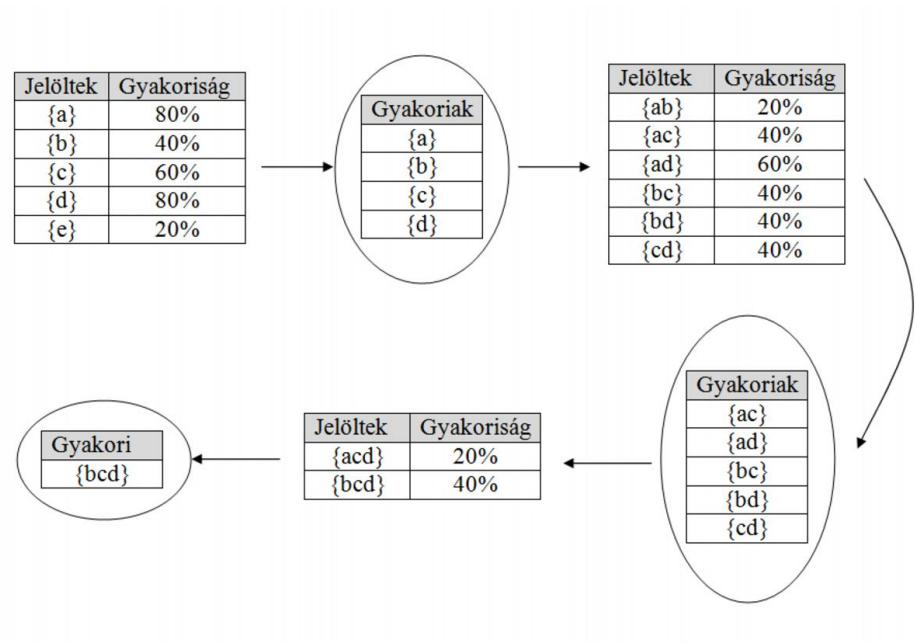
A vizsgált adathalmazok, melyekből gyakori elemhalmazokat szeretnénk kinyerni nagyon változatosak. Ezek az adatbázisok különbözhetnek például a vizsgált tranzakciók számosságában, a tartalmazott gyakori elemhalmazok számában, abban, hogy a legnagyobb gyakori elemhalmaz hány elemet tartalmaz, illetve mekkorák (hány eleműek) a leggyakoribb gyakori elemhalmazok. Különbözőségeket fedezhetünk fel továbbá abban is, hogy a gyakori elemhalmazok ritkák, vagy sűrűek-e, tehát elemszámuk hogyan viszonyul az összes elemek számához.

Számos olyan algoritmus létezik, amely a gyakori elemhalmazok adatbázisból történő kinyerését teszi lehetővé. Mivel azonban a vizsgált adathalmazok nagy eltéréseket mutatnak, ezért nem létezik olyan algoritmus, amely minden esetben a leghatékonyabb lenne. Összehasonlító vizsgálatok azt mutatják, hogy az Apriori, az ECLAT és az FP-growth algoritmusok rendelkeznek a legjobb mutatókkal ezen a területen. Míg a memóriahasználat terén a klasszikus Apriori algoritmus büszkélkedik kiemelkedő eredményekkel, addig a sebesség terén az ECLAT és az FP-growth algoritmusok emelkednek ki.

Az *Apriori algoritmus* a 4.2.1 fejezetben ismertetett Apriori elv alapján dolgozik. Ezen elvet felhasználva az algoritmus a $k+1$ elemű gyakori elemhalmazokat a k elemű gyakori elemhalmazokból állítja elő oly módon, hogy a k elemű gyakori elemhalmazokból $k+1$ elemű gyakori elemhalmaz jelölteket generál, majd minden jelöltre megvizsgálja, hogy gyakorisága nagyobb-e a küszöbszámként meghatározott minimális gyakoriságnál. Amennyiben igen, akkor a vizsgált $k+1$ elemű elemhalmaz már nem csak jelölt, hanem valóban gyakori elemhalmaz, ellenkező esetben viszont nem az, ezért az algoritmus törli, s a $k+2$ elemszámú jelöltek generálásakor már nem veszi figyelembe. Jellegét tekintve az Apriori algoritmus egy szélességi keresést hajt végre, melynek során előbb előállít minden k elemű gyakori elemhalmazt, majd csak ezt követően lép tovább a $k+1$ elemű gyakori elemhalmazok feltárásának irányába. Az algoritmus működésének kulcskérdése a jelöltek generálása. A jelöltek generálásához az algoritmus egy rendezést hoz létre az elemek listájában, s a tranzakciók elemeit sorba rendezi. Egy $k+1$ elemű jelölt generálása oly módon történik, hogy megkeresi azon k elemű rendezett gyakori elemhalmazokat, amelyek az első $k-1$ elemükben azonosak, s csak az utolsó elemükben térnek el. Ekkor a $k+1$ elemű jelölt két gyakori k elemű halmazból úgy áll elő, hogy az első $k-1$ eleme a közös rész lesz, a k. eleme azon k elemű gyakori halmaz utolsó eleme lesz, amely az elemek rendezett listájában előbbre található, a $k+1$-dik eleme

pedig a másik k elemű gyakori halmaz utolsó eleme lesz.

Példaként tekintsük a $T = \{\{cdab\},\{ac\},\{eda\},\{cbd\},\{ad\}\}$ kiindulási tranzakcióhalmazt. A minimális gyakorisági küszöb legyen 30%. Az elemek rendezése történjen lexikografikus rendezés alapján, tehát a következő sorrend adódik: $\{\{abcd\},\{ac\},\{ade\},\{bcd\},\{ad\}\}$. Az algoritmus működését a 4.1 ábrán követhetjük nyomon.



4.1. ábra. Példa az Apriori algoritmusra

Az Apriori algoritmus működésének gyengesége, hogy a jelöltek gyakoriságának ellenőrzésekor mindig végigolvassa az eredeti adatbázist. Ezen probléma megoldására született meg az *AprioriTID algoritmus*, amely az eredeti adatbázisból folyamatosan törli azon tranzakciókat és ritka elemhalmazokat, amelyek a későbbiek során már biztos nem járulnak hozzá az újabb gyakori elemhalmazok generálásához.

Az *ECLAT (Equivalence CLAss Transformation) algoritmus* az Apriori algoritmus szélességi keresésével szemben egy mélységi keresést hajt végre. Az ECLAT a jelöltek gyakoriságának ellenőrzése révén nyújt kimagasló eredményt, mivel a hagyományos vízszintes adattárolást átalakítja függőleges adattárolásra. De mit is jelent ez pontosan? Az adatbázisok a tranzakciókat leggyakrabban horizontális módon tárolják, azaz egy tranzakció azonosítóhoz rendelik hozzá a tranzakció által tartalmazott elemeket. A vertikális adattárolás ezzel ellentétben az egyes elemekhez rendeli azon tranzakciók azonosítóját, melyben az adott elemek előfordulnak. Ezeket a tranzakciók azonosítóiból álló listákat nevezzük TID listáknak, vagy TID halmazoknak. A 4.2 ábra egy példán keresztül szemlélteti a horizontális és vertikális adattárolás közti különbséget (az „e” elem TID listáját az ECLAT algoritmus működéséből fakadóan nem tüntettük fel).

Az ECLAT által alkalmazott vertikális adatszerkezet nagy mértékben hozzájárul az elemhalmazok gyakoriságának kiszámításához. Egy elemhalmaz gyakorisága a TID halmazok által oly módon számítható ki, hogy vesszük az elemhalmaz két részhalmazát, melyre igaz,

TID	Elemek
1	abcd
2	ac
3	ade
4	bcd
5	ad

(a) Horizontális szerkezet

Tranzakciók	Elemek: a	b	c	d
	1	1	1	1
	2	4	2	3
	3		4	4
	5			5

(b) Vertikális szerkezet

4.2. ábra. Horizontális és vertikális adattárolás

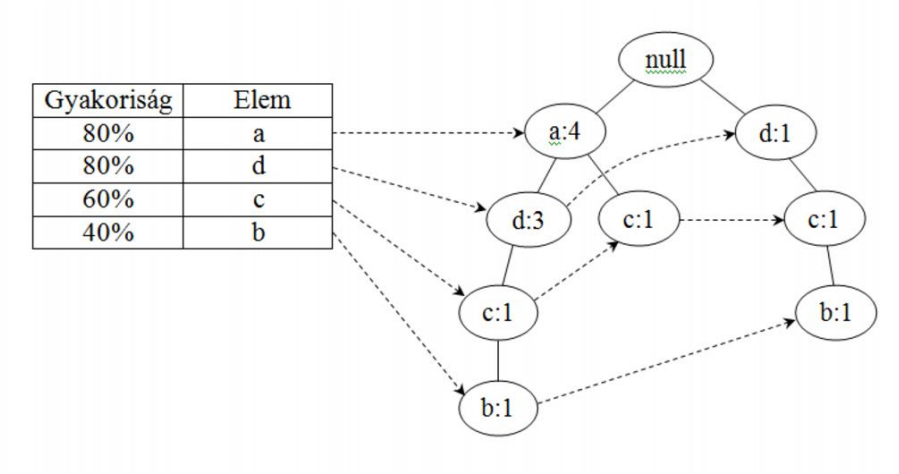
hogy uniójuk a vizsgált elemhalmazt adja, s ezen részhalmazok TID halmazainak metszetének számossága adja a vizsgált elemhalmaz előfordulásainak számát. A előfordulások számának ismeretében a gyakoriság egy osztást követően könnyen meghatározható.

A 4.2 ábra példáját követve az ECLAT algoritmus első lépésben kiszámolja az egyes elemek gyakoriságát az eredeti adatbázisból, majd a gyakori elemekhez létrehozza a TID halmazukat. Ha a példában feltételezzük, hogy a minimálisan elvárt előfordulások száma 2, akkor az „e” elem nem gyakori, ezért a neki megfelelő TID lista már létre sem jön. Ezt követően az algoritmus veszi az első elemet, vagyis az „a” elemet, s legenerálja hozzá a lehetséges 2 elemszámú gyakori halmaz jelölteket: $\{ab\}, \{ac\}, \{ad\}$. Ezen halmazokra a megfelelő TID részhalmazok (pl. $\{a\}$ és $\{b\}$) metszeteként meghatározza az előfordulások számát, amely rendre a következő lesz: $\{ab\}:1, \{ac\}:2, \{ad\}:3$. Mivel az $\{ab\}$ nem teljesíti a minimális előfordulások számára meghatározott feltételt, ezért ő nem gyakori, a másik kettő elemhalmaz viszont igen. Ezen két elemhalmazból kiindulva az algoritmus legenerálja az $\{acd\}$ jelöltet, melynek előfordulása analóg módon 1-nek adódik, tehát szintén nem gyakori. Ezáltal az algoritmus ennek a mélységi ágnak a végére ért, s folytatja működését a „b” elem vizsgálatával (de az „a” elemmel már nem veszi az unióját).

Összességében láthatjuk, hogy az ECLAT a jelöltállítás során nem vizsgálja, hogy a jelölt összes részhalmaza gyakori-e, s éppen ebből adódóan legalább annyi jelöltet állít, mint az Apriori algoritmus. A jelöltek gyakoriságának meghatározása a TID listáknak köszönhetően viszont sokkal gyorsabb az Apriori algoritmusnál, s ez az algoritmus futási idejében is megmutatkozik.

Az *FP-growth algoritmus* a gyakori elemhalmazok feltárását egy speciális adatszerkezeten, az FP-fán (Frequent Pattern Tree) hajtja végre. Az FP-fa a tranzakciókat tárolja rendkívül tömör formában a következő módon. Első lépésben az algoritmus megkeresi a gyakori elemeket, majd ezeket az előfordulásuk szerint csökkenő sorrendbe rendezi. Az algoritmus a tranzakciók elemeit szintén csökkenő gyakoriság szerint sorba rendezi, s törli belőlük a ritka elemeket. Ezt követően veszi a leggyakoribb elemet, majd azon tranzakciók alapján, amelyek tartalmazzák ezt az elemet létrehoz egy faszerkezetet. Ezt oly módon teszi meg, hogy a fa gyökere (null) alá felveszi a leggyakoribb elemet, majd az első tranzakció alapján növeszti a fát úgy, hogy csökkenő gyakorisági sorrendben egymás alá felveszi az elemeket. Minden csúcs egy elemnek feleltethető meg, őket ágak kötik össze. A csúcsok rendelkeznek egy

számlálóval is, ami az első tranzakció feldolgozása után mindegyiknél 1-es értéket tartalmaz. Ezután a következő szűrt tranzakció alapján folytatódik a fa növesztése, amikor is szintén a gyökértől indul ki, s ha a fa már tartalmazza az adott tranzakció egy részét (pontosabban az elejét, prefixét), akkor ennek a résznek nem növeszt új ágat, hanem megnöveli az érintett csúcsok számlálóját. Ha a tranzakció egy része nem létezik még a fában, akkor ahhoz a részhez az algoritmus a fában a megfelelő prefix után új ágat növeszt. Ez az eljárás rekurzív módon folytatódik az összes gyakori elem és a hozzájuk tartozó szűrt tranzakciók bejárásával. Az eredményképpen létrejött FP-fa a horizontális tárolás egy tömör formája. Mindemellett az algoritmus a vertikális tárolást is kódolja oly módon, hogy minden gyakori elemhez tárolja a gyakoriságát és egy láncolt listában hivatkozik azon tranzakciókra az FP-fában, amelyek az adott elemet tartalmazzák. Az FP-growth algoritmus ezen tömör és rendkívül hatékony tárolási formát használja a gyakori elemhalmazok kinyeréséhez. Az algoritmus pontos leírása a [17] irodalomban található meg.



4.3. ábra. Példa az FP-growth által alkalmazott FP-fára és a láncolt listára

A 4.3 ábra az $T = \{\{cdab\}, \{ac\}, \{eda\}, \{cbd\}, \{ad\}\}$ tranzakcióhalmazhoz tartozó FP-fát és a gyakori elemek láncolt listáját szemlélteti. A példában feltételezett minimális gyakorisági küszöb 30%. Láthatjuk, hogy mivel az „e” elem nem gyakori, ezért azt az algoritmus az FP-fa építésénél nem használja fel. Az ábrán látható szaggatott nyilak az elemekhez tartozó tranzakciók láncolt listáját szemléltetik.

4.3. Osztályozás

4.3.1. Az osztályozás fogalma

Az elemzendő adathalmaz egyedei gyakorta feloszthatók diszkrét csoportokra, úgynevezett osztályokra. Az *osztályozás* feladata, hogy a csoportokat leíró modelleket alkosson, és új, ismeretlen egyedek osztályba való tartozására becslést adjon. Az osztályozó algoritmusoknak számos felhasználási területe van, így például osztályozó algoritmusokat alkalmaznak a banki szférában a hitelfelvevők megbízhatósági faktorának becslésére, amikor a korábbi

hitelfelvevők tulajdonságai (pl. családos-e, havi jövedelme) és törlesztési szokásai alapján megbízhatósági csoportokat állítanak fel (pl. megbízhatósági mutatóval leírva 1-től 5-ig), majd a bank új ügyfeleinek esetében az ügyfelek tulajdonságai alapján következtetnek a hitelvisszafizetési megbízhatóságukra.

Az osztályozás tehát egy kétlépcsős folyamat, amely egy osztályozó modell létrehozásából, majd a modell használatából épül fel. Az osztályozó modell különféle formát ölthet, így például létrehozhatunk HA-AKKOR alakú szabályokat, a tényleges működést elrejtő neurális hálókat, vagy tetszőleges egyéb logikát alkalmazó modelleket is.

Az osztályozó modell felépítése során a vizsgált adathalmaz egyedeinek osztálybesorolása ismert. Ezen osztálybesorolást az úgynevezett osztálycímke attribútum határozza meg. Az osztálycímke attribútum lehet a vizsgált adathalmaz része, illetve ennek hiányában az adatelemző feladata ezen tulajdonság létrehozása.

A célunk tehát az, hogy a tulajdonságok felhasználásával az osztályokat minél pontosabban leíró modelleket hozzunk létre. A feladat megvalósítására számos osztályozó algoritmust publikáltak, azonban ezen algoritmusok működésükből adódóan lényeges eltéréseket mutatnak. Az alkalmazandó algoritmus kiválasztásakor a következő aspektusokat kell figyelembe venni:

- Az *előrejelzés pontossága*: Az egyik leglényegesebb kérdés, hogy a létrehozott modell az új egyedek osztályokba történő besorolására milyen pontos becslést nyújt. Az osztályozók pontosságának mérését a 4.3.2 fejezetben mutatjuk be részletesebben.

- Az *osztályozás sebessége*: Az osztályozás sebessége két szempontból is fontos momentum. Egyrészt fontos a modell kialakításának sebessége, másrészt pedig a kialakított modell felhasználásának sebessége, vagyis hogy a modell az alkalmazása során milyen gyorsan tud becslést adni az új minták besorolására vonatkozóan. Amennyiben olyan feladatot kell támogatni, ahol gyakori a modellalkotás, akkor a modellalkotás terén gyors eredményeket szolgáltató algoritmusokat kell előnyben részesíteni. Abban az esetben viszont, ha az új egyedek osztályba tartozásának becslése a mindennapi tevékenység részét képezi (pl. banki ügyfelek megbízhatóságának becslése) mindenképpen olyan algoritmus alkalmazása javallott, amely ezen a területen szolgáltat gyors eredményeket.

- A *modell értelmezhetősége*: Míg bizonyos osztályozó algoritmusok olyan eredményeket szolgáltatnak, amik a felhasználók számára is könnyen interpretálhatók (pl. szabályok formájában), addig más algoritmusok fekete dobozként működnek, a létrehozott modellek rejtve maradnak, s csupán a bemenet és a kimenet ismert. Éppen ebből adódóan az algoritmus kiválasztásakor figyelembe kell venni, hogy a leendő felhasználók a modellt csak használni, vagy értelmezni is szeretnék-e.

- Az *algoritmus robosztussága*: Az elemzendő adathalmaz gyakran tartalmaz zajos adatokat, illetve sok adat hiányozhat belőle. Természetesen ezen hiányosságokat az adatok előkészítése során amennyire csak lehet ki kell küszöbölni, azonban ezt nem minden esetben lehet teljességgel elvégezni. Amennyiben az előkészített adathalmaz sok üres cellát tartalmaz, illetve a rendelkezésre álló adatok zajjal terheltek, akkor mindenkép-

pen olyan osztályozó algoritmust kell választani, melynek működését ezen hiányosságok kevésbé befolyásolják.

- Az *algoritmus skálázhatósága*: Az elemzendő adathalmaz gyakorta óriási méretű, s rengeteg attribútumot tartalmaz. Ebben az esetben a kiválasztott algoritmusnak olyannak kell lennie, amely nagy adathalmaz feldolgozását is el tudja végezni, s működése nem szakad félbe esetlegesen memóriahiány, vagy egyéb ok miatt.

Mint láthatjuk, az osztályozó algoritmus kiválasztását számos tényező befolyásolja, s gyakorta nem is olyan egyszerű minden elvárásnak eleget tenni. Általánosan elfogadott gyakorlat, hogy egy-egy probléma megoldására több algoritmust is segítségül hívunk, s a kapott eredmények ismeretében vonjuk le a következtetéseket.

4.3.2. Az osztályozás pontossága

Az osztályozó modell a vizsgált adathalmaz egyedei alapján jön létre, bizonyítania azonban új, az elemzők számára nem ismert egyedek osztályozása során kell. Hogyan lehet mégis az új egyedek osztályozása előtt megbecsülni az osztályozó módszer pontosságát? Ebben a fejezetben erre a kérdésre adjuk meg a választ.

Az osztályozás pontosságát az ismert egyedek alapján kell megbecsülni. Azonban ha az osztályozó pontosságát azon az adathalmazon mérjük, amelyiken a modell kialakítását elvégeztük, akkor egy túlzóan optimális becslést kapunk, amely nem nyújt érdemi információt az új egyedek osztályozási pontosságára vonatkozóan. Éppen ebből adódóan a rendelkezésre álló adathalmazt két részre szokás felosztani. Az egyik halmaz a *tréning halmaz*, amelyet a modell kialakításához használunk fel, a másik halmaz pedig a *teszt halmaz*, amelyen az osztályozó pontosságát mérjük. Az arányokat tekintve célszerű a tréning halmazt nagyobbra választani, hogy az osztályozó modell kialakításában minél több, változatosabb egyed vehessen részt. A tréning és teszt halmazok felosztására a következő technikák terjedtek el:

- Az egyik legegyszerűbb tesztelési módszer a *partícionálás (visszatartó, vagy százalékos felosztás)* módszere, amikor a rendelkezésre álló mintát két elkülönülő részhalmazra osztjuk. A javasolt felosztási arány szerint általában $2/3$ rész a tréning halmaz, $1/3$ rész a teszt halmaz. A felhasználó a vizsgált minták számosságának ismeretében azonban ettől eltérő százalékos felbontást is választhat. A partícionálás módszere elsősorban nagy egyedszámú adathalmaz vizsgálatakor alkalmazható.

- Az előző módszer továbbfejlesztett verziója a *véletlen mintavételezés módszere*, amikor a partícionálást k-szor végezzük el véletlenszerűen egymás után. Ehhez kapcsolódóan a modell kialakítása és tesztelése is k-szor történik, s az osztályozó pontosságát a k db tesztelés pontosságának átlaga adja. Ez a módszer kisebb elemszámú mintahalmaz osztályozójának becslésére alkalmas, hiszen így a modell felépítése és tesztelése során a partícionáló módszerhez viszonyítva több elemet vehetünk figyelembe.

- A *kereszt-validálás* során a mintákat k db részhalmazra osztjuk. A modell felépítését és tesztelését k-szor hajtjuk végre oly módon, hogy minden esetben kiválasztunk egy (korábban még ki nem választott) részhalmazt, s azt tekintjük teszt halmaznak, a többi

$k-1$ darabot pedig együttesen tréning halmaznak. Az osztályozó pontosságát az k db pontosság átlaga adja. Tapasztalatok alapján a kereszt-validálás $k = 10$ esetben adja a legjobb becslést az osztályozó pontosságára vonatkozóan, s mivel ez az egyik legpontosabb becslési módszer, ezért széles körben használatos.

- A *rétegzett kereszt-validálás* az előző módszer továbbfejlesztett változata, amely a k db halmaz kialakításánál azt is figyelembe veszi, hogy az egyes halmazokban a vizsgált osztályok eloszlása hasonló legyen.

- A *leave-one-out* a kereszt-validálás speciális esete, amikor k értéke pontosan megegyezik a vizsgált minták számával, ezáltal mindig csak egy mintát hagyunk ki a tréning halmazból. A módszer előnye, hogy a kereszt-validálásnál pontosabb modellt kapunk, hiszen minden iterációban több elemet használunk a modell kialakításához, azonban nagy k esetén ez rendkívül időigényes.

- A *bootstrap* módszer lényege, hogy az n db mintát beszámozzuk 1-től n-ig, majd generálunk n darab 1 és n közé eső véletlenszámot oly módon, hogy az ismétlődéseket megengedjük. Azon minták, amelyek sorszámát legalább egyszer legeneráltuk, a tréning halmazba tartoznak, a többi minta a teszt halmazt alkotja. Annak a valószínűsége, hogy egy minta a tréning halmazba kerül megközelítőleg $63,2\%$, míg a teszt halmazba való kerülésnek körülbelül $36,8\%$. Ebből adódóan a tréning halmaz mérete körülbelül $63,2\%$-a az eredeti mintahalmaznak. Az osztályozó hibája a $hiba = 0,632 \times hiba_{teszt} + 0,368 \times hiba_{tréning}$ képlet alapján adódik. A módszert többször alkalmazva a végső becsült hiba az egyes hibák számtani átlagaként számítható ki.

Mint láthatjuk számos eljárás létezik a tréning és teszt halmazok kialakítására. Most már csupán az a kérdés, hogy hogyan lehet kiszámolni egy osztályozó modell pontosságát a teszt halmaz ismeretében? Az *osztályozó pontossága* legegyszerűbb módon a teszt halmazon helyesen osztályozott minták számának és a teszt halmaz elemszámának hányadosaként határozható meg:

$$pontosság = \frac{helyesen\ osztályozott\ minták\ száma}{összes\ minta\ száma} \tag{4.10}$$

A helyesen osztályozott fogalma alatt azt értjük, hogy az osztályozó a mintát abba az osztályba sorolta, amelyikbe az osztálycímke attribútuma alapján tartozik.

Gondolhatnánk azt is, hogy kész vagyunk és hátradőlünk, azonban ezt korántsem tesszük jól. Gondoljunk csak arra az esetre, hogy egy csalások detektálására alkalmazott osztályozó esetében adott például 1000 minta, melyből 3 a csalás osztályába tartozik. Ha a kialakított modell csupán 1 esetet sorol a csalás osztályába, amely viszont éppen nem oda tartozik, akkor mindemellett még 996 esetet jól osztályoz. Pontossága tehát $0,996$, vagyis $99,6\%$. Ez alapján az osztályozónk nagyon is jónak tűnik, holott egyetlen csalást sem becsült jól. Ebből adódóan érdemes pár újabb mérőszámot bevezetni az osztályozók értékelésére vonatkozóan.

Tekintsük a 4.1 táblázatban látható *keveredési mátrixot*, ahol 2 osztály esetén (*Pozitív, Negatív*) mutatjuk be, hogy az osztályozó milyen hatékonysággal működik. Az ábrán az *a*

szimbólum jelöli azon minták darabszámát, melyek a *Pozitív* osztályba tartoznak, s az osztályozó oda is sorolta be őket. A b szimbólum azon minták darabszámát jelöli, melyek ugyan a *Pozitív* osztályba tartoznak, azonban az osztályozó a *Negatív* osztályba sorolta őket. A c és d értékek értelmezése analóg módon adódik.

	becsült Pozitív	becsült Negatív
tényleges Pozitív	a	b
tényleges Negatív	c	d

4.1. táblázat. Keveredési mátrix 2 osztály esetén

Ezen keveredési mátrix alapján az osztályozó modell pontosságát a következő mérőszámokkal jellemezhetjük:

- *Helyesen pozitív arány*: A helyesen osztályozott pozitív minták aránya. Szokás ezt a mérőszámot az osztályozó érzékenységének is nevezni. Kiszámítása:

$$TP = \frac{a}{a+b} \tag{4.11}$$

- *Tévesen pozitív arány*. A tévesen osztályozott negatív minták aránya. Kiszámítása:

$$FP = \frac{c}{c+d} \tag{4.12}$$

- *Helyesen negatív arány*. A helyesen osztályozott negatív minták aránya. Kiszámítása:

$$TN = \frac{d}{c+d} \tag{4.13}$$

- *Tévesen negatív arány*. A tévesen osztályozott pozitív minták aránya. Kiszámítása:

$$FN = \frac{b}{a+b} \tag{4.14}$$

- *Megbízhatóság (precision)*: A pozitív osztályba sorolt mintákon belül a valóban pozitív minták aránya.

$$P = \frac{a}{a+c} \tag{4.15}$$

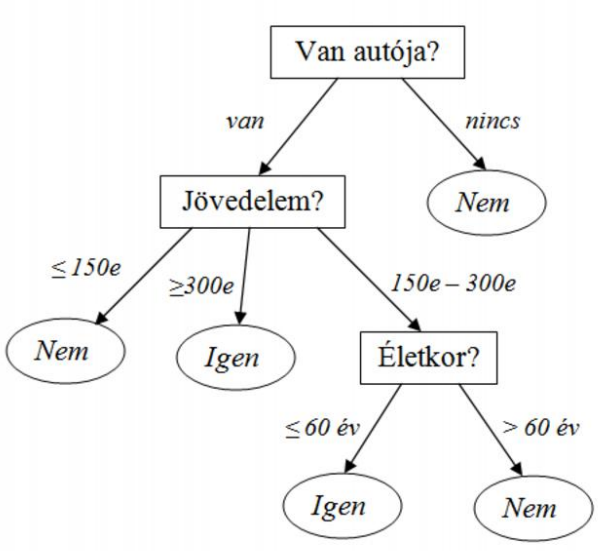
Az osztályozók helyességének becslésére egyéb mérőszámok is léteznek, azonban a keveredési mátrixot, amely könnyen általánosítható 2-nél több osztály esetére is, szinte minden adatbányász alkalmazás megadja. Érdemes időt szakítani áttekintésére, hiszen böngészésével árnyaltabb képet kaphatunk a kialakított osztályozó modellről. Így például a biztosítási példánál láthatjuk, hogy ha a csalásokat, mint elsődlegesen feltárandó célt tekintjük a pozitív osztálynak, akkor a megbízhatóság értéke 0%-nak adódik.

4.3.3. Gyakori osztályozó algoritmusok

A különféle adatbányász alkalmazások számos osztályozó algoritmus implementációját tartalmazzák. A következőkben a döntési fákon alapuló osztályozó algoritmusokat mutatjuk be részletesebben, mivel a felhasználó ezen algoritmusok működését tudja leginkább befolyásolni. Ezt követően a fejezet záró soraiban röviden felvillantunk néhány egyéb osztályozó algoritmust is.

A *döntési fákon alapuló osztályozó algoritmusok* a legkedveltebb módszerek közé tartoznak, mivel az eredményképpen létrejött modellek könnyen értelmezhetőek, ugyanis a modellek döntési fák, illetve szabályok formájában leírhatóak. A *döntési fák* olyan fa alakú gráfok, melyek köztes csúcsaiban az attribútumok értékeire megfogalmazott kérdések, az élek mentén pedig a lehetséges válaszok helyezkednek el. A fa levelei az osztálycímkéket tartalmazzák. Egy új eset osztályozása úgy történik, hogy elindulunk a fa gyökerétől, majd az egyes csomópontoknál megvizsgáljuk a vizsgált mintának azon attribútumát, melyre a kérdés vonatkozik, s az attribútum értékének megfelelő válasz irányába haladunk tovább. Elérve a döntési fa adott ágon lévő legutolsó csomópontját, vagyis a levelét, megkapjuk a legvalószínűbb osztálybesorolást.

Döntési fára a 4.4 ábrán láthatunk példát. A példában bemutatott osztályozó arra keresi a választ, hogy egy ügyfél vesz-e GPS-t. A lehetséges osztályok: „Igen” és „Nem”. Az ábrán az osztálycímkék ovális keretben láthatók, míg a döntésig vezető tesztkérdéseket téglalap keretezi.



4.4. ábra. Példa döntési fa

Az elkészült döntési fából könnyen generálhatók HA-AKKOR alakú szabályok oly módon, hogy a döntési fa egy ága mentén a kérdés-válasz párokat konjunkcióval fűzzük össze, a következtetést pedig a levél adja. A 4.4 ábrán látható döntési fa alapján 5 db szabály generálható, melyek közül az egyik a következő: Ha van autója és jövedelme több mint 300 ezer, akkor vásárol GPS-t.

A döntési fák generálása során az osztályozó algoritmusok különféle elv alapján kiválasztanak egy attribútumot, majd ezen attribútum értékei mentén szeparálják a mintákat. Ezen eljárást rekurzív módon végzik mindaddig, amíg a következő megállási feltételek egyike teljesül:

- Nincs olyan szétosztási lehetőség, mellyel javítani tudnánk az adott csomópont által kínált osztályozáson. Például, adott részhalmazban minden minta egy osztályba tartozik.
- Az algoritmus elért egy előre definiált maximális famélységet.
- Nincs már több olyan attribútum, mely mentén tovább oszthatnánk a mintákat.

A levelek létrehozásakor az algoritmusok azt az osztályt határozzák meg levélcímkeként, amelybe az adott levélen lévő tanító minták többsége tartozik. Az algoritmusok jellemzően nem a maximálisan felépíthető fát adják eredményül, hiszen ezek a fák túlságosan is illeszkednének a tréning halmaz adataihoz, tehát túl specifikusak lennének. A túltanítás elkerülése végett az eljárások bizonyos ágakat a működés során nem engednek létrehozni, illetve utólagosan levágják őket. Ezt hívjuk elő-, illetve utónyesésnek. Továbbá megfigyelhetjük azt is, hogy az adatbányász szoftvercsomagok általában nem csupán a becsült osztályt adják meg a leveleken, hanem az adatok adott osztályba való tartozásának valószínűségét is. Ezen információ birtokában az elemzők és szakértők még árnyaltabb képet kaphatnak az osztályozás bizonyosságáról.

A döntési fákon alapuló algoritmusok különféle módon választják ki, hogy a következő lépésben mely attribútum mentén szeparálják az elemeket. Az elv abban egységes mindegyik módszer esetében, hogy mindig azt az attribútumot kell választani, amely a leghatékonyabban szolgálja az osztályozás feladatát. Ennek megfelelően a fában minél közelebb van egy tesztkérdés a gyökérhez, annál nagyobb információtartalommal bír az osztályozásra vonatkozóan. Az *ID3 algoritmus*, s ennek továbbfejlesztett verziói (C4.5 és C5.0) az *információnyereség elve* alapján hozzák meg döntésüket. Az információnyereség a következőképpen határozható meg: adott S adathalmaz, $C_i (i = 1, 2, \ldots, m)$ osztályok. Jelölje s_i a C_i-ben lévő minták számát. Az S halmaz entrópiája, vagyis a rendezetlenségének mértéke:

$$I(S) = -\sum_{i=1}^{m} p_i log_2 (p_i), \tag{4.16}$$

ahol p_i a C_i halmazba való tartozás valószínűsége:

$$p_i = \frac{s_i}{\sum_{i=1}^{m} s_i} \tag{4.17}$$

Tekintsük az A attribútumot, amely értékkészletének vágása mentén v darab partíció ($S_1, S_2, \ldots, S_v$) jön létre. Legyenek N és P az osztályok, elemeik száma rendre n és p. Jelölje p_i az S_i-n belüli P-beli minták darabszámát, n_i pedig az S_i-n belüli N-beli minták darabszámát. Az A attribútum mentén történő vágás során a minták osztályba sorolásának várható információigénye:

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i) \tag{4.18}$$

Az A attribútum mentén történő vágás a következő információnyereséget eredményezi:

$$Nyereség(A) = I(p,n) - E(A) \tag{4.19}$$

Az algoritmus minden lépésben azt az attribútumot választja, amelynél legnagyobb a nyereség értéke.

Elemzői szempontból fontos még kiemelni, hogy míg az ID3 csak kategorikus adatokkal tud dolgozni, addig a C4.5 és C5.0 már folytonos változókat is kezel, s automatikusan határozza meg ezekre az attribútumokra a legjobb vágási feltételt. Az algoritmus működése során egy adott attribútumot nem tesztel újra, ha azt már korábban vágási feltételként kiválasztotta. Ezen algoritmusok mellett természetesen léteznek egyéb algoritmusok is, melyek szintén döntési fákon alapulnak. Így például az információnyereség elve helyett gyakran alkalmazott módszer a Gini-index alapján történő vágás, melyről részletesebben az [1] irodalomban olvashatunk.

A *k-legközelebbi szomszéd* osztályozási technika a mintákat egy n-dimenziós térben képzeli el, ahol n az osztályozás során figyelembe vett attribútumok darabszáma. Az algoritmus az eddig bemutatott módszerektől eltérően nem hoz létre modellt, amely leírná a vizsgált attribútumok szerepét, hanem csupán az új egyedek osztályozását végzi el. Az osztályozás alapja az n-dimenziós tér, melybe elhelyezve az osztályozandó mintát az algoritmus megkeresi a k db legközelebbi szomszédját, majd a vizsgált mintát abba az osztályba sorolja, amely osztály előfordulása leggyakoribb a k-legközelebbi szomszédok körében. A minták közelségének meghatározása folytonos attribútumok esetén általában az euklideszi távolság alapján történik, az egyéb attribútumok esetén alkalmazható távolságfüggvényeket pedig a 4.4.2 fejezetben mutatjuk be. A módszer hátránya, hogy nagyon érzékeny az adathibákra, viszont kis számításigényű, s megfelelő k bemeneti paraméter mellett viszonylag megbízható eredményt szolgáltat.

A *Bayes-osztályozás* egy statisztikai alapokon működő osztályozó algoritmus, amely a Bayes-elvet használja fel működése során. Az osztályozó előzetes modellt nem épít, csupán a tréning halmaz feltételes valószínűségei alapján ad javaslatot az új egyedek besorolására. Előnye a skálázhatósága, mivel az adatminták számával a futási idő csak lineárisan növekszik. Mindemellett az algoritmus a hiányzó adatokat is képes kezelni oly módon, hogy a valószínűségek számításakor a hiányzó attribútumértékeket tartalmazó adatmintákat nem veszi figyelembe. Az algoritmus működése során azonban egy naiv feltételezéssel él, miszerint a vizsgált attribútumok teljesen függetlenek egymástól. Ezen hiányosságot küszöbölik ki a *Bayes-féle hihetőségi hálókon alapuló osztályozó algoritmusok*, melyek működésük során figyelembe veszik attribútumok között létrejött (pl. felhasználó által megadott) függőségi kapcsolatokat is.

Az osztályozási problémák megoldására gyakran hívnak neurális hálókat is segítségül. A *neurális hálókat alkalmazó osztályozó algoritmusok* előnye, hogy robosztusak, meglehetősen nagy pontossággal dolgoznak és gyorsan adnak becslést új egyedek osztályozása esetén. Hátrányuk viszont, hogy a tanítási folyamat rendkívül időigényes, s az elkészült modell nem értelmezhető. A felhasználó új minta osztályozásakor csupán egy fekete dobozt lát, melybe a bemenetet az új minta képezi, a kimenet pedig maga az osztályozás eredménye.

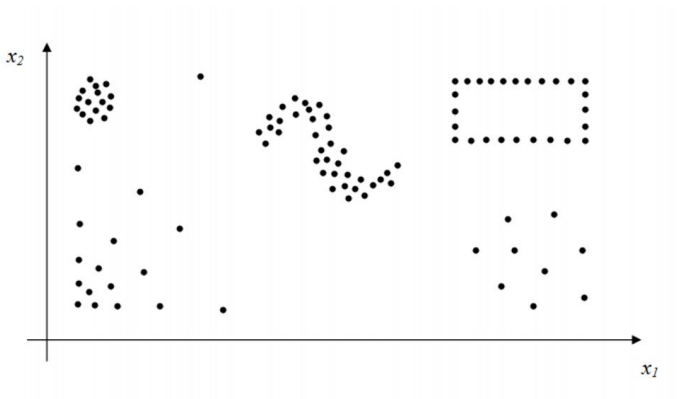
4.4. Csoportosítás

4.4.1. A csoportosítás fogalma, fajtái

A *csoportosítás (klaszterezés)* során a vizsgált halmaznak egy olyan felosztását keressük, amely az egymáshoz hasonló egyedeket azonos osztályba, az egymástól nagy mértékben különböző elemeket pedig különböző osztályokba sorolja. A csoportosítás karakterisztikus jellege az osztályozással történő összehasonlítása során domborodik ki. Az osztályozás során ugyanis a vizsgált adathalmaz egyes egyedeinek osztályokhoz történő tartozása már eleve ismert, ezzel szemben a csoportosítás esetében ezeket az osztályokat nem ismerjük, csupán keressük azokat.

A csoportosítás rendkívül széles körben elterjedt adatelemzési módszer, számos gyakorlati alkalmazása létezik. Így például csoportosító algoritmusokat használnak a cégek ügyfeleik elemzése során, amikor is az ügyfeleket csoportokba sorolják azon célból, hogy a csoportok számára eltérő, számukra speciálisan érdekes üzleti ajánlatokat közvetítsenek (direkt marketing). Az alkalmazói kör azonban nagyon széles, számos példát lehetne hozni az egészségügy, a pénzügy, a telekommunikáció és tetszőleges kutatási területről is. A csoportosításnak egy speciális alkalmazási területe, amikor a csoportok feltárása kapcsán azokat az elemeket keressük, amelyek egyik csoportba sem tartoznak, illetve esetleg önmagukban hoznak létre kis csoportokat. Az ilyen elemeket szokás *outliereknek* (vagy kiugró értékeknek) nevezni, melyek részben adathibákra, részben speciális esetekre, gyakran pedig csalásokra hívják fel a figyelmet. A csoportosítás ilyen jellegű alkalmazása rendkívül elterjedt a banki és biztosítási szférában, illetve a különféle biztonsági rendszerek is tartalmaznak ilyen jellegű beépített modulokat.

A csoportosítás szépségét, s egyben nehézségét is jelöli, hogy a keresett csoportok nagyon eltérőek lehetnek. Legegyszerűbb talán az egyedeket, mint adatpontokat a térben elképzelni, s ekkor könnyen rájöhetünk, hogy az egyes csoportok eltérő alakzatúak, sűrűek, s elemszámúak lehetnek. Ilyen eltérő csoportokra mutat példát a 4.5 ábra. Az ábrán láthatunk konvex, konkáv, tömör, lyukas, sűrű, ritka csoportokat, olyan csoportot, ahol az adatpontok sűrűsége a csoporton belül is változik, valamint outlier adatot is.



4.5. ábra. Különféle megjelenésű csoportok

A vizsgált adathalmaz egyedei azonban nem minden esetben sorolhatók be egymástól jól

elkülönülő, diszkrét csoportokba, mivel az egyes csoportok gyakran átfedhetik egymást, részben összemosódhatnak. A valóságot gyakorta sokkal jobban modellezzük abban az esetben, ha *fuzzy csoportosítást* hajtunk végre, minek eredményeképpen a vizsgált mintáknak az egyes csoportokhoz való tartozásának valószínűségét kapjuk meg. Tekintsünk egy konkrét példát! Tételezzük fel, hogy az $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N$ egyedeket szeretnénk csoportosítani, s eredményképpen 2 csoport (C_1 és C_2) jön létre. A kemény (nem fuzzy) csoportosító algoritmusok egy-egy elemet egy konkrét csoportba sorolnak be, így például eredményül azt kaphatjuk, hogy az $\mathbf{x}_1$ objektum a C_1 csoport része. Ezzel szemben a fuzzy csoportosítás eredménye ennél árnyaltabb képet fest egy elem csoportba tartozásáról, így például eredményül azt láthatjuk, hogy $\mathbf{x}_1$ $0,8\%$-os valószínűséggel tartozik a C_1 csoportba és $0,2\%$-os valószínűséggel a C_2 csoportba. Fuzzy csoportosítás esetén tehát a csoportba tartozás mértékét az úgynevezett *tagsági függvény* értéke határozza meg, melynek értéke $[0,1]$ intervallumban mozog, s egy elem esetén a csoporttagsági értékek összege pontosan 1. A fuzzy tagsági függvények tömör tárolása a $c \times N$-es fuzzy partíciós mátrixban valósul meg, ahol a mátrix tetszőleges μ_{ij} eleme a j. objektum i. csoportba való tartozásának mértékét definiálja, c pedig a csoportok száma.

Olyan univerzális algoritmus, amely tetszőleges esetben, tetszőleges csoportok feltárására alkalmas, nem létezik. Azonban vannak olyan általános jellemzők, amelyeket a csoportosító algoritmusoktól elvárunk. Az egyik legfontosabb tulajdonság, hogy a csoportosítandó egyedek sorrendje - vagyis az, hogy például a relációs adatbázisban milyen sorrendben szerepelnek a rekordok - ne befolyásolja a csoportosítás eredményét. Természetesen fontos, hogy a csoportosítást végző algoritmus sok tulajdonsággal jellemzett (nagy dimenzionalitású) és tetszőlegesen nagy elemszámú minta csoportosítását is viszonylag gyorsan el tudja végezni, és a zajos adatok ne befolyásolják nagy mértékben a működését. Miután a csoportosítandó objektumokat általában különféle adattípusú attribútumok írják le, ezért egy jó csoportosító algoritmusra jellemző, hogy képes a különböző típusú adatokat kezelni. Jó lenne, ha egy algoritmus tetszőleges alakú és sűrűségű csoportok felismerését lehetővé tenné, de ez a kritérium általában sajnos nem teljesül. Ezért egy-egy csoportosító algoritmus kiválasztása előtt fontos feltérképezni, hogy milyen típusú csoportok felismerésére alkalmas. Az algoritmusok bemenő paraméterei részben segítik az elemzők munkáját, részben meg is nehezítik azt. A paraméterek finomhangolásával részletesebb betekintés nyerhető a vizsgált adathalmaz összefüggéseibe, illetve a csoportosítás eredménye is javítható ezáltal. Azonban azt sem szabad elfelejteni, hogy téves paraméterbeállítás mellett a valóságtól nagyon elrugaszkodott eredmények is adódhatnak.

Felmerül a kérdés, hogy hogyan lehet értékelni a csoportosítás eredményét? Az adatvizualizáció, s ezen belül a 3. fejezetben ismertetett dimenziócsökkentési eljárások a csoportosítás folyamán kiemelt jelentőséggel bírnak. Ha az emberi szem számára is láthatóvá válnak a minták, ezek térbeli elhelyezkedése, illetve a csoportosítás eredménye (például színezéssel), akkor szubjektív vélemény formálható a csoportosítás eredményének tekintetében. A csoportosítás szubjektív értékelése mellett azonban természetesen léteznek az eredmény jóságát objektíven leíró mérőszámok is, melyeket a 4.4.4 fejezetben mutatjuk be.

Mint láthatjuk, a csoportosítás számos kihívást tartogat az elemző számára. Eltérő eredményeket kaphatunk ugyanazon algoritmus többszöri, különböző paraméter beállításokkal végzett futtatása során, és a különböző algoritmusok ugyanazon adathalmaz esetében is gyakran eltérő csoportosítási eredményt szolgáltatnak. Általánosan követendő elvként elmondha-

tó, hogy a csoportosítás elvégzéséhez érdemes több algoritmust is segítségül hívni, és ezen algoritmusok paraméterbeállítását és az eredményt részletesen áttekinteni.

4.4.2. Különbözőség és hasonlóság mérése

Az egyedek csoportosításának alapját azok hasonlóságának, illetve különbözőségének megállapítása adja. *Folytonos értékkészletű* attribútummal jellemzett egyedek esetén azok távolságát könnyen kiszámíthatjuk oly módon, hogy az n attribútummal jellemzett egyedeket egy n-dimenziós adattérben képzeljük el, s az így kapott n-dimenziós adatpontok távolságát határozzuk meg. A távolság kiszámításához leggyakrabban az euklideszi távolságnormát szokás segítségül hívni, de emellett számos egyéb távolságnormát is alkalmazhatunk (pl. Minkowski-távolságok egyéb formái, Mahalanobis-távolság). Gráf alapú csoportosító algoritmusok esetén az euklideszi távolságnorma helyett gyakran használatosak olyan hasonlóságmértékek, amelyek az egyedek gráfbeli kapcsolatán alapulnak (pl. közös szomszédok száma).

Kategorikus értékkészletű attribútumokkal jellemzett objektumok esetén az egyedek hasonlóságának kiszámításához a folytonos attribútumok távolságmértékei nem használhatók. *Bináris attribútumok* esetén a kontingenciatáblázatot hívhatjuk segítségül. Adott $\mathbf{x}_i$ és $\mathbf{x}_j$ ob-

		$\mathbf{x}_i$	
		1	0
$\mathbf{x}_j$	1	a	b
	0	c	d

4.2. táblázat. 2×2-es kontingenciatáblázat

jektumok, és jelölje 0 és 1 a bináris attribútumok által felvehető értékeket. Szimbolizálja a azon bináris attribútumok számát, amelyen $\mathbf{x}_i$ és $\mathbf{x}_j$ is 1-es értéket vesznek fel. A b, c, d értékek analóg értelmezhetők. Amennyiben a bináris értékek egyenrangúak, akkor az $\mathbf{x}_i$ és $\mathbf{x}_j$ egyedek hasonlósága a következő képlet alapján számítható:

$$s_{ij} = \frac{a+d}{a+b+c+d} \tag{4.20}$$

Aszimmetrikus esetben, amikor az egyik bináris érték a másikhoz képest kiemelt szereppel bír (betegség előfordulása), a kevésbé fontos értékek egyezőségének számosságát a számlálóban nem kell figyelembe venni. Ez alapján ha a kiemelt értéket 1-gyel jelöljük, akkor a hasonlóság a következőképpen számítható:

$$s_{ij} = \frac{a}{a+b+c+d} \tag{4.21}$$

Felsorolás típusú változók esetén a hasonlóság szintén különféle módon számolható. A legelterjedtebb megoldás, hogy a felsorolás típusú attribútumot annyi bináris attribútummal helyettesítjük, ahány értéket felvesz. Így a felsorolás típusú attribútumok hasonlóságának számítása visszavezethető bináris típusú változók hasonlóságának számítására. Fontos azonban kiemelni, hogy ezen attribútumokat aszimmetrikus bináris változókként kell kezelni, hogy a felbontásból adódó sok 0 érték egyezőségét ne vegyük figyelembe.

A *rendezett típusú* változók esete visszavezethető folytonos változók esetére oly módon, hogy a rendezett értékeket a $[0,1]$ intervallumba normalizáljuk a következő módon:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}, \tag{4.22}$$

ahol a z_{if} értékek a normalizált értékek, melyek az $\mathbf{x}_i$ objektum f rendezett típusú változójához rendelt $r_{if} \in 1,2,\ldots,M_f$ rangszámok értékeiből adódnak.

Mivel általános esetben az egyedeket különféle típusú attribútumok jellemzik, ezért ezen egyedek különbözőségének meghatározásához az előző különbözőség, illetve hasonlóság számítási módszereket kell kombináltan alkalmazni oly módon, hogy az attribútumértékek esetében a különbözőséget számítjuk ki, s ebből egy távolságmátrixot építünk fel. Ehhez a kategorikus attribútumok esetén kiszámolt hasonlóságértékekből (s_{ij}) a távolság értéke a $d_{ij} = 1 - s_{ij}$ képlet alapján adódik.

4.4.3. Gyakori csoportosító algoritmusok

Mielőtt rátérnénk a csoportosítás eredményének értékelésére, tekintsük át röviden a leggyakoribb algoritmusokat, ugyanis az itt ismertetett fogalmak nélkülönözhetetlenek az eredmények értékelése során.

A csoportosítási feladatok végrehajtására megszámlálhatatlanul sok algoritmus létezik, s mint már korábban említettük, nincs köztük egy sem, amely univerzális lenne. Ezen algoritmusok jelentős mértékben eltérnek egymástól azon tekintetben, hogy miként értelmezik a csoportokon belüli hasonlóságot és a csoportok különbözőségét, illetve abban is, hogy miként hozzák létre a csoportokat. A főbb algoritmuscsoportok és egy-egy nevezetesebb képviselőjük működésének bemutatása során jelölje $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ a csoportosítandó objektumok halmazát.

A *partícionáló algoritmusok* célja a vizsgált objektumhalmaz k csoportra történő felosztása, ahol k a felhasználó által meghatározott bemeneti paraméter. Ezen algoritmusok esetében a felosztás iteratív módon történik oly módon, hogy minden iterációban kiszámolnak egy hibamértéket, majd az elemek átpartícionálásával keresik azon felosztást, melynél ennek a hibának az értéke a legkisebb. A legnevesebb ilyen hibamérték a *teljes négyzetes hiba*, amely az objektumok és a csoportközéppontok eltérését összegzi a következőképpen:

$$E(C) = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} ||\mathbf{x}_i - \mathbf{c}_j||^2 \tag{4.23}$$

ahol C_j a j. csoportot, $\mathbf{c}_j$ pedig ezen csoport középpontját jelöli. Mint látni fogjuk a különböző algoritmusok a csoportközéppontokat eltérően értelmezik. A képletben szereplő $||\cdot||$ távolságnorma az euklideszi normát jelöli, s ezáltal ezen hiba csak folytonos attribútumok esetén értelmezhető.

A partícionáló algoritmusok legnevesebb képviselője a *k-átlag* algoritmus. Az algoritmus inicializálásként az N db objektumot véletlenszerűen k darab nem üres csoportba osztja. Ezt követően kiszámítja a csoportközéppontokat (centroidokat), majd minden objektumot azon csoporthoz rendel, amely centroidhoz legközelebb található. Miután ez megtörtént, az

algoritmus iteratív módon folytatja tovább működését az újabb csoportközéppontok meghatározásával és az elemek átpartícionálásával mindaddig, amíg valamely megállási feltétel nem teljesül (pl. egyik elemet sem kell átsorolni, vagy a hiba egy küszöbszám alá csökken, vagy az algoritmus elér egy maximális lépésszámot). A *k*-átlag algoritmus működése során a centroidok kiszámítása a csoportba sorolt objektumok attribútumértékeinek átlagaként adódik. Ebből fakad az algoritmus egyik hiányossága is, miszerint a centroidok csak folytonos attribútumértékek esetén számolhatók ki. A *k*-átlag algoritmus ezen hiányosságát küszöböli ki a *k-medoid* módszer, amely a centroidok kiszámítása helyett a középhez legközelebb elhelyezkedő objektumokat választja csoportközéppontoknak. Az ehhez szükséges hasonlósági mértékek már kategorikus értékkészletű objektumok esetén is könnyen értelmezhetőek (lásd [1]), ezáltal a *k*-medioid algoritmus olyan objektumok csoportosítására is alkalmas, melyek folytonos és kategorikus attribútumok által adottak. A *k*-átlag és *k*-medoid módszerek elsősorban jól elkülönülő, tömör csoportok feltárását teszik lehetővé, s nagyon érzékenyek az outlier adatokra. Ezen algoritmusok futását nagy mértékben befolyásolja a kiinduló csoportfelosztás. Egy-egy rossz felosztás esetén előfordulhat, hogy nem az optimális csoportosítást adják eredményül, s a minimalizáció során csupán egy lokális minimumot találnak meg. Ennek kiküszöbölése végett érdemes ezen algoritmusokat többször, különféle inicializálással futtatni, majd azt a megoldást elfogadni, ahol a teljes négyzetes hiba értéke a legkisebb.

A mellékletben található `katlag_demo1.avi` és `katlag_demo2.avi` fájlok a *k*-átlag algoritmus futását hivatottak szemléltetni. Mindkét fájl esetében lépésenként rögzítettük a csoportok kialakulását oly módon, hogy a piros karikák az egyes csoportközéppontokat, az eltérő színezések pedig a hozzájuk tartozó objektumokat szemléltetik. A két bemutató anyag összehasonlításakor láthatjuk, hogy a *k*-átlag algoritmus még ezen egyszerű példa esetében is rendkívül érzékeny a csoportközéppontok kezdeti meghatározására, s bár mindkét esetben optimális eredményt ad, azt eltérő lépésszámban éri el.

A *fuzzy c-átlag* csoportosítás a *k*-átlag algoritmus továbbfejlesztett verziója, melynek eredményeképpen fuzzy csoportok jönnek létre. A fuzzy *c*-átlag csoportosítás működése során a fuzzy *c*-átlag hibafüggvény minimalizálására törekszik, amely hibafüggvény a négyzetes hibaösszeg fuzzy szemlélettel történő kiterjesztése, s a következőképpen adható meg:

$$J_m(\mathbf{X},\mathbf{U},\mathbf{V}) = \sum_{i=1}^{c}\sum_{k=1}^{N} (\mu_{ik})^m \, ||\mathbf{x}_k - \mathbf{v}_i||^2 \tag{4.24}$$

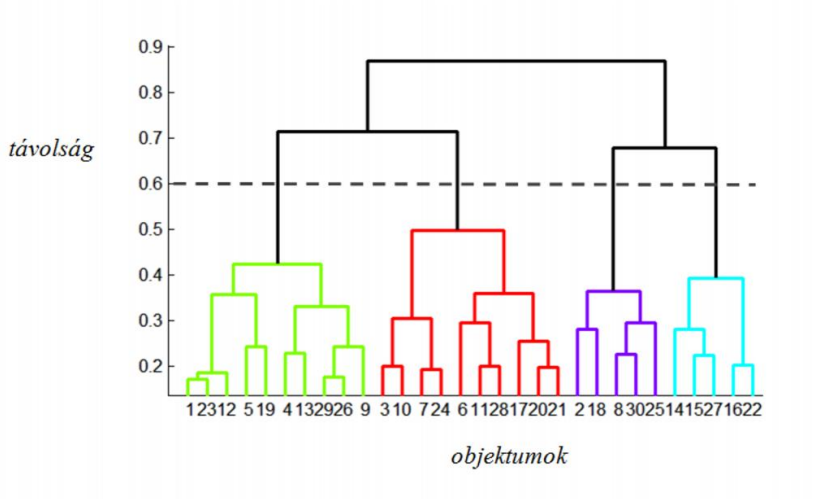
ahol $\mathbf{X}$ a csoportosítandó objektumok halmaza, N ezen halmaz számossága, c a csoportok száma, $\mathbf{U}$ a fuzzy partíciós mátrix, melynek egy eleme μ_{ik}, $\mathbf{v}_i$ az i. csoport középpontja, és m egy 1-nél nagyobb súlyozási kitevő, amely a fuzzy jelleget határozza meg. A fuzzy *c*-átlag algoritmus működése hasonló a *k*-átlag algoritmuséhoz, azonban a fuzzy *c*-átlag esetében az inicializálás során a partíciós mátrix elemeinek is fel kell venniük egy kezdőértéket, illetve az egyes iterációkban a csoportközéppontok mellett a tagsági értékek is újraszámításra kerülnek. Az algoritmus addig fut, amíg ezen tagsági értékek változása kisebb nem lesz egy előre definiált küszöbértéknél.

A mellékletben található `fuzzycatlag_demo.avi` állomány a fuzzy *c*-átlag algoritmus futását szemlélteti. A csoportközéppontokat jelen esetben piros csillagok jelölik, míg az egyes objektumoknak az egyik kiválasztott csoporthoz való tartozásának mértékét (tagsági értékek) szürkeárnyalatos színezéssel ábrázoltuk. A fuzzy *c*-átlag algoritmus részletes

ismertetése az [1] irodalomban található meg.

A *hierarchikus csoportosító módszerek* a partícionáló módszerekkel ellentétben az adathalmazt nem előre definiált számú csoportra osztják. Ezen algoritmusok a vizsgált adathalmaz több, egymásba ágyazott felosztását eredményezik oly módon, hogy az eredményül kapott felosztások esetén a csoportok száma eltérő. Mit is jelent ez pontosan? Tegyük fel, hogy N darab objektumot szeretnénk csoportosítani. Egy teljes hierarchikus csoportosítás esetén olyan eredményeket kapunk, ahol a csoportok száma rendre $1,2,\ldots,N$ vagy fordítva. Az *egyesítő hierarchikus csoportosító algoritmusok* első lépésben N db csoportot hoznak létre (vagyis minden objektum külön halmazba tartozik), majd lépésről lépésre a csoportok összevonásával eljutnak ahhoz az eredményhez, ahol minden elem 1 csoportba tartozik. A *felosztó hierarchikus algoritmusok* ellentétes irányban haladnak, vagyis 1 csoportból indulnak ki, majd a csoport(ok) felbontása által eljutnak ahhoz az esethez, amikor minden elem külön csoportba tartozik.

A hierarchikus algoritmusok esetén felmerül a kérdés, hogy mely felosztást fogadjuk el optimális csoportosítási eredményként. Az eredmények értékelésében az elemzők munkáját nagy mértékben segíti az eredményül kapott csoportosítások dendrogramon történő ábrázolása, melyre a 4.6 ábrán látunk példát. A *dendrogram* egy diagram, amely a hierarchikus algoritmus futásának eredményét szemlélteti oly módon, hogy az egyes csomópontok a csoportoknak feleltethetők meg, az élek hossza pedig az egyesített/szétbontott csoportok távolságát szemlélteti, melynek értelmezésében a dendrogram mellett megadott skála segít. A hierarchikus algoritmusok futását célszerű ott befejezni, ahol ezen él kiugróan hosszabb a többi élhez viszonyítva. A 4.6 ábrán látható példa esetében az egyesítő hierarchikus algoritmus futását célszerű akkor befejezni, ha az egyesítendő csoportok távolsága nagyobb, mint 0,6 (lásd szaggatott vonal). A vizuális kiértékelés mellett természetesen számos numerikus mérőszám is segítheti az elemzők munkáját az algoritmusok eredményének kiértékelésében.



4.6. ábra. Dendrogram

A csoportok egyesítését és felosztását a különféle hierarchikus algoritmusok eltérő módon hajtják végre. Egyesítő algoritmusok esetén minden iterációban a leghasonlóbb csoportok kerülnek összevonásra, azonban a hasonlóság különféleképpen értelmezhető. Így például

tekinthetjük leghasonlóbbnak azon csoportokat, melyek legközelebbi elemei között legkisebb a távolság (egyszerű kapcsolódás elve), de azokat is, melyek elempárjainak átlagos távolsága a legkisebb (átlagos kapcsolódás elve), illetve számos egyéb lehetőség is létezik. A felosztó módszerek működése ennél jóval komplikáltabb, hiszen egy halmaz felosztásakor sokkal több variációs lehetőség létezik.

A hierarchikus módszerek előnyeként azt szokás kiemelni, hogy sok képviselőjük képes tetszőleges típusú adatokkal dolgozni, hátrányuk viszont az, hogy a korábban végrehajtott egyesítési/szétbontási lépések már nem vonhatók vissza, illetve a felosztó módszerek számítási költsége igen magas. A hierarchikus csoportosító algoritmusok legjellegzetesebb képviselői a CURE [10], ROCK [11] és CHAMELEON [20] algoritmusok.

Mindezen algoritmusok mellett azonban számos egyéb elven működő csoportosítási módszer is létezik. Így például a *sűrűség alapú algoritmusok* az elemek sűrűsége alapján próbálják meghatározni a csoportokat, azonban ezen módszerek nehezen birkóznak meg azon csoportok feltárásával, melyeken belül az objektumok sűrűsége változik. A *gráf alapú módszerek* az elemek között értelmezett gráfokon (például minimális feszítőfa) hajtják végre a csoportosítást oly módon, hogy a gráfból törlik a nem releváns éleket, s az így adódó erdőben a fák reprezentálják az eredményül kapott csoportokat. A gráf alapú csoportosító algoritmusok nagy előnye, hogy tetszőleges alakú (akár konkáv) csoportok felismerését is lehetővé teszik. Mindezek mellett léteznek *modell alapú módszerek* is, melyek fő jellegzetessége, hogy olyan csoportokat keresnek, melyek illeszkednek valamilyen (pl. matematikai) modellre.

Látható tehát, hogy számos csoportosító algoritmus létezik. Általános elvként elmondható, hogy a csoportosítási feladat végrehajtásához elengedhetetlen feltétel a választott algoritmus működésének és paraméterezésének beható ismerete.

4.4.4. A csoportosítás eredményének értékelése

A csoportosítás eredményének vizuális kiértékelésében hatékony segítséget nyújthatnak a megfelelően kiválasztott dimenziócsökkentési módszerek. Ezen megjelenítési lehetőségeket azonban számos adatbányászati implementáció nem tartalmazza, így ekkor az eredmények értékelése során csupán numerikus értékekre támaszkodhatunk.

Milyen numerikus adatokkal értékelhetjük a csoportosítás eredményét? Számos csoportosító algoritmus működésének alapja valamilyen hibaérték minimalizálása. Ezen hibaértékek vizsgálata az eredmény értékelése mellett a keresett csoportok számának meghatározásában is az elemzők segítségére lehet a következő módon: ábrázoljuk a hiba értékét a csoportok számának függvényében, majd válasszuk azt a csoportszámot, ahol a hibafüggvényben egy törést (könyököt) látunk. Magasabb csoportszám esetén előfordulhat ugyan, hogy a hiba értéke tovább csökken, azonban ez nem feltétlenül az optimális besorolást jelzi (pl. nem az az optimális csoportfelosztás, hogy minden objektum külön csoportba tartozik). A hibaértékek értékelése azonban minden esetben csak akkor végezhető el szakszerűen, ha ismert a hiba matematikai jelentése is, illetve arról sem szabad elfelejtkezni, hogy a különféle algoritmusok különféle hibamértékkel dolgozhatnak, melyek egymással nem, vagy csak konverzió után hasonlíthatók össze.

Természetesen léteznek az algoritmusoktól független mérőszámok is, melyek a csoportosítás „jóságát” hivatottak jelezni. Ezek közül az egyik leggyakrabban használatos mérőszám

a *Dunn-index* [9], amely elsősorban kompakt és jól szeparált eredménycsoportok értékelésére alkalmas. A Dunn-index értéke a következőképpen határozható meg:

$$DI(c) = \min_{i \in c} \left\{ \min_{j \in c, j \neq i} \left\{ \frac{\delta(C_i, C_j)}{\max_{k \in c} \{\Delta(C_k)\}} \right\} \right\}, \tag{4.25}$$

ahol

$$\delta(C_i, C_j) = \min \left\{ d(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i \in C_i, \mathbf{x}_j \in C_j \right\}, \tag{4.26}$$

$$\Delta(C_k) = \max \left\{ d(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i, \mathbf{x}_j \in C_k \right\}, \tag{4.27}$$

ahol C az eredménycsoportokat jelöli, melynek számossága c, és d egy távolságfüggvény. Ezek alapján láthatjuk, a Dunn-index a csoportok távolsága és átmérője alapján értékeli az eredményt, s azt a csoportosítást tekinthetjük a leginkább érvényes csoportosításnak, ahol $DI(c)$ értéke a legnagyobb. Az index alkalmazásának hátránya, hogy c és N növekedésével kiszámítása nagyon költségessé válik. Ezen hátrány kiküszöbölése és egyéb speciális igények kielégítése céljából a Dunn-indexnek számos variációja létezik.

A kemény csoportosítási eredményeket értékelő indexek hátránya, hogy átfedő csoportokat nem megfelelően tudnak értékelni. Ezen hiányosságot küszöbölik ki a fuzzy tagsági mértékeket is figyelembe vevő indexek. Az egyik legegyszerűbb ilyen index a *partíciós együttható* (Partition Coefficient) [4], amely a csoportok átfedésének mértékét méri, s a következőképpen határozható meg:

$$PC(c) = \frac{1}{N} \sum_{i=1}^{c} \sum_{j=1}^{N} \left(\mu_{ij}\right)^2 \tag{4.28}$$

ahol μ_{ij} a j. objektum i. csoporthoz való tartozásának mértékét kifejező tagsági függvény értéke. Minél nagyobb a $PC(c)$ értéke (legfeljebb 1 lehet), annál jobb felosztást értékel. A partíciós együttható hiányossága, hogy kiszámítása az objektumokhoz ténylegesen nem kapcsolódik, illetve értéke a c csökkenésével monoton csökken.

Az *osztályozási entrópia* (Classification Entropy) [5] a csoportfelosztás fuzzy jellegét méri, s a következőképpen számolható ki:

$$CE(c) = -\frac{1}{N} \sum_{i=1}^{c} \sum_{j=1}^{N} \mu_{ij} \log\left(\mu_{ij}\right) \tag{4.29}$$

Az osztályozási entrópia értékelése során azon felosztás fogadható el jó felosztásnak, amely esetben az entrópia a legkisebb.

A Xie és Beni által javasolt *szeparációs mérték* (Separation measure, Xie-Beni index) [39] hasonló a Dunn-indexhez és értéke a következőképpen számítható:

$$XB(c) = \frac{\sum_{i=1}^{c} \sum_{j=1}^{N} \mu_{ij}^2 ||\mathbf{x}_j - \mathbf{v}_i||^2}{N \min_{i \neq j \in \{1,\ldots,c\}} ||\mathbf{v}_j - \mathbf{v}_i||^2} \tag{4.30}$$

A Xie-Beni index a csoportok kompaktságának és szeparációjának arányaként értelmezhető, s az $XB(c)$ értéke az optimális csoportszám esetén lesz a legkisebb.

A felsorolt indexen túl számos egyéb index is létezik, mellyel a csoportosítás eredményét értékelhetjük. A minél részletesebb értékeléshez javasoljuk a különféle mérőszámok és a vizuális lehetőségek együttes használatát.

4.5. Egyéb speciális adatelemzési feladatok

Az adatelemzési feladatok számos esetben öltenek sajátos jelleget. Ez a sajátos jelleg adódhat az elemzendő adatok speciális formájából, illetve az elemzendő témakör sajátosságából is. A következőkben három ilyen speciális adatelemzési területet tekintünk át vázlatosan, melyek sorra a következők: idősor adatok elemzése, a web bányászata és szövegbányászat. Mivel mindegyik témakör rendkívül szerteágazó, így teljes körű ismertetésükre nem, csupán egy rövid betekintés nyújtására vállalkozunk.

4.5.1. Idősor adatok elemzése

Idősor adatok alatt kronológiailag egymást követő adatok gyűjteményét értjük. A mindennapi életben számos olyan megfigyelés létezik, melyek időben egymást követő adatok sorozatával írhatók le, így például valamely tőzsdeindex mozgása, a napi hőmérsékleti adatok, vagy egy EKG regisztrátum. Ezen adatok tárolása és elemzése nagy mértékben különbözik a korábban ismertetett módszerektől, hiszen az adatok valós értékei mellett még egy fontos momentumot figyelembe kell venni, mégpedig azok sorrendjét.

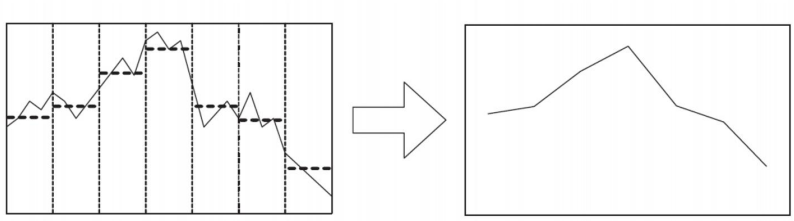
A dimenzionalitás csökkentése

Az idősor adatok kezelésének egyik nehézsége abból fakad, hogy nagy a dimenzionalitásuk, amely kifejezés idősor adatok esetén speciálisan az adatok számát jelöli. Éppen ebből fakadóan az egyik legfontosabb feladat a *dimenzionalitás csökkentése*, amely a legegyszerűbb módon történhet például véletlen, vagy valamilyen szabály szerinti *mintavételezéssel*. Nem nehéz azonban elképzelni, hogy ezen technika számos hátrányt rejt magában, így például alacsony mintavételezésnél torzulhat az eredeti adatsor alakja. A *PAA (Piecewise Aggregate Approximation)* módszere ezt fejleszti tovább oly módon, hogy az $Y = (y_1, y_2, \ldots, y_m)$ idősort felosztja n egyenlő hosszú szegmensre, ahol n a csökkentett dimenziójú adatsor dimenzióját jelöli, majd minden szegmensre kiszámítja az adatok átlagát, s ezen átlagot hozzárendeli az egyes időintervallumok közepéhez. Tehát a tömörített $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_n)$ idősor a következőképpen adódik:

$$\hat{y}_k = \frac{1}{e_k - s_k + 1} \sum_{i=s_k}^{e_k} y_i \tag{4.31}$$

ahol s_k és e_k az idősor k. szegmensének kezdő és végpontja. A módszert szemléletesen a 4.7 ábra mutatja be, ahol bal oldalon az eredeti idősor, jobb oldalon pedig a tömörített verziója látható. A módszer továbbfejlesztéseként Keogh javaslata alapján [21] az egyes szegmensek hosszát sem kell rögzíteni, hanem az adaptív módon változhat az idősor alakjának megfelelően.

A fenti két tömörítő eljárás során ténylegesen az idő jelentette azt a domaint, ami mentén a dimenziócsökkentést végrehajtottuk. A tömörítő algoritmusok másik családja szakítva ezen nézőponttal a vizsgált adatsort egy másik nézőpontba transzformálja. Ezen eljárások közül legelterjedtebbek a diszkrét Fourier transzformáció, amely az adatsort a frekvencia függvényében vizsgálja, és a diszkrét wavelet transzformáció. Ezen módszerekről bővebben az [1] irodalomban olvashatunk.



4.7. ábra. Idősor tömörítése PAA-val

Egy idősor elemzése

A legegyszerűbb idősorelemzés során csupán egyetlen idősor adatait kell elemeznünk. Általánosságban elmondhatjuk, hogy az idősorok a következő négy komponensből állnak össze: trend, szezonális ingadozás, ciklusos változás, véletlen ingadozás. Az idősorok elemzés során az elemző elsődleges feladata a vizsgált idősor ezen jellemzőinek feltárása.

A *trend* az idősor alakulásának fő irányát mutatja, vagyis azt, hogy alapvetően merre halad az idősor. A szezonális, ciklusos és véletlen változások ezen trend értékét korrigálják különféle módon. A *szezonális ingadozás* szabályos időszakonként visszatérő, állandó periódushosszúságú hullámzás, amely mindig azonos irányban téríti el az idősor értékét az alapirányzattól. Ilyen szezonális jellemző lehet például a csokoládéeladások mértékének növekedése a Mikulás- napot és húsvétot megelőző időszakokban. A *ciklusos változás* hosszabb időtávlatban megfigyelhető trend körüli ingadozást jelent. Mindezen három tényezőhöz adódik még a *véletlen ingadozás* értéke, amely jellemzően valamely váratlan eseményhez (pl. természeti csapás) köthető.

A négy tényező között *additív*, illetve *multiplikatív kapcsolat* állhat fenn. Amennyiben a szezonális ingadozás mértéke állandó nagyságú, akkor additív kapcsolatról, ha viszont a szezonális ingadozás mértéke az aktuális trendérték nagyságával arányosan változik, akkor multiplikatív kapcsolatról beszélünk. Matematikai formulával ezen kapcsolatok a következőképpen írhatók le (a 4.32 egyenlet az additív, a 4.33 pedig a multiplikatív modell):

$$y = t + s + c + v \tag{4.32}$$

$$y = t \cdot s \cdot c \cdot v \tag{4.33}$$

Az egy idősorra kiterjedő elemzések során általában ezen komponensek meghatározása, s az ez alapján adódó várható értékek meghatározása a cél. Mivel a véletlen ingadozások értéke nem tervezhető, ezért ezen komponenssel külön nem szokás foglalkozni.

Az *idősor trendjének meghatározása* történhet analitikus módon, illetve a mozgóátlagok módszerével is. Az *analitikus trendszámítás* során a regressziószámítást hívjuk segítségül, s az idősor fő irányultsága alapján lineáris, exponenciális, polinomiális, vagy egyéb trendfüggvénnyel próbáljuk leírni az idősor alakulását. A *mozgóátlagok* alkalmazása esetén az idősor elejétől az idősor végéig időpontonként egyesével lépkedve végigcsúsztatunk egy k méretű ablakot, kiszámoljuk az ablakba eső (k db) érték átlagát, majd ezen átlagot az ablak közepén található időponthoz rendeljük. Amennyiben k értéke páros szám, akkor mivel nem létezik az a mintavételezési időpont, ahova az adatot rendelhetnénk (pl. $k = 4$, és $i = 1,2,3,4$

időpontok esetén nincs $i = 2,5$ időpont) oly módon járunk el, hogy kiszámoljuk egymást követő 2 ablak átlagát, majd ezen átlagok átlagát rendeljük a 2 ablak középső időpontjához (pl. $k = 4$ esetén kiszámítjuk $i = 1,2,3,4$ értékek és $i = 2,3,4,5$ értékek átlagát, majd ezen átlagok átlagát az $i = 3$ időponthoz rendeljük). Értelemszerűen adódik, hogy az idősor elejéhez és végéhez nem tudunk így értékeket kiszámolni, itt legfeljebb a valós adatokat helyettesíthetjük vissza. A mozgóátlagok módszerével elérhetjük, hogy megfelelő k választása esetén kisimítjuk az idősorunkat, amely így már jobban sejteti a valós trendet, s akár a továbbiakban analitikus módon is modellezhetjük. A *kronológikus mozgóátlag* a mozgóátlag egy speciális esete, amely esetben az első és utolsó elemeket csak fele súllyal vesszük figyelembe az átlagok számítása során:

$$\bar{y}_{kron} = \frac{(y_1 + y_k)/2 + \sum_{i=2}^{k-1} y_i}{k-1} \tag{4.34}$$

Amennyiben a mérési időpontok nem egyenlő távolságra helyezkednek el egymástól, használhatjuk a súlyozott kronológikus átlagot is, ahol a súlyok a távolságokból származnak. Fontos még kiemelni, hogy ha van szezonalitás az idősorban, akkor a perióduson belüli időszakok számát, vagy annak többszörösét kell választani k értékének, hogy a szezonális hatások egyformán érvényesüljenek, s az idősor kisimuljon.

A *szezonális hatás kiszámítása* függ attól, hogy additív, vagy multiplikatív jellegű idősorról van-e szó. Amennyiben a komponensek között additív kapcsolatról beszélhetünk, akkor a nyers szezonális hatás (mivel a szezonális hatás a periódusok azonos időpontjaiban egyenlő mértékű) a periódus j. pontjában a következőképpen számítható ki:

$$s_j = \frac{\sum_{i=1}^{n/p} y_{ij} - t_{ij}}{n/p}, \tag{4.35}$$

ahol n az időpontok száma, p a periódus hossza, y_{ij} az i. periódus j. pontja, t_{ij} pedig a hozzá tartozó trend értéke. A képletben szereplő átlagolás jelentősége a váratlan hatások befolyásának csökkentésében van. Multiplikatív modell esetén s_j értéke a következő:

$$s_j = \frac{\sum_{i=1}^{n/p} y_{ij}/t_{ij}}{n/p}, \tag{4.36}$$

A *ciklikus mozgások kiszámításához* feltételezzük, hogy a vizsgált idősort mentesítettük a szezonális adatoktól. A vizsgált idősor ciklikus komponensét oly módon határozzuk meg, hogy mozgóátlagokat számolunk, majd a mozgóátlagokra trendet illesztünk. Ezen trendfüggvény és a mozgóátlagok különbsége adja a ciklikus változások értékét, hiszen ezen különbség mutatja meg, hogy a tényleges, szezonmentesített (és feltételezhetően véletlen mozgást nem tartalmazó) valós adat mennyire tér el a várható trendtől.

Az idősorok elemzése azonban sokszor nem korlátozódik egyetlen idősor analízisére. A következőkben több idősor összehasonlítására térünk át.

Idősorok összehasonlítása

Az idősor adatok összehasonlítása számos adatelemzés tárgya lehet. Így például idősorokat hasonlítunk össze, amikor arra vagyunk kíváncsiak, hogy vajon 2 tőzsdeindex mozgása mennyire hasonlít egymásra, illetve akkor is amikor bizonyos betegségek esetén az EKG diagramokon keressük a hasonlóságokat. Az idősorok összehasonlításának két fő módja van, méghozzá a *teljes idősorok hasonlítása*, amikor az idősorokat teljes hosszukban tekintjük, és ez alapján határozzuk meg a távolságukat, illetve a *részszekvencia keresés*, amikor egy hosszabb idősorban keressük azon szekvenciákat, melyek hasonlítanak egy előre definiált rövidebb idősorhoz.

Amennyiben *teljes idősorokat hasonlítunk össze*, akkor mindig 2 idősor összehasonlítására kell gondolnunk, s arra keressük a választ, hogy ezek milyen mértékben hasonlóak, illetve másképp fogalmazva, mennyire térnek el egymástól. Első megközelítésre azt gondolhatnánk, hogy elegendő kiszámolni a két idősor időpontokkénti euklideszi távolságát, s már készen is vagyunk. Ezen módszer azonban számos hiányosságot nem képes kezelni (pl. zajszűrés, elcsúszás), és nem alkalmazható olyan idősorok esetében sem, amelyek nem azonos hosszúak. Márpedig nem egyenlő hosszú idősorok összehasonlítására gyakran van szükség, gondoljunk csak a beszédfelismerésre, ahol ugyanazt a mondatot gyorsabb és lassabb verzióban is fel kell ismerni. Ezen hiányosságot küszöböli ki a legnépszerűbb idősor hasonlósági mérték, a *dinamikus idővetemítés (Dynamic Time Wrapping)*. A dinamikus idővetemítés a Q és P idősoroknak nem csak az azonos időpillanatban keletkezett adatait hasonlítja össze, hanem létrehoz egy $n \times m$-es mátrixot, ahol n a Q, m pedig a P idősor hossza, s a mátrix d_{ij} eleme a q_i és p_j adatpontok euklideszi távolságának négyzetét tárolja. A dinamikus idővetemítés $W = w_1, \ldots, w_k$ útja a mátrixelemek egy rendezett listája, amely mentén a legkisebb költséggel juthatunk el az d_{11} elemtől az d_{nm} elemig. Az idővetemítés útjára vonatkozóan teljesülnie kell a következő korlátozásoknak:

- Keretes feltétel: első eleme a d_{11}, utolsó eleme pedig a d_{nm}.
- Folytonossági feltétel: csak szomszédos cellákba lehet lépni, tehát egy $w_{k-1} = (a', b')$ és őt követő $w_k = (a, b)$ indexű elem esetén $a - a' \leq 1$ és $b - b' \leq 1$.
- Monotonitási feltétel: mindig a végső cella felé kell közelíteni, tehát $a - a' \geq 0$ és $b - b' \geq 0$

Számos út létezik a mátrixban, amely a fenti feltételeknek eleget tesz, azonban az elemzés szempontjából csupán az az út érdekes, melynek a költsége a legkisebb. Jelen esetben az út költsége a távolságok összegeként értelmezendő.

Miután az idősorok növekedésével a bejárható utak száma exponenciálisan nő, ezért célszerű csökkenteni a számítási költségeket. Számos módszer létezik a költségek csökkentésére vonatkozóan, így például alkalmazhatunk a mátrixban értelmezett térbeli bejárási korlátozásokat, illetve ezen célt szolgálja a *kumulált távolságok* tárolása is. Jelölje $\gamma(i, j)$ a kumulált távolságot, amely az aktuális cella és a szomszédos cellák kumulált távolságai minimumának összegeként adódik, vagyis:

$$\gamma(i, j) = d(q_i, p_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} \tag{4.37}$$

Az ilyen elven működő dinamikus programok működésük során tárolják a kumulatív távolságokat is, ezáltal az egyes résztávolságok újraszámítása már nem szükséges, vagyis a számítási költség csökken.

Részszekvencia keresése esetén adott egy Q szekvencia (kérdés), melynek hossza n, P a vizsgált idősor, melynek hossza m ($m \geq n$), és ε a távolság hibája. A feladat annak a kérdésnek a megválaszolása, hogy a Q részszekvencia előfordul-e a P sorozatban ε hibatűrésen belül. Ha $\varepsilon = 0$, akkor a keresett sorozatnak pontosan elő kell fordulnia a P szekvenciában. A probléma legtriviálisabb megoldásaként a távolságot euklideszi távolságként értelmezzük, s a P idősort végigvizsgáljuk az 1. elemétől az $m-n+1$ eleméig n hosszúságú részszekvenciákként, s minden esetben kiszámítjuk a két sorozat távolságát. Ha ez kisebb, mint ε, akkor azt találatnak minősítjük. A módszernek számos változata és gyorsítása létezik.

Az idősorok elemzése a fentieken túl számos egyéb érdekes kérdést is tartogathat. Így például adatbányászati eszközökkel csoportosíthatjuk az idősorainkat, illetve osztályozási feladatokat hajthatunk végre rajtuk, melynek eredményeit asszociatív szabályokban összegezhetjük. A feladatok speciális jellege miatt jelen jegyzet ezen fejezetekkel nem foglalkozik.

4.5.2. A web bányászata

A web óriási adatforrás az adatelemzés szempontjából. A web dinamikusan növekvő tartalma mellett ezen tartalmak kapcsolódása, s a webhez csatlakozó felhasználók tevékenységei szolgáltatják az elemzések tárgyát képező adatokat. A web bányászata a rendelkezésre álló adatok alapján tehát a következő részterületekre osztható: a *web tartalmának bányászata*, a *web struktúrájának bányászata*, és a *web használatának bányászata*.

A web bányászata azonban több okból fakadóan rendkívül összetett és sok kihívást tartogató feladat. A rendelkezésre álló adatok mennyisége mindamellett, hogy óriási, rendkívül gyorsan növekszik és változik is. Az elemzések végrehajtásakor nem feledkezhetünk meg arról a tényről sem, hogy a weben található adatok esetenként tévesek, nem valós információt takarnak. Az elemzéseket tovább nehezíti az a tény is, hogy a web nem tekinthető strukturált adatforrásnak, egy-egy weblap felépítése, a fejlesztéséhez használt eszközök, s ezáltal a megvalósult tartalom is jelentősen eltéréseket mutathat.

A *web tartalmi bányászatának* célja a web tényleges tartalmából, illetve az azokat leíró elemekből történő hasznos információ kinyerése. Eredményeképpen olyan szolgáltatások valósulnak meg, mint például a tartalom alapú keresés, illetve a különféle online vásárlói rendszereket összefogó árösszehasonlító szolgáltatások. A web tartalmának bányászata szoros kapcsolatban van az adatbányászattal, hiszen a webbányászat során számos adatbányászati technika alkalmazható (pl. weboldalak osztályozása, csoportosítása), azonban míg az általános adatbányászati algoritmusok strukturált adatokon dolgoznak, addig a web többnyire strukturálatlan, illetve félig-strukturált adatokat tartalmaz. Mindezek mellett bár a webtartalom bányászata szoros kapcsolatot mutat a szövegbányászattal is, mégis túlmutat azon a félig-strukturált jellegéből adódóan. Mivel a web tartalmának formája jelentős eltéréseket mutathat oldalanként (pl. telefonszám esetén: $+36301234567$, vagy $(30)123-4567$), ezért a módszerek elsődleges célja a tartalmak feltárását leíró szabályok létrehozása - amely történ-

het automatikusan és szakember által vezérelt módon is -, majd ezen szabályok alkalmazása új weboldalakból történő információkinyerés esetén.

A web azonban több információt tartalmaz, mint a rajta található tartalom. A weblapokon elhelyezett hivatkozásokkal a weblapok egy bonyolult rendszere alakul ki, amely alapján árnyaltabb képet kaphatunk az egy-egy weblapon elhelyezett tartalom fontosságáról. A *web struktúrájának bányászata* tehát a web kapcsolati rendszerében lévő hasznos információk feltárását célozza meg.

A weblink bányászat egyik tipikus alkalmazási területe a releváns, autentikus weboldalak keresése. Cél az olyan weboldalak megtalálása, amelyek nem csak fontosak, de jó minőségűek is, vagyis mérvadóak az adott tárgyban. A webes keresés általában rendkívül sok találatot eredményez, s a találatként megjelölt weboldalak manuális áttekintése rendkívül időigényes, gyakorlatilag sokszor kivitelezhetetlen. Éppen ezért fontos, hogy az eredménylistát szűkítve, fontossági szempontból sorba rendezve tárjuk a felhasználó elé. Ebben nyújt segítséget a weblink analízis. A weblink bányászat alapgondolata, hogy az oldalak rangja (rangossága) a rá hivatkozó oldalak rangjától függ. A kapcsolatok ugyanis nagy mennyiségű rejtett humán magyarázatokat tartalmazhatnak, illetve a kapcsolat által mutatott oldal olyan oldalnak tekinthető, amit az oldal szerzője jóváhagy. Természetesen gondolnunk kell arra is, hogy ezen kapcsolatok egy része reklám és marketing céllal lett az oldalra elhelyezve, illetve számolnunk kell azzal a ténnyel is, hogy egy cég például sohasem fog olyan linket elhelyezni, amely egy konkurens cég weblapjára mutatna.

A weboldalak rangsorolásának két fő megközelítése létezik. Az első megközelítés alapján az oldalakat keresési kulcsszavaktól független módon rangsoroljuk, míg a második megközelítés szerint a rangsorolás során figyelembe vesszük a keresőszavakat is. A második megközelítés előnye, hogy keresésfüggő eredményt ad, tehát más-más keresőszavak esetén nemcsak a találati lista lesz más, hanem azon belül a weblapok sorrendje is dinamikusan változik. Ezzel szemben az első megközelítés esetén a rangsorolást keresésektől függetlenül csak egyszer kell elvégezni, viszont egy-egy weblap rangja statikus, nem függ a keresőkifejezéstől. Az első módszert valósítja meg a Google által bejegyzett és védett PageRank algoritmus, míg a második módszer elterjedt algoritmusa a HITS algoritmus.

A *PageRank algoritmus N* darab oldal rangsorolását végzi el az „azok az oldalak fontosak, amelyekre fontos oldalak hivatkoznak” elv alapján. Az algoritmus futási eredményeképpen minden oldal rendelkezik egy PageRank mutatóval, amely az oldal fontosságát jelöli. Ez a PageRank gyakorlatilag annak a valószínűsége, hogy egy sztochasztikus szörföző az adott weblapra ér. Sztochasztikus szörfözésen azt értjük, hogy egy szörföző oly módon lépked a weblapok között, hogy kiindul egy tetszőleges weboldalról, s az ott elhelyezett linkek közül véletlenszerűen, egyenletes eloszlás szerint választva továbbhalad, majd minden további meglátogatott weboldalról ugyanígy halad tovább. Gondolva a zsákutca weboldalak problémájára azt is feltételezzük, hogy ez a véletlen szörföző bizonyos időnként elunja magát, s egy tetszőlegesen választott (tehát nem feltétlenül link alapján elérhető) weboldalról folytatja a szörfözését tovább. A gyakorlatban a PageRank algoritmus nem ilyen véletlen szörfözés révén számolja ki az egyes oldalak PageRank-jét, hanem egy szavazási módszert alkalmaz, ahol minden oldal szavazattal jutalmazza az általa hivatkozott weboldalakat, és szavazatokat kap a rá hivatkozó weboldalaktól. Kezdetben minden oldal azonos szavazattal rendelkezik, s ezt a szavazatot osztja szét egyenletesen az általa hivatkozott oldalak között. Ezt követő-

en a weblapok minden iterációban a rendelkezésre álló szavazatok egy bizonyos százalékát egyenlően szétosztják a hivatkozott oldalak között, a maradékot pedig „befizetik” a közösbe. A következő iterációban szétosztandó szavazatok a hivatkozó weblapok szavazataiból és a közösből egyenlő mértékben részesült szavazatokból állnak elő. Így érvényesül az az elv, hogy azon weblapok, amik fontosak egyre több szavazatot kapnak, s ezáltal egyre több szavazatot tudnak szétosztani, s az általuk hivatkozott weblapokat is fontossá teszik. Az iteráció egy megállási feltétel (pl. a PageRank-ek sorrendje nem, vagy csak kis mértékben változik) teljesüléséig folytatódik tovább.

A *HITS algoritmus* (Hyperlink-Induced Topic Search) a PageRank algoritmussal ellentétben már nem témafüggetlen, hanem a keresési kifejezésektől függő rangsorolást valósít meg. Az algoritmus alapgondolata, hogy a fontos, releváns oldalakat tekintély és gyűjtő oldalakra osztja. Azon weboldalakat nevezzük *authority (tekintély) oldalaknak*, amelyekre sok hivatkozás érkezik, s feltételezzük, hogy ezen oldalak a tekintélyüket a rajtuk elhelyezett információ releváns jellegével vívták ki. Egy-egy keresés során az ilyen authority weboldalak megtalálása a cél. Ebben a keresésben azonban nagy mértékben segíthetnek a *hub (gyűjtő) oldalak* is, melyek egy adott témakörben gyűjtik össze a az authority oldalak listáját. Az algoritmus egy tetszőleges keresőmotor találati listájából indul ki oly módon, hogy veszi ennek az első t előfordulását, és ez lesz a kiindulási bázis. Feltételezve, hogy ezen oldalak között vannak authority és hub oldalak is, a bázist tovább bővíti azzal, hogy hozzáveszi az ezen oldalakra hivatkozó oldalakat és az ezen oldalakról hivatkozott oldalakat is. Mivel a rendkívül népszerű, illetve a sok linket tartalmazó weboldalak számos új weboldal bázisba történő bevonását eredményezhetik, ezért egy weboldal által generált felvehető lapok számát maximalizálják. Ezt követően az algoritmus authority és hub súlyértékeket rendel minden bázisbeli oldalhoz, hogy iteratív módon meghatározza azok releváns jellegét. A hub érték az általa hivatkozott lapok authority súlyának összegeként, az authority érték pedig a rá hivatkozó lapok hubsúlyainak összegeként adódik. Az algoritmus futása során viszonylag kevés iteráció után kialakulnak a weblapokra jellemző authority és hub súlyok értékei, melyek az eredménylista rangsorolásának alapját adják.

A *webhasználat bányászatának* alapját a webszerverek által automatikusan tárolt weblog fájlok adatai biztosítják. Ezek számos adatot tartalmaznak, így például azt, hogy az egyes felhasználók mikor, milyen IP címről érkeztek, milyen oldalakat látogattak meg, mennyi időt töltöttek az egyes oldalakon. Ezeknek az adatoknak az elemzése segítségünkre lehet abban, hogy megfigyeljük, megértsük a felhasználói viselkedéseket, s ezáltal javítsuk, személyre szabjuk a weblapok szolgáltatásait. Az adatok elemzésében számos adatbányászati módszer alkalmazható (pl. szekvenciaanalízis, osztályozás), illetve segítségül hívhatjuk az adattárházak által nyújtott eszközöket is, amelyeket az 5. fejezetben ismertetünk.

4.5.3. Szövegbányászat

Napjainkban is igaz az a megállapítás, miszerint az adatoknak csak kis hányadát rögzítik strukturált formában, jelentős része strukturálatlan, szabad szöveges formában kerül tárolásra. Gondoljunk csak a könyvekre, a különféle gazdasági, kutatási, s egyéb jelentésekre, az orvosi dekurzusokra, zárójelentésekre, az e-mailekre, illetve a web szöveges tartalmi részeire, amelyek mind-mind szöveges formában tárolják az információt.

Felmerül a kérdés, hogy ezen szövegek feldolgozhatóak-e informatikai eszközökkel, ha igen, akkor milyen módszerekkel, s milyen jellegű tudás nyerhető ki belőlük. Az első kérdésre szerencsére igen a válasz, a másodikra pedig az, hogy ezen szövegek feldolgozásához a *szövegbányászat* kelléktárát kell segítségül hívnunk. A szövegbányászat hasonló célokat fogalmaz meg, mint az adatbányászat (így például osztályozási, csoportosítási feladatok elvégzése), de azon túlmutatva speciális feladatokat is megvalósít (pl. témafigyelés, kivonatolás). A legfontosabb eltérés azonban a feldolgozandó adatok megjelenési formájában mutatkozik meg. Míg az adatbányászati algoritmusok a strukturált formában tárolt adatok elemzését valósítják meg, addig a szövegbányászat célja a strukturálatlan szövegek, vagyis a dokumentumok feldolgozása, az azokból történő hasznos információ kinyerése. Miután jelen jegyzet elsődleges célja a strukturált formában tárolt adatok elemzési módszereinek bemutatása, ezért a szövegbányászatról, mint a strukturálatlan adatok elemzésének eszközéről csak figyelemfelkeltés szintjén kívánunk megemlékezni. A témában részletesen elmélyedni kívánó Olvasó figyelmébe a [35] irodalmat ajánljuk, amely a magyar nyelv sajátosságait is figyelembe véve mutatja be részletesen a szövegbányászat témakörét.

Az elemzendő adatok megjelenési formájából adódóan a szövegbányászat folyamatának első fontos lépése a dokumentumok előkészítése, vagyis olyan jellegű feldolgozása, amely által a természetes nyelvi szövegek mintegy valamilyen modellé konvertálva a matematikai módszerekkel dolgozó algoritmusok által is feldolgozhatóvá válnak. Erre a célra különféle *dokumentumreprezentációs modellek* léteznek, melyek közül legelterjedtebb az algebrai alapú vektortérmodell. A *vektortérmodell* alapja azon intuíció, hogy azok a dokumentumok hasonlítanak a leginkább egymásra, melyek szókészlete, a bennük előforduló szavak gyakorisága a leginkább egyező. A vektortér modellben minden egyes dokumentumnak egy vektor feleltethető meg, amely vektor a dokumentumban előforduló szavakra vonatkozóan ad információt. Adott tehát egy $D = \{d_1, \ldots, d_m\}$ dokumentumhalmaz, s egy n darab szót tartalmazó szótár. A dokumentumok adott szótár szerinti tömör reprezentációja az $m \times n$-es *szó-dokumentum mátrix* által valósul meg, ahol a mátrix a_{ij} eleme a szótár j. szavának d_i dokumentumban történő előfordulásáról nyújt információt. A mátrix a_{ij} elemének kiszámítása különféle módon történhet. Legegyszerűbb esetben $a_{ij} = 0$, ha a j. szó az i. dokumentumnak nem eleme, ellenkező esetben 1 (bináris súlyozás). Ez a reprezentáció természetesen szegényes, hiszen a szavak előfordulásának frekvenciája sok információt hordoz. Ebből adódóan az a_{ij} értékek kiszámíthatók a szavak előfordulásának gyakorisága alapján is. A modell tovább fejleszthető oly módon, hogy nem csak azt vesszük figyelembe, hogy az egyes szavak milyen gyakorisággal fordulnak elő egy-egy dokumentumban, hanem azt is, hogy előfordulnak-e más dokumentumokban. Hiszen az a szó, ami minden dokumentumban gyakori, valószínűleg kevésbé informatív számunkra, mint az a szó, ami az egyik dokumentumban gyakori, a többi dokumentumban viszont nem az, és akár elő sem fordul. Az ilyen elveken nyugvó *tf-idf súlyozást* használó vektortérmodell az egyik legelterjedtebb dokumentumreprezentációs modell.

A szó-dokumentum mátrix kialakítása természetesen számos kérdést felvet. Egyrészt létre kell hozni egy szótárt, melyhez kapcsolódóan a dokumentum szavainak feldolgozását kell megvalósítani. Ehhez a dokumentumot általában tokenekre szokás bontani, amely leggyakoribb esetben a szavakra bontást jelenti. Az azonos karaktersorozatokat tartalmazó tokeneket osztályozva létrejönnek az úgynevezett típusok, s ezen típusokból épül fel a nyers

szótár. Másrészt, mivel a szó-dokumentum mátrix teljes formájában nagyon nagy méreteket ölt, ezért érdemes csökkenteni a dimenzióját. Ezt a célt szolgálja a stopszavak törlése a mátrixból, amely azon szavaknak megfelelő sorok törlését jelenti, amely szavak nagy gyakorisággal fordulnak elő a dokumentumgyűjteményben, s ezáltal plusz információt várhatóan nem hordoznak. A stopszavak listája a dokumentumgyűjtemény elemzése révén jön létre, majd felhasználói megerősítést követően válik véglegessé. A vektortér dimenziója tovább csökkenthető a különféle dimenziócsökkentési eljárások által. A dimenziócsökkentés létrejöhet a releváns jellemzők kiválasztásával, illetve a jellemzők kombinálása által is. A stopszószűrés is tulajdonképpen egy jellemzőszelekciós eljárás, azonban a jellemzőszelektálás nem csupán nyelvi megfontolásokon alapulhat, hanem számos matematikai megközelítés eredményképpen is létrejöhet (pl. információnyereség elvének, vagy χ-négyzet statisztika alkalmazásával). A jellemzők egyesítésének módszere a rendelkezésre álló nagy számosságú jellemzőhalmazt kombinálja kisebb számosságúvá. Erre a célra leggyakrabban a szinguláris értékfelbontás (singular value decomposition) és a főkomponens analízis által nyújtott matematikai vektortranszformációs eszközök használatosak.

Mindezen alapokból kiindulva a szöveges dokumentumok feldolgozásának, elemzésének számos érdekes válfaja létezik. A szövegek osztályozása során a cél a dokumentumoknak előre meghatározott osztályokba, illetve tematikus kategóriarendszerekbe történő besorolása. Mivel a dokumentumreprezentációs vektortérmodell alkalmas a dokumentumgyűjtemény dokumentumai közt fennálló hasonlóság, illetve különbözőség kiszámítására, ezért ez alapján elvégezhető a dokumentumok csoportosítása is, amely főként a keresési feladatok végrehajtása során nyújt jelentős segítséget. A nagy adathalmazok, dokumentumok gyors áttekintését és feldolgozást különféle módszerekkel támogatja a szövegbányászat. Léteznek kivonatoló, összefoglalás-készítő technikák, amelyek a szövegekből automatikus módon generálnak rövid, összefoglaló leírásokat. A szövegekből történő információkinyerési módszerek szintén a szövegek gyors feldolgozását szolgálják. Mindezek mellett számos egyéb szövegbányászati funkció, alkalmazási lehetőség létezik, s a témakör fejlődése dinamikusan halad tovább. A leggyakrabban alkalmazott technikákba, felhasználási lehetőségekbe a [35] irodalom nyújt részletes betekintést.

5. fejezet

Adattárházak

5.1. Az adattárházak létjogosultsága, fogalma

Az adattárházak kialakulása, s az általuk nyújtott adatelemzési lehetőségek története az 1980-as évek végére nyúlik vissza, amikor is egy, az IBM kutatói által írt cikkben [8] bemutatták az általuk fejlesztett üzleti döntéshozó rendszert, s azt a modellt, amely segítségével ezen döntéshozó rendszer az operatív feladatokat ellátó adatbázisrendszerekből létrehozható. De mit is jelent az adattárház fogalma, illetve milyen igény alapján jöttek létre?

Egy adott szakterület menedzserei (pl. gazdasági vezetők) által a mindennapok során végrehajtott elemzések célirányos kérdésekre, összefüggésekre, változásokra keresik a válaszokat. Egy nagyvállalat életében számos úgynevezett *operatív adatbázisrendszert* alkalmaznak, melyek célja az adott részterület mindennapi feladatainak ellátása. Ilyen operatív adatbázisrendszernek tekinthető külön-külön például egy raktárnyilvántartó rendszer, egy számlázó rendszer, egy alkalmazottakkal és beosztásukkal kapcsolatos nyilvántartás, illetve egy könyvelési alkalmazás is. Ezen alkalmazások részben átfedő adatokat tárolnak a vállalat egészére vonatkozóan. Mindemellett azonban mindegyik rendszer alkalmazásának megvan az elsődleges célja, feladata, melynek kapcsán az alkalmazások sajátos beépített elemeket, programmodulokat tartalmaznak, s ezáltal speciális feladatok ellátását teszik lehetővé. Ezen rendszerek mindamellett, hogy a fejlesztésüket meghatározó elsődleges célokat megfelelően ellátják, viszonylag szegényes, csak az adott részterületre vonatkozó elemzési lehetőségeket biztosítanak. A különféle heterogén rendszerek az imént említett kapcsolódási pontjaik alapján azonban egy egységes rendszerbe, architektúrába szervezhetők anélkül, hogy az alkalmazások speciális elemei sérülnének. Ez az egységbe szervezés jelen esetben nem egy univerzális, minden funkciót betöltő új komplex alkalmazás létrehozását jelenti, hanem egy olyan új rendszer kialakítását, amely mintegy a meglévő alkalmazások „fölé” hoz létre egy újabb alkalmazást. Ezen alkalmazás célja olyan átfogó elemzések végrehajtása, melyek az egymástól független adatbázisrendszerek adatain alapulva komplex kérdések megválaszolását teszik lehetővé. Ezeket, a szervezet adatait összegyűjtő, s az átfogó elemzéseket szolgáltató technológiákat együttesen szokás adattárháznak nevezni.

Az adattárháznak számos definíciója létezik, melyek bár ugyanazon filozófiát tükrözik, kis mértékben mégis eltérnek egymástól. A legelterjedtebb definíció talán Inmon-tól származik, kinek megfogalmazásában az *adattárház* az adatoknak egy témaorientált, integrált,

időfüggő, nem illékony gyűjteménye, a vezetői döntések támogatása céljából [16]. Mivel a témaorientált, integrált, időfüggő és nem illékony jelzők az adattárházak leglényegesebb tulajdonságaira hívják fel a figyelmet, ezért érdemes részletesen végignézni, hogy mit is jelentenek pontosabban.

- Az adattárház *témaorientált*, mivel mindig valamilyen konkrét témakörrel (pl. termékek értékesítése) kapcsolatos adatokat foglal össze azon célból, hogy a vizsgált témakörben rendelkezésre álló adatok alapján a témakörön belül gyors, hatékony kiértékelést, döntéshozatalt biztosítson. A vizsgált témakört tekintve az adattárház egyszerű és tömör nézetét nyújtja az adatoknak, s nem tartalmaz olyan adatokat, melyek csupán a napi operatív feladatok elvégzéséhez szükségesek, de nem fontosak a döntéshozatal szempontjából.

- Az adattárház *integrált*, mivel több heterogén adatbázis, adatforrás egyesítésével jön létre. A különböző adatforrásokból származó adatok átkonvertálása az adattárházba egy rendkívül összetett folyamat, mivel ezen tevékenység során számos adattisztítási és adatintegrációs feladatot kell megoldani.

- Az adattárház *időfüggő* (idő-variáns) jellegét az adja, hogy a benne tárolt adatok általában historikus jellegűek, vagyis a vizsgált témakör jellemző adatai hosszabb időszakra visszamenően elérhetőek. Míg egy napi operatív feladatokat ellátó adatbázisrendszer működésének elengedhetetlen feltétele az éppen valós, aktuális adatok tárolása és kezelése, addig a stratégiai elemzések, vezetői döntések a historikus adatok elemzésén alapulnak. Ennek megfelelően az adattárházban az időhorizont jelentősen hosszabb, mint az operatív feladatokat ellátó adatbázisrendszerekben, továbbá működésükhöz az sem elengedhetetlen feltétel, hogy naprakész adatokat tartalmazzanak.

- Az adattárház *adatai nem illékonyak*, mivel az adattárházból csak nagyon ritka esetben törlődnek adatok, a már bent lévő adatok pedig alapvetően változatlanok. A napi operatív feladatokat ellátó adatbázisrendszerekből természetesen bizonyos időközönként átkerülnek az új adatok az adattárházakba is, azonban ezek az új adatok az adattárház régi adatait nem írják felül, hanem időbélyeggel ellátva új értékekként kerülnek tárolásra.

Az Inmon féle adattárház definíció utal még arra is, hogy az adattárházak elsődleges feladata a vezetői döntések támogatása. Ezt a momentumot számos egyéb adattárház definíció nem is tartalmazza, mivel az adattárházak alkalmazása egyéb célokat is támogathat, de jellemzően igaz, hogy az adattárház alkalmazások által biztosított elemzési, kiértékelési lehetőségek leginkább a menedzserek, vezetők döntéshozatali mechanizmusában jutnak kiemelkedő szerephez.

Az adattárházak fogalma szorosan összefonódott az *OLAP* (online analytical processing), vagyis az *online analitikai feldolgozás* fogalmával, melyet gyakorta szokás szembeállítani az *OLTP* (online transaction processing), tehát az *online tranzakciófeldolgozás* fogalmával. Ezen fogalmak tulajdonképpen két, egymástól lényegileg eltérő adatkezelési módszert takarnak. A hagyományos, mindennapi operatív feladatok ellátását szolgáló adatbázisrendszerekben a felhasználói kérések feldolgozása tranzakciókezelés által valósul meg. Ezek a

tranzakciók jellemzően gyorsan lefutnak, kevés adatot érintenek és az adatbázis aktuális adatain dolgoznak. Mindemellett, az alkalmazások jellegéből adódóan számos tranzakció futhat egymás mellett, melyek együttműködésének megoldása fontos feladat. Ezzel szemben az OLAP rendszerek – melyek az adattárházakban jellegzetes módon nyilvánulnak meg – elsősorban nagy mennyiségű, historikus adatok elemzését valósítják meg hatékony módon. Ezen rendszerekre kevésbé jellemző a párhuzamosság, a tranzakciók az adatokat legtöbbször csak olvassák és nem írják, viszont az egyes tranzakciós műveletek sokkal nagyobb adatmennyiséget fognak át, s általában hosszabb ideig futnak. Ebből adódóan az OLTP és OLAP alkalmazások tervezése lényeges eltérést mutat, ugyanis míg a hagyományos OLTP rendszerek általában a koncepcionális modellek (pl. Egyed-Kapcsolat Modell) relációs implementációján alapulnak, addig az OLAP alkalmazások az 5.2 fejezetben bemutatásra kerülő többdimenziós adatmodellt valósítják meg. Láthatjuk tehát, hogy az OLTP és OLAP rendszerek más-más funkciókat látnak el, s ebből fakadóan teljesen eltérő tulajdonságokkal rendelkeznek. Az OLTP és OLAP rendszerek főbb eltéréseit a 5.1 táblázat foglalja össze.

Jellemző	**OLTP**	**OLTP**
Funkció	napi feladatok ellátása	adatelemzés
Felhasználók	adatrögzítők	vezetők, menedzserek
Adatok	aktuális, részletes	historikus, összesített
Adatelérés	írás és olvasás	legtöbbször olvasás
Munka egysége	rövid, egyszerű tranzakciók	komplex lekérdezések
Elért adatmennyiség	általában kevés rekord	jellemzően sok adat
Adatbázis mérete	pár MB-GB	jellemzően nagyobb (GB, TB)

5.1. táblázat. Az OLTP és OLAP rendszerek főbb eltérései

Az adattárházak tehát a napi operatív feladatokat ellátó adatbázisrendszerek mellett, azokkal mintegy együttműködve biztosítják az online adatelemzés lehetőségét a szakértők számára. A következő fejezetekben ezen adatelemzési alapfogalmakat és lehetőségeket tekintjük át részletesebben.

5.2. A többdimenziós adatmodell

Az adatmodellek a modellezni kívánt valóságot írják le különféle szinteken. A koncepcionális, vagy más néven magas szintű adatmodellek az emberi gondolkodásmódhoz közel álló absztrakt megfogalmazásai a modellezni kívánt adathalmaznak, valóságnak. Az alacsony szintű, vagy más néven logikai adatmodellek az adatok logikai szervezését emelik ki, a tényleges implementációhoz közel álló, de továbbra is absztrakt megfogalmazásai a modellezendő témakörnek. Az adatmodellek fizikai szintje az adatok tényleges tárolásának leírását jelenti.

Míg a hagyományos adatbázisrendszerek az adatmodellek logikai szintjét tekintve általában relációs adatmodellen alapulnak, addig az adattárházak a többdimenziós adatmodellt implementálják. Ezen adatmodell a relációs modellhez képest teljesen új fogalmakat használ, melyek közül legfontosabbak az adatkocka, a dimenzió, a dimenziók hierarchiája és a tényadat. A következőkben tekintsük át a többdimenziós adatmodell fontosabb definícióit.

A *többdimenziós adatmodell* célja az elemezni kívánt adathalmaznak az elemzési szempontokat kiemelő absztrakt leírása, modellezése. A többdimenziós adatmodell az adatokat dimenziók mentén ábrázolja, s mint látni fogjuk ezen dimenziókhoz hierarchiákat határoz meg. A dimenziók által létrejön az adatkocka struktúrája, melynek egyes cellái a tényadatok alapján számítódnak ki. De mit is jelentenek ezek a fogalmak pontosan?

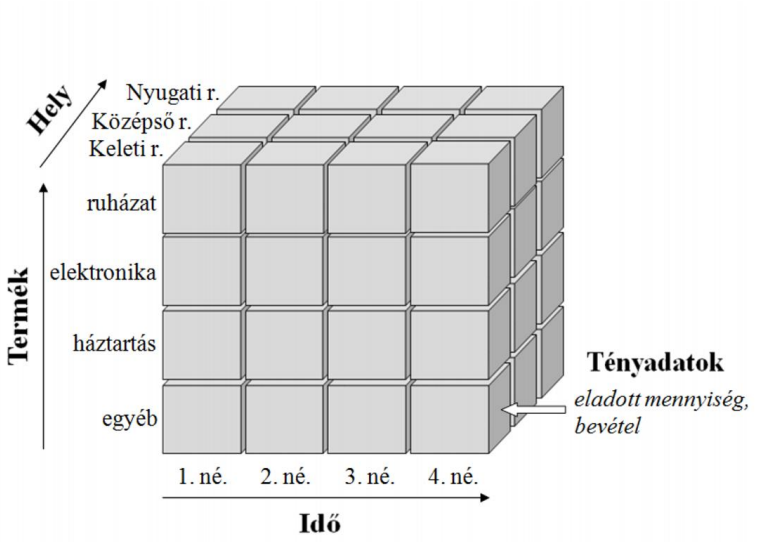
Tényadatoknak nevezzük a vizsgált témakör azon jellemző tulajdonságait (adatait), melyeket elemezni szeretnénk. Ezen adatok jellemzően numerikus értékek, melyek az egyes dimenziók mentén általánosabb szintre aggregálhatóak, illetve részletesebb kifejtésbe bonthatóak. Egy bolti értékesítés esetén elsődlegesen az eladott áruk mennyisége, a bevétel, a felmerült költségek pontos értéke, illetve ezek változása mentén fogalmazhatók meg az elemzői kérdések. Ennek megfelelően a kialakítandó adatkocka tényadatai az ezen adatokat tartalmazó tulajdonságok értékei.

Dimenzióknak nevezzük a vizsgált témakör azon tulajdonságait, melyek a tényadatokat nem átfedő csoportokba kategorizálják. Ezen dimenziók elsődleges célja a tényadatok csoportosítása, szűrése és címkézése. A termékek értékesítésének vizsgálata során tipikus dimenzió jellegű tulajdonság lehet az időt, a helyet, vagy a termék típusát leíró attribútum. Minden egyes dimenzió értékkészlete külön-külön hierarchiába szervezhető, vagyis a dimenzió által felvett értékek meghatározható szabály szerint egymásba ágyazhatóak. Egy-egy dimenzióra akár több hierarchia is meghatározható. Ezeket a hierarchiákat nevezzük a *dimenzió hierarchiájának*. Az idő dimenzió egyik lehetséges hierarchiájaként például a nap-hét-hónap-negyedév-év lebontást, a hely dimenzió egy lehetséges hierarchiájaként pedig például a bolt-település-megye-ország besorolást határozhatjuk meg. Mint a következőkben látni fogjuk, az adatkockán végezhető műveletek egy része az egyes dimenziókhoz rendelt hierarchiaszintek megváltoztatásán alapul.

Az *adatkocka* a tényadatok dimenziók mentén történő szemléltetése. Az imént említett példánál maradva amennyiben a bevételt, mint tényadatot szeretnék elemezni az idő, a hely és a terméktípus dimenziók mentén, akkor egy 3-dimenziós adatkockát kapunk, ahol az egyes dimenziók kategóriái alkotják a kocka éleit, a dimenzióknak megfelelő összesített bevételi értékek pedig a dimenzióértékek metszéspontjaiban képzelendők el. Természetesen nagyobb dimenziószám esetén már nem tényleges kockára, hanem „hiperkockára” kell gondolnunk, melyet az egyszerűség kedvéért szintén adatkockának szokás nevezni.

A tényadatok, a dimenziók és a belőlük összeálló adatkocka szemléltetés bemutatása a 5.1 ábrán látható. Az ábra a klasszikus adattárház példát szemlélteti, melyben egy több kereskedelmi egységet felölelő áruházlánc értékesítési adatait szeretnénk elemezni. Ezen elemzés céljából az eladott áruk mennyiségét és a bevételt, mint tényadatokat az idő, a hely és a termékkategória dimenziók mentén ábrázoljuk és értékeljük. Az egyes dimenziókhoz képzeljük el a következő hierarchiákat: idő: nap-hét-hónap-negyedév-év; hely: bolt-régió-ország; termékkategória: termék-alkategória-főkategória. A *kocka részletezettségi szintje* - melyet szokás az információ granuláltságának is nevezni - attól függ, hogy az egyes dimenziók mentén

a hozzájuk meghatározott hierarchia mely szintjét ábrázoljuk. A 5.1 ábrán az idő dimenzió mentén a negyedév, a hely dimenzió mentén a régió, a termékkategória dimenzió mentén a főkategória hierarchiaszintek szerinti értékek látszanak. Természetesen az adatkockák más és más részletezettségi szinten is megtekinthetőek a dimenziókhoz definiált hierarchiákból adódóan. Általában jellemző, hogy a felsővezetőket a kevésbé részletes lebontás, míg a középvezetőket és az alsóbb vezetőket az ő hatáskörüket érintő, részletesebb lebontás érdekli. Az ezen nézetek kialakításhoz kapcsolódó adatkocka műveleteket a 5.3 fejezetben mutatjuk be.



5.1. ábra. Adatkocka

Az adattárház alapú elemzések tehát ezen logikai adatmodell vizuális böngészésén alapulnak. Mielőtt rátérnék az adattárházak által biztosított elemzési lehetőségek részletes bemutatására, röviden tekintsük át, hogy milyen koncepcionális adatmodellek, illetve fizikai megvalósítás kötődik a többdimenziós adatmodellekhez.

Mint ismert, a koncepcionális adatbázis-tervezés során az Egyed-Kapcsolat Modellek hatékony segítséget nyújtanak a relációs adatmodellek kialakításához. Miután a többdimenziós modell teljesen más struktúrán alapszik, mint a relációs adatmodell, ezért az Egyed-Kapcsolat Modell az eredeti formájában nem alkalmas a többdimenziós gondolkodásmód szemléltetésére. Ezen okból kiindulva számos javaslat látott napvilágot az Egyed-Kapcsolat Modell többdimenziós kiterjesztésére vonatkozóan (pl. [19], [32]). Miután egységesen elfogadott, követendő stratégia nem létezik, ezért ezen adatmodell javaslatok egymás mellett párhuzamosan fejlődnek, s a tervezők maguk választják meg, hogy melyik modellt preferálják. Mindezek mellett számos objektum orientált tervezési módszer is létezik a többdimenziós adatbázisokhoz kapcsolódóan (pl. [27], [37]), de egységes stratégia ezen a területen sem alakult még ki.

A többdimenziós adatmodell megvalósítása a különféle rendszerekben különféle módon történik. Az alapján, hogy az egyes adattárház implementációk a többdimenziós adatmodell megvalósítása során milyen mértékben nyúlnak vissza a relációs adatbázis sémához meg-

különböztetünk MOLAP, ROLAP és HOLAP rendszereket. A *MOLAP* (Multidimensional OLAP) rendszerek olyan adattárház megoldások, ahol az adatok tárolása a többdimenziós adatmodellre specializáltan történik. A MOLAP rendszerek szakítva a relációs szemlélettel az adatokat általában többdimenziós tömbökben tárolják, s ezen rendszerekben az adatkockában megjelenítendő aggregált adatok is többnyire tárolásra kerülnek. A többdimenziós struktúrához optimalizált tárolásból, indexelési és elérési technikából fakadóan ezeket a rendszereket rendkívül gyors adatlekérdezés jellemzi. Ezzel szemben a *ROLAP* (Relational OLAP) fogalma olyan adattárház alkalmazásokat takar, amelyek relációs vagy kiterjesztett-relációs adatbázis-kezelőket használnak az adatok tárolására és kezelésére. Elterjedésük főként a relációs gondolkodás térhódításából fakad, s előnyük, hogy nagy adatmennyiség esetén is könnyen skálázhatóak. Ezen rendszerek az adatkockákban megjelenítendő aggregált adatokat általában külön segédtáblázatban tárolják, melyek karbantartásának minősége nagy mértékben befolyásolja az OLAP lekérdezések pontosságát. A *HOLAP* (Hybrid OLAP) technológia az előző két módszer előnyeit ötvözi. A forrásadatok tárolása relációs adatbázis-kezelő rendszerekben valósul meg, ezáltal elérésük, aktualizálásuk könnyen megvalósítható, a hozzájuk kapcsolódó aggregált adatok tárolása viszont MOLAP technológiák alkalmazásával történik, így gyors lekérdezésmegválaszolást tesznek lehetővé.

Mivel jelen jegyzet célja elsősorban az adatelemzési lehetőségek áttekintése, ezért a ROLAP, MOLAP és HOLAP architektúrák megvalósításának további fontos kérdéseire (pl. indexelési technikák, csillag-, hópehely-, galaxis sémák tervezése) most nem térünk ki. Ezen témakörökben értékes leírások találhatók a következő irodalmakban: [1] [33].

5.3. Az adattárház alapú adatelemzés

Az adattárházak nyújtotta elemzési lehetőségek az adatkockákon végezhető műveletek által valósulnak meg. Mint már korábban említettük, az egyes adatkockák a felhasználók igényeinek megfelelően különböző nézeteket ölthetnek. Az elemzők az adatkockákon keresztül böngészik az adattárház adatait, s az adatokban megbúvó trendek, összefüggések az adatkockák manipulálása által válnak láthatóvá. Ha például egy adatkockában egy, vagy több dimenziót részletesebben kifejtünk, tehát a dimenzióhoz létrehozott hierarchiában alacsonyabb szintre lépünk, akkor az adattárház adataiba is részletesebb betekintést nyerünk.

A többdimenziós adatkockákhoz kapcsolódó műveletek nevei a magyar szakirodalomban rendkívül vegyes képet mutatnak. Egységesen elfogadott és meghonosodott magyar elnevezések hiányában leginkább az angol elnevezések használatosak. Ez okból fakadóan a következőkben az egyes OLAP műveletek definiálásakor mi is együtt használjuk az angol és leginkább használatos magyar kifejezéseket. A többdimenziós adatkockán végezhető főbb OLAP műveletek tehát a következők:

- *Felgöngyölítés (roll-up, aggregáció)*: A roll-up művelet során az adatkockában az adatok összevonása, aggregálása történik. Eredményeképpen az adatkocka egyes celláiban található értékek nagyobb intervallumokat ölelnek át, ezáltal globálisabb, átfogóbb következtetéseket vonhatunk le belőlük. Felgöngyölítést hajtunk végre, ha egy, vagy több dimenzió mentén magasabb hierarchiaszintre lépünk, illetve aggregált adatokat kapunk abban az esetben is, ha az adatkockából valamely dimenziót, vagy dimenziókat

töröljük. Az előbbi megoldást választjuk például, hogyha az idő dimenzió mentén a havi lebontásról áttérünk negyedéves, vagy éves részletezettségi szintre. A második lehetőséget választva, ha a példa adatkockánkból elhagyjuk a hely dimenziót, akkor az értékesítési adatokat nagyobb általánosságban, helytől függetlenül megjelenítve böngészhetjük.

- *Lefúrás (drill-down)*: A drill-down művelet az imént bemutatott roll-up művelet ellentettje. Alkalmazása által az adatokat részletesebb felbontásban tekintheti meg a felhasználó. Ilyen részletesebb felbontást új dimenziók bevezetésével, illetve a meglévő dimenziók hierarchiaszintjén lefelé, vagyis a részletesebb adatok irányába történő elmozdulás által érhetünk el.

- *Szeletelés (slice)*: A szeletelés egy adott dimenzión végrehajtott szelekció. Szeletelést hajtunk végre abban az esetben például, ha az idő dimenzió mentén kiválasztjuk az 1. negyedévi adatokat. Eredményeképpen a kocka egy szelete adódik, amely a kiválasztott értékkel kapcsolatos adatokat tartalmazza. Azon adatok, amelyek nem részei a szeletnek, azok nem a kiválasztott értékkel (jelen esetben az 1. negyedév) kapcsolatosak. A szeletelés műveletét leggyakrabban abban az esetben alkalmazzuk, ha valamely dimenzió típusú tulajdonság egy kiválasztott értékéhez kapcsolódó adatok vizsgálatát szeretnénk elvégezni.

- *Kockázás (dice)*: A kockázás művelete során nem csupán egy, hanem több dimenzió mentén is szelekciót hajtunk végre. Eredménye a szelekciók közös metszeteként adódó részkocka. A kockázás műveletét hajtjuk végre abban az esetben például, ha az idő dimenzió mentén kiválasztjuk az 1. és 2. negyedévet, a termékek közül a ruházati termékeket, a hely dimenzió mentén pedig a Nyugati régió adatait. Mint ebből a példából is látható, a kockázás művelete a vizsgált témakör egy részterületének analízisét hivatott elősegíteni.

- *Elforgatás (pivot)*: Az elforgatás művelete az adatok megjelenítésében jelent változtatást, méghozzá oly módon, hogy a dimenziók orientációja változik meg. Legegyszerűbb esetben ez megvalósulhat a sorok és oszlopok cseréjével. Például elforgatást hajtunk végre, ha egy kimutatást, melyben a termékek soronként, az idő pedig oszloponként szerepel úgy módosítunk, hogy az idő lesz a sorkomponens, a termékek pedig az oszlopkomponens. Számos adattárház alkalmazás megengedi, hogy az oszlopok, vagy a sorok mentén több dimenzió is szerepeljen. Amennyiben valamely dimenziót sorból oszlopba, illetve oszlopból sorba áthelyezünk, akkor szintén az elforgatás műveletét hajtjuk végre. Bár az elforgatás művelete nagyon egyszerű, használatával mégis pillanatok alatt létrehozhatunk olyan új jelentéseket, melyek teljesen más megvilágításba helyezik a vizsgált adatokat.

A fenti felsorolás a leggyakoribb OLAP műveleteket definiálta. Emellett léteznek egyéb műveletek is, melyek szintén az elemzők tevékenységét hivatottak segíteni. Ezek közül a leggyakrabban a következő két műveletet használatos:

- *Keresztülfúrás (drill across)*: A keresztülfúrás művelete több adatkocka együttes alkalmazásán alapul. Használatával az egyik adatkockáról a másik adatkockára ugor-

hatunk át ugyanazon dimenzióbeállítások mentén. Alkalmazása összetett elemzések során rendkívül hasznos, hiszen a meghatározott tulajdonságok mentén azonos értékekkel rendelkező adatok gyors összehasonlítását teszi lehetővé.

- *Részletezés (részletek kibontása, drill trough)*: A részletezés az eredeti adatbázis azon adatait (rekordjait) mutatja meg, amelyekből a kockában kiválasztott cella értéke származik. Ezen művelet tehát visszaadja az eredmény cella forrásadatait, s ezáltal tipikusan a kiugró értékek analízise során jelenthet nagy segítséget.

Az alapvető OLAP műveletek mellett az adattárház alkalmazások számos olyan egyéb lehetőséget is biztosítanak a felhasználók számára, melyek alkalmazásával az elemzések hatékonysága tovább növelhető. Ilyen például a különféle aggregációs függvények használata és a szerteágazó grafikai megjelenítések lehetősége. Mint már korábban említettük, az adatkocka egyes cellái aggregált adatokat tartalmaznak. Az összesített adatok kiszámításához a különféle adattárház alkalmazások számos beépített aggregációs függvényt kínálnak a felhasználók számára. Így például az alapvető összeg, darab, minimum, maximum és átlagértékek kiszámítása mellett az adatok szórása, kovarianciája is könnyen kiszámítható. Mindemellett az OLAP alkalmazások általában lehetőséget biztosítanak új, származtatott értékek kiszámítására is, melyek alkalmazása szintén az adatelemzők munkáját hivatott segíteni. A riportok eredményeinek különféle grafikonokon történő ábrázolása a kiszámított adatokat szemléletesebbé teszi, ezáltal a keresett összefüggések, trendek könnyebben felismerhetővé válnak. Mint korábban említettük, egy kép gyakran többet ér ezer számnál is, s ez az elv jelen esetben is hasonlóan igaz.

Az adattárház rendszerek alkalmazásának azonban további előnyei is vannak. A korábbi fejezetekben ismertetett dimenziócsökkentési és adatbányászati eljárások alkalmazása általában haladó informatikai ismereteket feltételez az elemzők részéről. Ezzel szemben az adattárházak alkalmazásának nagy előnye, hogy az elemzők mélyrehatóbb informatikai ismeretek nélkül hajthatják végre az OLAP műveleteket, s értelmezhetik az eredményül kapott jelentéseket. Az adattárház alkalmazások ugyanis olyan grafikus felhasználói felületet biztosítanak a felhasználók számára, amelyek által az adatkockán végezhető műveletek rendkívül könnyen elvégezhetőek, és az eredmények megjelenítése könnyen és dinamikusan változtatható. Ilyen egyszerűbb adatkocka-kezelési funkció és grafikus megjelenítés már az Excel programban, illetve az OpenOffice táblázatkezelő programjában is elérhető.

A mellékletben található `OLAP_demo.avi` fájl az Excel programban mutatja be az adatkocka létrehozását, valamint a felgöngyölítés, a lefúrás, a szeletelés és a részletezés műveletét. A bemutató anyagban láthatjuk, hogy bár 3-dimenziós adatkockát hoztunk létre, az Excel csak 2-dimenziós vetületét mutatja a kockának, s a harmadik dimenzió csupán szűrési feltételként jelenik meg.

Mindezen egyszerű kezelési lehetőségek mellett az adattárházak létrehozása természetesen körültekintő informatikai tevékenységet igényel, mely során meg kell oldani az adatok betöltésének és frissítésének problematikáját, s választani kell a rendelkezésre álló tárolási szerkezetek közül. Ezen túlmenően, a felhasználói igényeknek megfelelően meg kell tervezni és létre kell hozni a megfelelő adatkockákat, beleértve a dimenziók kiválasztását és az egyes dimenziók hierarchiájának definiálását. Összességében azonban elmondható, hogy megfe-

lelő adattárház alkalmazások létrehozásával olyan adatelemző eszközt adhatunk a szakértők kezébe, melynek használatával már önállóan is hatékony adatelemzést hajthatnak végre.

Irodalomjegyzék

[1] Abonyi J. (szerk): Adatbányászat - a hatékonyság eszköze. Computerbooks, 2006.

[2] P. Adriaans, D. Zantige: Adatbányászat. Panem, 2002.

[3] R. Bellman: Adaptive Controll Process: A Guided Tour. Princeton University Press, 1961.

[4] J.C. Bezdek: Numerical Taxonomy with Fuzzy Sets. *J. Math. Biol.*, 1, (1974), pp. 57–71.

[5] J.C. Bezdek: Pattern recognition with fuzzy objective function algorithms. New York: Plenum, 1981.

[6] Bodon F.: Adatbányászati algoritmusok.
http://www.cs.bme.hu/~bodon/magyar/adatbanyaszat/tanulmany/index.html

[7] I. Borg, P. Groenen: Modern Multidimensional Scaling: Theory and Applications. *Springer Series in Statistics*. Springer Verlag, New York, 1997.

[8] Barry A. Devlin, Paul T. Murphy: An Architecture for a Business and Information System. *IBM Systems Journal*, 27(1), (1988), pp. 60–80.

[9] J. C. Dunn: Well Separated Clusters and Optimal Fuzzy Partitions. *Journal Cybern.*, 4, (1974), pp. 95–104.

[10] S. Guha, R.Rastogi, K. Shim: Cure: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD Conference*, (1998), pp. 73–84.

[11] S. Guha, R.Rastogi, K. Shim: Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th ICDE*, (1999), pp. 512–521.

[12] Hunyadi L., Vita L.: Statisztika I. AULA Kiadó, 2008.

[13] J. Han, M. Kamber: Adatbányászat – Koncepciók és technikák. Panem, 2004.

[14] H. Hotelling: Analysis of a complex of statistical variables into principal components. *Journal of Education Psychology*, 24, (1933), pp. 417–441.

[15] A. Hyvärinen, J. Karhunen, E. Oja: Independent Component Analysis. John Wiley and Sons, 2001.

[16] W.H. Inmon: Building the data warehouse. Wiley, 2005. (Fourth edition)

[17] Iványi A. (szerk): Informatikai algoritmusok. ELTE Eötvös Kiadó, 2005.

[18] T. Jolliffe: Principal Component Analysis. Springer, New York, 1996.

[19] Anand S. Kamble: A conceptual model for multidimensional data. *Proceedings of Asia-Pacific Conference on Communications*, (2008), pp. 29–38.

[20] G. Karypis, E.-H. Han, V. Kumar: Chameleon: A hierarchical clustering algorithm using dynamic modeling. *COMPUTER*, 32 (1999), pp. 68–75.

[21] E. Keogh, M. Pazzani: A simple dimensionality reduction technique for fast similarity search in large time series databases. *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, (2000), pp. 122–133.

[22] J. Kittler: Feature set search algorithms. in: C.H. Chen (Ed.), *Pattern Recognition and Signal Processing*, Sijthoff and Noordhoff, Alphen aan den Rijn, Netherlands, (1978), pp. 41–60.

[23] T. Kohonen: Self-Organizing Maps. Springer, third edition, 2001.

[24] J.B. Kruskal: Multidimensional Scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1), (1964), pp. 1–27.

[25] Lukács O.: Matematikai statisztika. Műszaki Könyvkiadó, 2006.

[26] Münnich Á., Nagy Á., Abari K.: Többváltozós statisztika pszichológus hallgatók számára. http://psycho.unideb.hu/statisztika/

[27] T.B. Nguyen, A. Min Tjoa, R. Wagner: An Object Oriented Multidimensional Data Model for OLAP. *WAIM '00: Proceedings of the First International Conference on Web-Age Information Management*, (2000), pp. 69–82.

[28] Obádovics J. Gy.: Valószínűségszámítás és matematikai statisztika. Scolar Kft., 2009.

[29] P. Pudil, J. Novovicová, J. Kittler: Floating search methods in feature selection. *Pattern Recognigion Letters*, 15, (1994), pp. 1119–1125.

[30] S. T. Roweis and L. K. Saul: Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, Vol 290, (2000), pp. 2323–2326.

[31] J.W. Sammon: A non-linear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5), (1969), pp. 401–409.

[32] C. Sapia, M. Blaschka, G. Höfling, B. Dinter: Extending the E/R Model for the Multidimensional Paradigm. In *Advances in Database Technologies*, Lecture Notes in Computer Science, Vol. 1552, (1998), pp. 105–116.

[33] Sidló Csaba: Összefoglaló az adattárházak témaköréről. http://scs.web.elte.hu/Work/DW/adattarhazak.htm

[34] S.S. Stevens: On the theory of scales of measurement. *Science*, **103**, (1946), pp. 677–680.

[35] Tikk D. (szerk): Szövegbányászat. Typotex, 2007.

[36] J.B. Tenenbaum, V. Silva, and J.C. Langford: A global geometric framework for nonlinear dimensionality reduction. *Science*, **290**, (2000), pp. 2319–2323.

[37] J. Trujillo, M. Palomar, J. Gómez: An Object Oriented Approach to Multidimensional Databases & OLAP Operations. *International Journal of Computer & Information Science*, 1(2), (2000), pp. 75-85.

[38] J. Tukey: Exploratory Data Analysis. Addison-Wesley, 1977.

[39] Xie X.L., Beni G.: A validity measure for fuzzy clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(8), (1991), 841–847.

[40] Y. Wu and K.L. Chan: An extended isomap algorithm for learning multiclass manifold. In *Proceeding of IEEE International Conference on Machine Learning and Cybernetics (ICMLC2004)*, volume 6, (2004), pp. 3429–3433.

Tárgymutató