Adatbányászat feladatok
(2011. november 29)

(megoldás-**vázlatokkal**)

1. **Asszociációs szabályok**

a) Adott az alábbi tranzakciós adatbázis:

TID	**Items**
1	alma, körte, szilva
2	körte, szőlő, szilva
3	szőlő, banán, szilva
4	alma, banán, szilva, szőlő
5	körte, banán
6	alma, körte, banán
7	alma, körte, szőlő, banán

– Mekkora a { szőlő, banán } elemhalmaz támogatottsága (support-ja) ?
3/7

– Mekkora az { alma, körte } → {banán} asszociációs szabály i) támogatottsága, ii) konfidenciája, iii) lift-mutatója?

i) 2/7, ii) 2/3, iii) (2/7)/((3/7)*(5/7))

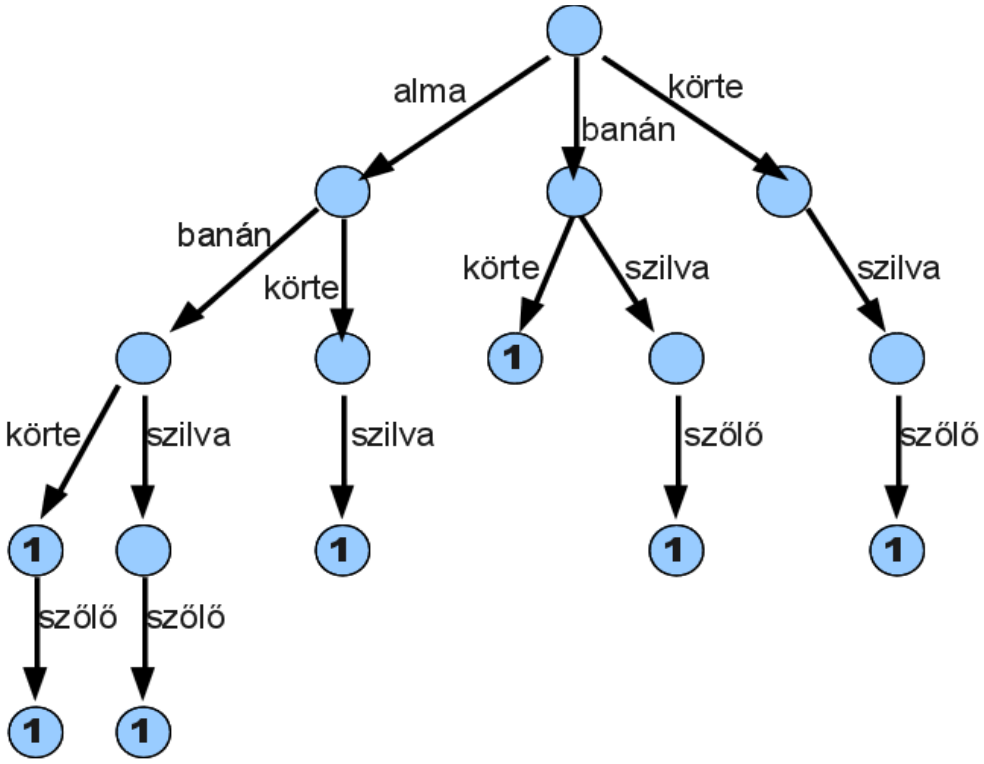
b)
– Milyen asszociációs szabály kereső algoritmusokat ismer?
APRIORI, ECLAT, FP-GROWTH
– Mely tulajdonságon alapul a legtöbb hatékony asszociációs szabály kereső algoritmus?
A support függvény antimonotonitásán: egy „összetett" minta csak akkor lehet gyakori, ha részmintái is gyakoriak.
– Bizonyítsa be, hogy az alábbi két asszociációs szabály konfidenciájára igaz az alábbi összefüggés!

1. asszociációs szabály: { a, b, c } → { d }
2. asszociációs szabály: { a, b } → { c, d }

Bizonyítandó összefüggés:
konfidencia({ a, b, c } → { d }) >= konfidencia ({ a, b } → { c, d })

konfidencia({a, b, c } → {d}) = support({a, b, c, d}) / support({ a, b, c}).
Mivel support({a, b, c}) <= support({a,b}), ezért:
support({a,b,c,d}) / support({a,b,c}) >= support({a,b,c,d}) / support({ a,b}) = = konfidencia ({ a, b } → { c, d })
Tehát: konfidencia({ a, b, c } → { d }) >= konfidencia ({ a, b } → { c, d })

c) Ábrázolja szófában (segítség: Christian Borgelt algoritmusa) az a) feladatpontbeli tranzakciós adatbázist! Az item-ek rendezése során válassza az ABC-sorrendet!



Miért hatékony Christian Borgelt támogatottság-számítási algoritmusa? (1-2 mondat)

A tranzakciókat és jelölteket egyaránt szófában tárolja, a támogatottság meghatározásakor alkalmazott rekurzív bejárásnak köszönhetően mind az azonos perfixű tranzakciókat, mind pedig az azonos prefixű jelölteket, ameddig csak lehet, egyidejűleg kezeli, a rekurzív hívások azon pontokon „ágaznak el”, ahol az azonos prefixű tranzakciók különböznek.

2. Osztályozó algoritmusok

a) A perceptron algoritmus és a szupport vektor gépek által létrehozott osztályozó modellek milyen lényeges közös tulajdonsággal rendelkeznek?

Mind a kettő egy „elválasztó hipersíkot” keres két osztály pontjai között, ezért alapvetően lineáris problémák megoldására alkalmasak. Az SVM-ek esetében azonban előfordulhat, hogy az eredeti adatot egy magasabb dimenziószámú térbe vetíti és ebben a magasabb dimenziószámú térben keresi az elválasztó hipersíkot. Ilyen módon az SVM-k nem-lineáris osztályozási problémák megoldására is alkalmasak.

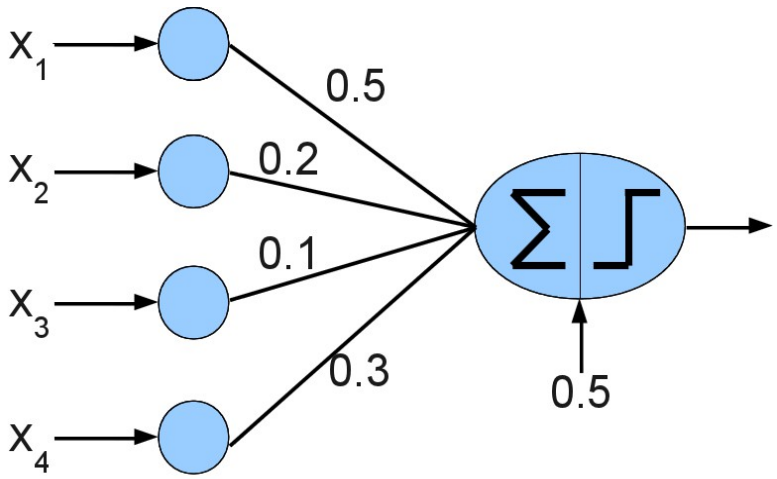
b) Mi a kapcsolat a szupport vektor gépek és a legközelebbi szomszéd osztályozó között?

Egy szupport vektor gép egy osztályozandó *x* adatpont címkéjét az alapján határozza meg, hogy *x* a két osztályt elválasztó hipersík melyik oldalára került. Ez – azaz, hogy *x* a hipersík melyik oldalára kerül – *x*-nek a szupport vektoroktól való távolsága alapján dönthető el. Így tehát egy szupport vektor gép ahhoz *hasonlóan* működik, mintha egy *k*-legközelebbi szomszéd osztályozó (megfelelő hasonlósági függvénnyel) csak a szupport vektorokat venné figyelembe, ezeket viszont súlyozottan, és közülük mindet, azaz *k* = szupport vektorok száma. (Megjegyzések: 1) A szupport vektor gépek ehhez hasonlóan működnek, de az osztályozási képlet *nem pontosan azonos* ezzel, lásd (5.62)-es formulát az „Introduction to Data Mining" c. könyvben.
2) A szupport vektorok meghatározása nem-triviális feladat, egy „standard" legközelebbi szomszéd osztályozó nem keresi meg a szupport vektorokat!)

c) Egy döntési fa tanításához használt adatbázisbeli osztályok eloszlása kiegyensúlyozatlan (imbalanced), azaz a pozitív osztály alulreprezentált (tanító adatok kb. 1-2%-a), míg a negatív osztály felülreprezentált (tanító adatok kb. 98-99%-a). Ezért az eredményként kapott modell végül minden teszt adatot a negatív osztályba sorol. Hogyan lehetne ezt a problémát orvosolni?

Mintavételezés: pozitív osztály oversampling-ja: ennek során a pozitív osztályból mintákat veszünk, ismételt mintavételezéssel, egészen addig, amíg a két osztály elemszáma nagyjából hasonlóvá nem válik. Mivel ehhez szükségszerűen egyazon pozitív objektumot többször is ki kell választanunk a mintába, duplikátumok jelennek meg, amelyeket minimális mértékben módosíthatunk (pl. egy-egy attribútum értékét kicsit megváltoztathatjuk).
VAGY: negatív osztály mintavételezésével: néhány mintát veszünk a negatív osztályból úgy, hogy kb. ugyanannyi legyen a pozitív és negatív minták száma a tanításhoz használt adatokban.
VAGY: Az előbbiekkel lényegében azonos a pozitív minták felülsúlyozása és a negatív minták alulsúlyozása.

d) Adott az alábbi perceptron, a két osztálycímkét +1-gyel és -1-gyel jelöljük.



Hogyan osztályozza a fenti perceptron ezt a mintát (adatpontot):
(Milyen osztálycímkét rendel hozzá?)
(x1=1, x2=-0.8, x3=-0.3, x4=1.5) ?

A +1 osztályba sorolja.

Tegyük fel, hogy az előbbi adatpont helyes osztálycímkéje +1. Ha ennek tudatában a fenti adatpontot a perceptron tanítása során használjuk, mely összeköttetések (élek) súlyai változnak, és hogyan (növekszik, csökken, változatlan marad) ?

A 0.5 és 0.3 súlyú élek súlya (kicsivel) növekszik, a másik két él súlya (kicsivel) csökken.

e) Miért lehetséges, hogy az ensemble modellek teljesítménye jobb, mint az ensemble-ben részt vevő modellek teljesítménye külön-külön? (Dietterich 3 indoka.)

Statisztikai ok: zajos és/vagy hiányos tanító-adatok miatt a modellek nem találják meg az optimális osztályozó függvényt, de annak közelében lesznek, és a modellek „átlaga" (többségi szavazása) közelebb lesz az optimális osztályozó függvényhez, mint a modellek külön-külön.

Számítási ok: a legtöbb osztályozó algoritmus valamilyen lokális keresést végez, nem garantált, hogy globális optimumot talál, előfordulhat, hogy csupán egy lokális optimumig jut el. (Tehát a talált osztályozó függvény csak lokálisan lesz optimális, nem globálisan.) Az ilyen lokálisan optimális osztályozási függvények „átlaga" (többségi szavazata) rendszerint jobb eredményt ad, mint az osztályozók külön-külön.

Reprezentációs ok: Elképzelhető, hogy az optimális osztályozó függvény nincs benne azon függvények terében, amelyben az algoritmus az optimális osztályozó függvényt keresi. Ugyanakkor ezekhez hasonló függvényeket sikerül megtalálni, melyek többségi szavazat közelítheti az optimális osztályozó függvényt.

Lásd. bővebben: T. Dietterich: Ensemble methods in machine learning

f) Mit mond ki a Bayes-tétel? Miért „naív" a NaiveBayes osztályozó algoritmus? Mutassa be egy (egyszerű) példán a NaiveBayes algoritmus működését!)

Lásd itt: http://www-users.cs.umn.edu/
~kumar/dmbook/dmslides/chap5_alternative_classification.pdf ,
49. ill. 57. fóiákon

3. Előfeldolgozás, dimenziószámcsökkentés

a) Mi a különbség az ISOMAP és MDS dimenziószám-csökkentő eljárások között?

Az ISOMAP egy szomszédossági gráfot készít és a pontpárok közti gráfban értelmezett legrövidebb út hosszát tekinti a két pont távolságának. Ezt követően hívja meg az MDS eljárást.

b) Egy automatikus jeltolmács rendszert készítünk. A jeltolmács rendszer feladata az, hogy a kézmozgások során jelbeszéddel „leírt" szavakat a jelbeszédet nem ismerő felhasználó számára érthető formában jelenítse meg. A rendszer egyik legfontosabb komponense a

jelbeszédi jelek felismerése. Ehhez egy ún. mozgásérzékelő kesztyűt (motion capturing glove) használ, amely az 5 ujj „behajlítottságának” nagyságát valamint a mutatóujj orientációját méri egymást kis időközönként követő pillanatokban. A „behajlítottságokat” százalékban (0-tól 100-ig) méri a rendszer, míg a mutatóujj orientációját fokban méri, 0 foknak felel meg, ha a mutatóujj felfelé mutat, 90 foknak ha a mutatóujj merőleges az előbbi irányra és jobbra mutat, 180 foknak, ha lefelé mutat, 270 foknak, ha balra mutat. A legnagyobb mért fokszám 359, ezután ismét a 0 fok következik. A felismerési feladatot egy legközelebbi szomszéd alapú osztályozóval szeretnénk megoldani. Szükséges-e az adatok valamilyen előfeldolgozása/konvertálása, ha igen, milyen előfeldolgozási/konverziós lépéseket javasol? Javasoljon hasonlósági mértéket a legközelebbi szomszéd alapú osztályozóhoz!

- Előfeldolgozás: a mutatóujj orientációja más skálán van, mint a többi változó, ezért normalizáció szükséges.
- Hasonlósági mérték: többváltozós DTW-n alapul. A DTW-számításakor a „belső” (lokális) hasonlósági mérték számításánál figyelembe kell venni, hogy a mutató-ujj orientációját mérő változó legnagyobb értéke „közel” van a legkisebb értékhez.

c) Az előző pontbeli jeltolmács rendszerben valamilyen technikai probléma miatt időnként előfordul, hogy nem sikerül a teljes jelet megfigyelni, hanem csak az első néhány tizedmásodpercet vagy másodpercet (nem ismert, hogy a jelbeszédi jelnek pontosan mekkora részét, felét, harmadát, kétharmadát, háromnegyedét, stb., sikerül előre megfigyelni). Ilyen esetben – amellett, hogy a rendszer jelzi a hibát a felhasználónak – a rendszer megkísérli a jelet a hiba ellenére is felismerni. Adaptálja az előző pontbeli hasonlósági mértéket és/vagy osztályozó algoritmust úgy, hogy akkor is (közelítőleg) helyes felismerést kapjunk, ha csak a jel eleje ismert. Tartsa szem előtt, hogy a felismerés számításigénye ne változzon aránytalanul sokat!

Tegyük fel, hogy az adatbázisbeli, teljes referencia-minta került a DTW mátrix „fölé” és a töredékes minta a DTW-mátrix mellé (lásd Buza: Fusion Methods for Time-Series Classification-beli elrendezést). A DTW definíciója alapján ekkor a DTW-mátrix utolsó oszlopának legkisebb (minimális) elemének nagysága egy jó hasonlósági mérték.
Az utolsó oszlopbeli minimális elem kiválasztása nem növeli jelentősen a számításidőt.

d) Miért adódnak nehézségek sokdimenziós adatok elemzése (pl. osztályozás, klaszterezés) során? Mi a „dimenziószám átka”?

- **az adatpontok várhatóan „ritkábbak” lesznek a nagy dimenziószámú térben (sparsity)**
- **távolságok „uniformizációja”**

4. Klaszterezés

a) Milyen kritériumokat kellene egy „jó” klaszterező algoritmusnak teljesítenie Kleinberg szerint? (1-1 mondat)

konzisztencia, gazdagság, skála-invariancia (lásd még Bodon Ferenc tanulmányában)

b) A hierarchikus és *k*-közép klaszterező algoritmusok közül melyik teljesíti a skála-invariancia kritériumát? Tételezzük fel, hogy mindkét algoritmus esetében rögzítjük a klaszterek számát.
Mind a kettőre teljesül a skála-invariancia.