Kovács Edith*, Szántai Tamás**

SPECIÁLIS GRÁFOKRA ÉPÜLŐ, CSŐDEELŐREJELZÉSEKNÉL IS HASZNÁLHATÓ KLASSZIFIKÁCIÓS MÓDSZER

Jelen cikkben arra óhajtunk rávilágítani, hogy hogyan alkalmazható a valószínűség-számítás a gépi tanulásban (machine learning), és mindez hogyan használható föl a klasszifikációs feladatokban. A cikk végén bemutatjuk a módszer alkalmazhatóságát vállalatok csődbejutásának előrejelzésére alkalmas pénzügyi feladatokra.

Az első részben szó lesz a gépi tanulásról, a klasszifikáció problémájáról, és annak a valószínűségi gráfmodellekkel való kapcsolódásáról. A második részben visszatekintünk a valószínűségi gráfmodellek eredetére, majd szót ejtünk a Markov- és a Bayes-hálózatokról. A harmadik részben az általunk bevezetett cseresznyefás módszert használjuk egy csődbejutás-klasszifikációs problémára.

Gépi tanulás

Hogyan is működik az agyunk? Hogyan reagálnak embertársaink különféle helyzetekre? Vajon meg tudjuk-e jósolni, hogy egyes lépéseinkre, cselekedeteinkre hogyan reagál egy ismerősünk? Lehet, hogy van néhány dolog, amire tudjuk, sok olyan is, amire (bizonyos valószínűséggel) sejtjük a választ, és jó néhány akad, amire nem tudunk válaszolni, mert adott kérdéskört még nem teszteltük hasonló helyzetben.

Egy géptől azonban elvárjuk, hogy a bevitt adatok (*input*) alapján eredményre jusson, reagáljon (*output*), ugyanakkor a mi feladatunk megtanítani, miből mire következtessen. Itt idézzük *Feynman* Nobel-díjas fizikust, aki azt mondta: „Egy kutyát jóval könnyebb kiismerni, mint egy embert. Azonban a kutyát sem tudjuk kiismerni.” A világ nagyon változatos: első látásra szinte megoldhatatlan problémának tűnik a megértése.

* *főiskolai docens, Általános Vállalkozási Főiskola*

** *egyetemi tanár, BME Matematika Intézet*

Jó példa a képfelismerés. Egy kép pixelekből áll. A pixelek színintenzitását számokká lehet konvertálni. Hogyan ismeri föl a fényképezőgép, ha egy arcot lát? Nyilván a pixelekhez tartozó változók alapján. Kérdés, mely pixeleket vegye figyelembe? Ha egy embernek tesszük föl a kérdést, gondolkodás nélkül rávágja: van két szeme, orra, szája – így ismeri fel az arcot, ilyen egyszerű. Ez tényleg egyszerű, de egy gépnek hogyan lehet mindezt megtanítani?

Vegyünk egy adatbázist, amiben N darab vektor van. Minden vektor tartalmazza n változó, és egy további $(n+1)$-edik, úgynevezett klasszifikáló változó értékét. Az előző példánál maradva, képzeljük el, hogy minden képhez tartozik egy vektor, ami tartalmaz n darab pixel-értéket. Az utolsó $(n+1)$-edik változó pedig 1, ha a képen van legalább egy arc és 0, ha nincs rajta arc. A gépet „megtanítjuk" N darab képre, tehát inputként N darab $n+1$ dimenziós vektort kap. Ha a gép mindenre emlékezni tudna, akkor, ha tesztként egy olyan képet „látna", amely az adatbázisában szerepel, mindjárt megtudnánk, van-e rajta arc vagy nincs. Ez az eljárás a következők miatt nem működik:

- Már egy fekete-fehér kép is – fölbontástól függően – jó néhány megapixelből áll, melyek mindegyike felvehet 256 különböző értéket. Ennek elraktározására nincs elegendő memória.
- Nem tudunk a gépnek minden lehetséges variációt megadni, lesznek olyan vektorok is, amelyek nem szerepelnek az adatbázisban.

A valószínűségi modellekre épülő tanulás ezekre a problémákra is próbál megoldást nyújtani. Az alábbiakban összefoglaljuk a gépi tanulás fő lépéseit:

1) Kódolás. A változók lehetséges „értékeihez" számokat rendelünk. Ha a változók folytonosak, akkor az értéktartományukat intervallumokra bontjuk, ha pedig minőségi tulajdonságokat megadó változókról van szó, akkor számokat rendelünk hozzájuk. A klasszifikáló változót Y-nal jelöljük. Így egy $(n+1)$ dimenziós $(X_1,\ldots,X_n,Y)$ együttes eloszlásból származott N nagyságú mintánk keletkezik. Ez a minta a *tanuló adatbázis* (*training set*).

2) Hozzárendelési függvény. Kell találni egy f hozzárendelési függvényt, amely egy tetszőleges n dimenziós értékvektorhoz hozzárendeli az Y klasszifikáló változó megfelelő értékét. Ezt a függvényt egy hipotézistérből kell kiválasztanunk. Ahhoz, hogy a kiválasztást meg tudjuk oldani, nagyon fontos, hogyan fogalmazzuk meg a hipotézisteret (ne legyen se túl általános, se túl specifikus).

3) Veszteség függvény. Szükség van egy L függvényre (*loss-function*), amellyel számszerűsíteni tudjuk az f hozzárendelési függvény „rosszaságát". Minél jobb az f hozzárendelési függvény, az L függvény értékének annál kisebbnek kell lennie.

Tegyük föl, hogy f hozzárendeli az $\mathbf{x}^j = \left(x_1^j,\ldots x_n^j\right)$ vektorhoz az $f(\mathbf{x}^j)$ klasszifikáló értéket. A mintában szereplő klasszifikáló értéket jelöljük $Y(\mathbf{x}^j)$-vel. Például a következő függvények lehetnek L függvények:

- $L = \sum_{j=1}^{N} 1_{\left[Y\left(\mathbf{x}^j\right) \neq f\left(\mathbf{x}^j\right)\right]}$ (annyiszor adódik össze az 1, ahányszor a valódi klasszifikáló érték és az f függvénnyel nyert klasszifikáló érték nem egyezik).
- $L = \sum_{j=1}^{N} \left\| Y\left(\mathbf{x}^j\right) - f\left(\mathbf{x}^j\right) \right\|^2$ (eltérések négyzetének összegét – a regressziószámításban alkalmaztuk).

■ Az *f* hozzárendelési függvény jósága úgy is mérhető, hogy vesszük a tanuló adatbázis minta empirikus eloszlásának és a hozzárendelési függvénnyel képezett (n+1)-dimenziós empirikus eloszlásnak *Kullback – Leibler*-távolságát:

$$KL = \sum_{j=1}^{N} P\left(\mathbf{x}^j, Y\left(\mathbf{x}^j\right)\right) log_2 \frac{P\left(\mathbf{x}^j, Y\left(\mathbf{x}^j\right)\right)}{P\left(\mathbf{x}^j, f\left(\mathbf{x}^j\right)\right)}.$$

A cél olyan *f* függvény kiválasztása, amely minimalizálja az *L* függvényt. Ezt például elérhetjük úgy is, hogy az *f* függvényt egy paraméteres függvénycsaládból választjuk, és az *L* minimalizálásával határozzuk meg a megfelelő paramétert (ilyen a lineáris regresszióillesztésnél az egyenes paramétereinek a meghatározása), vagy egyszerűen egy véges függvényhalmazból kiválaszthatjuk a legkisebb *L* értéket eredményezőt.

4) *Hibatesztelés.* Ez a lépés kétféle tesztelést tartalmaz. A tanuló adathalmazon történő hibázás számszerűsítését és az újabb tesztadatokon történő hibázás mérését. A rendelkezésünkre álló adathalmaz alapján megfogalmazzuk az *f* függvény hipotézisterét, majd ebből kiválasztunk egy olyan *f* függvényt, amely minimalizálja az *L* függvény értékét.
A tanuló adatokon számolt átlagos *L* függvényértéket nevezzük tanulási hibának (*training error*).
Az újabb tesztadatokon is alkalmazzuk a kiválasztott *f* függvényt, majd kiszámítjuk az *L* függvény értékét. Ez lesz az úgy nevezett tesztelési hiba (*test error*).

Fölmerül a kérdés, hogy ha a tanuló adatbázisunkon jól működik az *f* függvény, *L* elég kicsi, akkor van-e garancia arra, hogy az újabb tesztadatokon is beválik, vagyis ott is elég kicsi lesz az *L* értéke? Erre a kérdésre *Sam Roweis* előadásából kaphatjuk meg a választ (Roweis, 2006). Ott elhangzik, hogy a tanuláselméletben bizonyították a tételt, amely leegyszerűsítve így hangzik: „ha az *f* függvény jól működik egy elég nagy tanuló adatbázison, és nem túl komplex, akkor jól fog működni a tesztelő adathalmazon is. Pontosabban felülről korlátozható annak a valószínűsége, hogy sokkal rosszabbul működjön a teszt adathalmazon, mint tette azt a tanuló adathalmazon". Ez az állítás azért is tűnik természetesnek, mivel feltételezzük, hogy a tanuló minta eloszlása ugyanaz, mint a sokaságé.

Érdemes kitérnünk az előbbi mondatnak arra a megjegyzésére, hogy „nem túl komplex". Ezt egy példával támasztjuk alá: sokak számára lehet ismerős a *YouTube*. Ha valaki regisztrált, és a megnézett klipeket a tetszésének megfelelő számú csillaggal jelölte, akkor minden alkalommal, amikor bejelentkezik a honlapra, valamilyen ajánlatot kap, hogy mit lenne érdemes megnéznie. Ezt két módon generálhatja a program: tartalom alapján (ki szerepel rajta, milyen típusú zene, milyen hosszú, mikori stb.), vagy annak alapján, hogy azok, akik megnézték, és értékelték, milyen egyéb klipeket néztek meg. A gyakorlat bebizonyította, hogy a második módszer a legtöbb esetben hatékonyabb, pedig az első módszer több információt használ föl. Amikor több információt használunk föl, mint amennyi szükséges, és ezáltal rontunk a modellen, azt túlhatározottságnak[1] (*overfitting*) nevezik.

Ebben a cikkben a tanuló adatbázis alapján törekszünk megtanulni az $(X_1,\ldots,X_n,Y)$ együttes eloszlását. Ehhez egy keresési teret szükséges definiálni, hacsak nem valami speciális eset-

[1] Ockham's rasor *néven is ismert.*

tel állunk szemben, amikor például tudjuk, hogy normális az eloszlás, melynek csak a paramétereit kell megfelelően megválasztani. Általában azonban sokféle eloszlás közül választhatunk. Feltételezhetnénk például azt is, hogy az összes X_i változó független egymástól. Ekkor:

$$P(X_1,\ldots,X_n) = P(X_1)\cdots P(X_n),$$

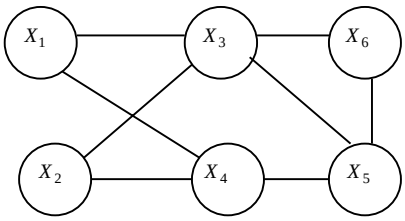
és az X_i változók együttes eloszlásához csak egyváltozós eloszlásokat kellene meghatározni. Ha azonban az X_i változó $m_i, i = 1,\ldots,n,$ az Y változó pedig m_Y különböző értéket vehet fel, akkor akár $m_1 \times \cdots \times m_n \times m_Y$ értéket is tárolni kellene, ami túl sok memóriát igényelne, mivel már *n* is igen nagy szám lehet (gondoljunk a pixelek számára, és hogy az egyes pixelek is 256 különböző értékeket vehetnek föl). Ez nagyon sarkított és haszontalan feltételezés lenne. Sokkal természetesebb, ha feltételes függetlenségeket tételezünk fel. Ezeket az úgynevezett valószínűségi gráfmodellekben is meg lehet jeleníteni.

Valószínűségi gráfmodellek az Ising-modelltől a Markov-hálókon át a Bayes-hálózatokig

Az első feltételes függetlenséget tartalmazó modell, *Ernst Ising*[2] német fizikustól származik (Ising, 1925), a modell kidolgozását tanára, *Wilhelm Lenz* (1888–1957) inspirálta. 1925-ben írt PhD-dolgozatában egy modellt adott meg, amely a mai napig ismert és a nevét viseli. Az Ising-modell feltételezi, hogy csak a szomszédos *spin*ek[3] között van kölcsönhatás, vagyis az egész rendszer csak a szomszédain keresztül hat minden egyes spinre. Ezt a modellt a mágnesességgel kapcsolatban tárgyalja Brush (1967).

Egy általánosabb, feltételes függetlenséget tartalmazó modell a Markov-modell vagy Markov-hálózat (*Markov random field*). Ez egy valószínűségi gráfmodell, amelyben a csúcsok valószínűségi változók (amelyeket ismérvekhez rendelünk hozzá), az élek hiánya pedig a feltételes függetlenséget jelzi (lásd: 1. ábra).

1. ábra



[2] *www.bradley.edu/las/phy/personnel/ising.html*

[3] *A* spin *a részecske térbeli transzformációs tulajdonságait leíró hullámfüggvény (állapotfüggvény). Az Ising-modellben két pozíciója van: fel vagy le. Ha felfelé mutat +1, ha lefelé, akkor -1 értéket vesz föl.*

Kicsit sarkosabban fogalmazva, minden valószínűségi változóra azok a valószínűségi változók hatnak „direkt” módon, amelyekkel él köti össze. Ez nem jelenti azt, hogy a többi változótól nem függnek. Ez képlettel röviden a következő módon írható le:

$$P(X_k \mid \mathbf{X}_{SZ}) = P(X_k \mid (X_1,\ldots,X_{k-1},X_{k+1},\ldots,X_n)),$$

ahol $\mathbf{X}_{SZ}$ a szomszédos valószínűségi változók vektora.

Az 1. ábrán látható X_5 változóra az X_3 X_4 és X_6 hatnak direkt módon, azaz

$$P(X_5 \mid X_3,X_4,X_6) = P(X_5 \mid X_1,X_2,X_3,X_4,X_6).$$

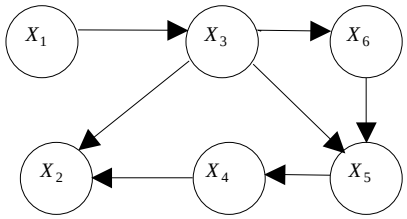
A változók mögött meghúzódó Markov-hálózat feltárása azért is hasznos, mert a gráfból megtudhatjuk, melyik valószínűségi változókat kell figyelembe vennünk egy adott valószínűségi változó tanulmányozása érdekében. Ehhez kapcsolódó fontos tételeket 1971-ben *Hammersley* és *Clifford* mondott ki és bizonyított be (publikálás nélkül), a témakör első publikációja azonban *Besag* nevéhez fűződik (Besag, 1974).

Egy másik fontos valószínűségi gráfmodell a Bayes-háló (*Bayes Network*). Ehhez már egy irányított körmentes gráf tartozik és a hozzárendelt együttes valószínűség-eloszlás:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)),$$

ahol $pa(X_i)$ az X_i szülei abban az értelemben, hogy $pa(X_i)$ minden csúcsából indul nyíl X_i-be. Tekintsük példaként a 2. ábra Bayes-hálóját:

2. ábra



Az ehhez rendelt együttes valószínűség-eloszlás:

$$P(\mathbf{X}) = P(X_1)P(X_2 \mid X_3,X_4)P(X_3 \mid X_1)P(X_4 \mid X_5)P(X_5 \mid X_3,X_6)P(X_6 \mid X_3).$$

A Bayes-hálókat gyakran használják például szövegfelismerési feladatokban (McCallum és Nigam, 1998).

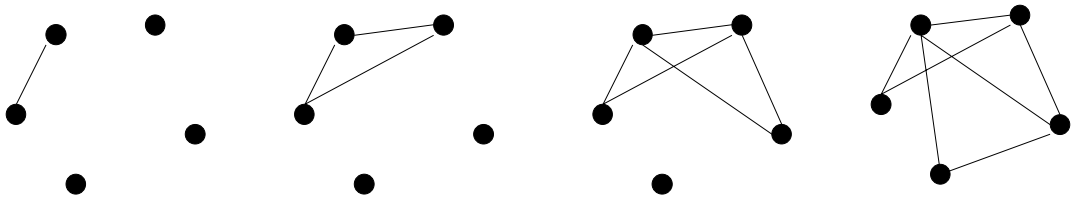
A Markov-hálózatokról és a Bayes-hálókról jó összefoglalókat találhatunk *Pearl* (1997) és *Jensen* (2007) könyveiben.

Visszatérve a feladatunkhoz, ilyen típusú speciális eloszláscsaládokból célszerű kiválasztani a valódit megközelítő együttes eloszlást. A következő fejezetben megmutatjuk, hogyan használtuk az általunk bevezetett *t*-cseresznye klaszterfákat abban a klasszifikációs feladatban, amelyben a csődbejutás előrejelzésére legalkalmasabb pénzügyi mutatókat kell kiválasztani.

A *t*-cseresznyeklaszterfa és a klasszifikációs feladat

A *t*-cseresznyefa egy gráfstruktúra, amit *Prékopa András* és *Bukszár József* vezetett be (Bukszár, Prékopa, 2001). Egy *t*-csresznyefa fölépítése látható a következő ábrán.

3. ábra



Ezt a fölépítést alkalmaztuk a *t*-cseresznyeklaszterfa megépítésében. A *t*-cseresznyeklaszterfa egy valószínűségi eloszlást definiál, amely háromváltozós és kétváltozós valószínűség-eloszlások szorzatainak hányadosaként áll elő. A *t*-cseresznyeklaszterfát úgy igyekszünk felépíteni, hogy az általa definiált valószínűségi eloszlás és a valódi eloszlás KL távolsága a lehető legkisebb legyen. Ha *n* valószínűségi változó együttes eloszlását közelítjük ily módon, akkor a *t*-cseresznyeklaszterfának *n*-2 darab, 3 valószínűségi változót tartalmazó klasztere és *n*-3 darab, 2 valószínűségi változót tartalmazó szeparáló halmaza lesz. A 4. ábrán láthatunk egy *t*-cseresznyeklaszterfát, és az általa definiált eloszlásfüggvényt.

A *t*-cseresznyeklaszterfákról, illetve annak általánosításáról, a *k*-adrendű *t*-cseresznyeklaszterfákról bővebben 2009-es (Szántai, Kovács, 2009/a és 2009/b) cikkeinkben lehet olvasni.

A klasszifikációs probléma, amelyre elméletünket alkalmazni óhajtjuk, a következő: Tekintsük a következő pénzügyi mutatókat:

Cash/Current Liabilities
Cash Flow/Current Liabilities
Cash Flow/Total Assets
Cash Flow/Total Debt
Cash/Net Sales
Cash/Total Assets
Current Assets/Current Liabilities
Current Assets/Net Sales
Current Assets/Total Assets
Current Liabilities/Equity
Equity/Fixed Assets
Equity/Net Sales
Inventory/Net Sales
Long Term Debt/Equity
MV of Equity/Book Value of Debt
Total Debt/Equity
Net Income/Total Assets
Net Quick Assets/Inventory
Net Sales/Total Assets
Operating Income/Total Assets
EBIT/Total Interest Payments
Quick Assets/Current Liabilities
Quick Assets/Net Sales
Quick Assets/Total Assets
Rate of Return to Common Stock
Retained Earnings/Total Assets
Return on Stock
Total Debt/Total Assets
Working Capital/Net Sales
Working Capital/Equity
Working Capital/Total Assets

Ezek közül azokat kellene kiválasztanunk, amelyek alapján, ha nem is pontosan, de nagy valószínűséggel meg tudjuk jósolni a klasszifikáló változó értékét, vagyis azt, hogy 1 év elteltével egy vállalat csődbe jut-e vagy sem.[4]

Az adatokat egy előadáshoz (Grabusts, 2004) tartozó adatbázisból nyertük. Sajnos ebben az adatbázisban csak 7 mutató szerepelt:

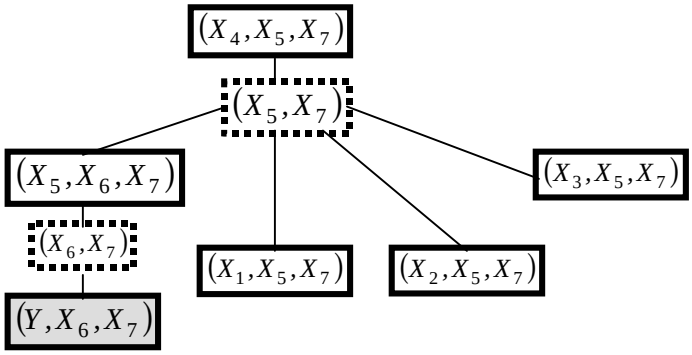
X_1 = Cash Flow/Liabilities;
X_2 = Profit or Loss/Total Assets;
X_3 = Liabilities/Total Assets;
X_4 = Working Capital/Total Assets;
X_5 = Current Assets/Current Liabilities;
X_6 = Current Assets/Total Liabilities;
X_7 = Current Assets/Total Assets.

Az általunk használt adatbázis (Rudorfer, 1995) 82 osztrák vállalat adatait tartalmazza, amelyből 16 hiányos adatokkal szerepel. A 66 teljes adattal rendelkezésre álló cégből véletlenszerűen kiválasztottunk 50-et, amit tanuló adatbázisnak tekintettünk. A kódolást a következő szempontok figyelembevételével vittük végbe: a mintában minden egyes X_i változót együtt tekintettünk az *Y* klasszifikáló változóval.

- Amennyiben növekvő számadatok ugyanazzal a klasszifikáló értékkel társultak, egy intervallumba soroltuk be.
- Amennyiben a klasszifikáló értékek szóródtak, az adott *X* változó értékkészletét közel egyforma gyakoriságú intervallumokra bontottuk.

Az intervallumokat növekvő sorrendbe számoztuk, így kódoltuk őket. Ezen adatok alapján megépítettük a következő *t*-cseresznyeklaszterfát:

4. ábra



[4] *Megjegyezzük, hogy e célra más eljárások is ismertek: Altman-féle Z mutató, logit analízis, Black –Sholes – Merton módszer, neurális hálók, fuzzy logika.*

Az ehhez tartozó eloszlás:

$$P_{cseri} = \frac{P(X_4, X_5, X_7)P(X_5, X_6, X_7)P(Y, X_6, X_7)P(X_1, X_5, X_7)P(X_2, X_5, X_7)P(X_3, X_5, X_7)}{[P(X_5, X_7)]^4 P(X_6, X_7)},$$

amely távolsága a mintához tartozó, empirikus eloszlástól KL=0,36.

A 4. ábrából kiderül, hogy az *Y*-ra legjobban ható mutatók az X_6 és X_7. Az *f* függvényt ezért célszerű úgy definiálni, hogy egy vállalat esetén azt az *Y* értéket vegye föl, amely a vállalatnál realizálódott X_6 és X_7 mutató értékek mellett *Y* értékeként a leggyakrabban fordult elő.

A tesztelési adatbázis 16 teljes vektorból és 16 hiányos vektorból állt. A 16 teljes vektor 93,75%-ában ismertük föl a helyes klasszifikáló értéket, míg a 16 hiányos vektor 81%-ában klasszifikáltunk helyesen.

Más módszerek felismerési arányai a következők voltak (Grabusts, 2004):

- a Z mutatót felhasználó módszerrel 82,5%-os,
- a 2-neurális hálóval 90,5%-os, míg az 1- és 3-neurális hálókkal 85,7%-os, a 4-neurális hálókkal 89%-os.

Megjegyezzük, hogy mi egy „nehéz” tesztelő mintát választottuk, mivel az tartalmazta az összes olyan esetet, ahol a többi módszer hibásan klasszifikált, illetve a hiányos adatokkal rendelkező vállalatok adatait is. Egy újabb kutatásnál ezek egy részét is érdemes belevenni a tanuló mintába. Módszerünk a többi módszer által helytelenül klasszifikált vállalatok egy részét is helyesen klasszifikálta.

Felhasznált irodalom

Besag, J. (1974): *Spatial Interaction and the Statistical Analysis of Lattice Systems.* Journal of the Royal Statistical Society, Series B, 36., 192–236.

Brush, S. (1967): *History of the Lenz-Ising mod*el. Rev. Mod. Phys. 39., 883–893.

Bukszár, J. – Prékopa, A. (2001): *Probability Bounds with Cherry Trees.* Mathematics of Operational Research, 26., 174–192.

Bukszár, J. – Szántai, T. (2002): *Probability Bounds Given by Hypercherry Trees.* Optimization Methods and Software, 17., 409–422

Grabusts, P. – Mendel (2004): *Analysing Bankruptcy Data with Neural Netwoorks.* Brno, 10th International Conference on Soft Computing, 111–117.

Jensen, F. V. – Nielsen, T. D. (2007): *Bayesian Networks and Decision Graphs, Information Science and Statistics.* New York, Springer, második kiadás.

Kovács, E. – Szántai, T. (2007): *On the Approximation of a Discrete Multivariate Distribution Using a the New Concept of t-Cherry Junction Three.* Luxemburg, Proceedings of IFIP/IIASA/GAMM, Workshop on Coping with Uncertainty 2007, IIASA. (Megjelenés alatt.)

Mc Callum, A. – Nigam, K. (1998): *A Comparison of Event Models for Naive Bayes Text Classification.* In: AAAI/ICML-98, Workshop on learning for Text Categorization, 41–48

Pearl, J. (1997): *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers, második kiadás.

Roweis, S. (2006): http://videolectures.net/mlss06tw_roweis_mlpgm/

Rudorfer, G. (1995): *Early bankruptcy detecting using neural networks.* New York, APL Quote Quad, ACM, 25/4: 171–178. Letöltés helye: http://godefroy.sdf-eu.org/apl95/ratios95.zip

Szántai, T. – Kovács, E. (2008): *Hypergraphs as a Means of Discovering the Dependence Structure of a Discrete Multivariate Probability Distribution.* International Conference on Applied Mathematical Programming and Modeling. (Megjelenés alatt.)