

Nenadzorovano učenje: odkrivanje vzorcev

Ljupčo Todorovski
Univerza v Ljubljani, Fakulteta za upravo

April 2018

Pregled predavanja

Razvrščanje v skupine (*clustering*)

- Problem razvrščanja v skupine
- Hierarhično razvrščanje v skupine
- Algoritem k-means

Povezovalna pravila (*association rules*)

- Algoritem APRIORI za odkrivanje pogostih množic postavk
- Od pogostih množic postavk do povezovalnih pravil

Nenadzorovano in nadzorovano učenje

Nadzorovano učenje

- Primeri $e \in \times_{i=1}^p D_i \times D_Y$,
 $e = (\mathbf{x}, y) = (x_1, x_2, \dots, x_p, y), x_i \in D_i, y \in D_Y$
- Rezultat učenja je napovedni model $y = m(\mathbf{x})$

Nenadzorovano učenje

- Primeri $e \in \times_{i=1}^p D_i$ oziroma $e = \mathbf{x} = (x_1, x_2, \dots, x_p), x_i \in D_i$
- Rezultat učenja so vzorci

Dva tipa vzorcev

Skupine primerov

- To so množice primerov, ki so si medsebojno podobni
- Algoritmi za razvrščanje v skupine (*clustering*)

Pogoste množice postavk (*frequent itemsets*)

- To so vrednosti spremenljivk, ki se pogosto pojavljajo skupaj
- Algoritem APRIORI za odkrivanje pogostih množic postavk
- Algoritem za odkrivanje povezovalnih pravil (*association rules*)

Problem razvrščanja v skupine

Vhoda

- Učna množica S primerov brez ciljne spremenljivke
- Želeno (pričakovano) število skupin k

Rezultat je razbitje množice S na k (disjunktnih) podmnožic S_i

$$S_i : \bigcup_{i=1}^k S_i = S, \forall i, j : S_i \cap S_j = \emptyset$$

Primeri v vsaki podmnožici S so medsebojno kar se da **podobni** (blizu).

Mera razdalje med primeri

$$d(e_1, e_2) = \sum_{i=1}^p \delta(x_{1i}, x_{2i})$$

- Pri Evklidski razdalji velja $d(e_1, e_2) = \sqrt{\sum_{i=1}^p \delta(x_{1i}, x_{2i})}$
- x_{ji} je vrednost spremenljivke X_i za primer e_j

δ je mera razdalje med vrednostmi $\delta : D_i \times D_i \rightarrow \mathbb{R}_0^+$

Če je X_i numerična, sta običajni izbiri

- Evklidska razdalja $\delta(v_1, v_2) = (v_1 - v_2)^2$
- Manhatnska razdalja $\delta(v_1, v_2) = |v_1 - v_2|$

Če je X_i diskretna, je običajna izbira $\delta(v_1, v_2) = I(v_1 \neq v_2)$

Iterativni algoritem za hierarhično razvrščanje

Začetni pogoj

Vsak primer e_i iz učne množice S je v svoji skupini S_i .

Iteracija

- V trenutnem naboru skupin izberimo dve **najbližji**: S_i in S_j
- Izbrani dve skupini zlijemo v eno $S_{ij} = S_i \cup S_j$

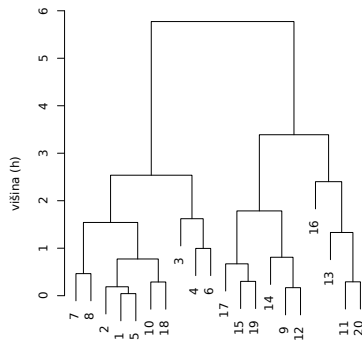
Ustavitveni pogoj

- V vsaki iteraciji število skupin zmanjšamo za 1
- Na koncu iteracije $|S| - 1$ dobimo eno skupino, ki je enaka S

Rezultat: dendrogram (hierarhija skupin)

Višina združevanja na osi y

Razdalja med skupinami primerov $\{2, 1, 5\}$ in $\{10, 18\}$ je nekaj manj kot 1.



Kako merimo razdaljo med skupinami (*linkage*)?

MAX ali *complete linkage*

$$D(S_1, S_2) = \max_{e_i \in S_1, e_j \in S_2} d(e_i, e_j)$$

MIN ali *single linkage*

$$D(S_i, S_j) = \min_{e_i \in S_i, e_j \in S_j} d(e_i, e_j)$$

MEAN ali *average linkage*

$$D(S_i, S_j) = \frac{1}{|S_1||S_2|} \sum_{e_1 \in S_1, e_2 \in S_2} d(e_1, e_2)$$

Ward-ova razdalja med skupinami (*linkage*)

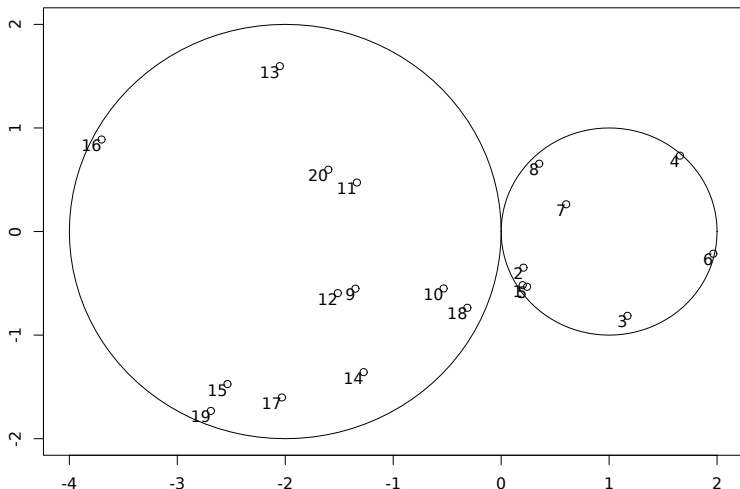
Na splošno lahko definiramo razdaljo nove skupine kot

$$\begin{aligned} D(S_1 \cup S_2, S_3) = & \alpha_1 D(S_1, S_3) + \alpha_2 D(S_2, S_3) \\ & + \beta D(S_1, S_2) + \gamma |D(S_1, S_3) - D(S_2, S_3)| \end{aligned}$$

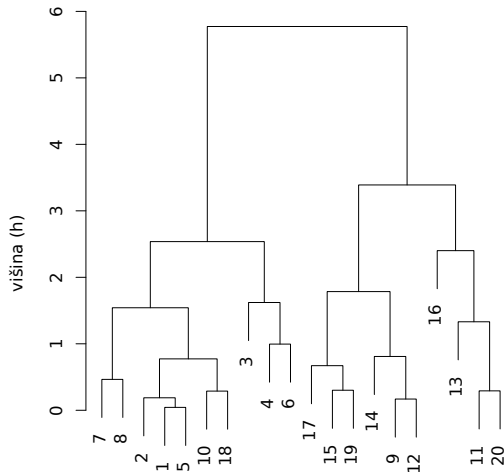
Za Ward-ovo razdaljo velja

$$\begin{aligned} \alpha_i = & = \frac{|S_i| + |S_3|}{|S_1| + |S_2| + |S_3|} \\ \beta = & - \frac{|S_3|}{|S_1| + |S_2| + |S_3|} \\ \gamma = & 0 \end{aligned}$$

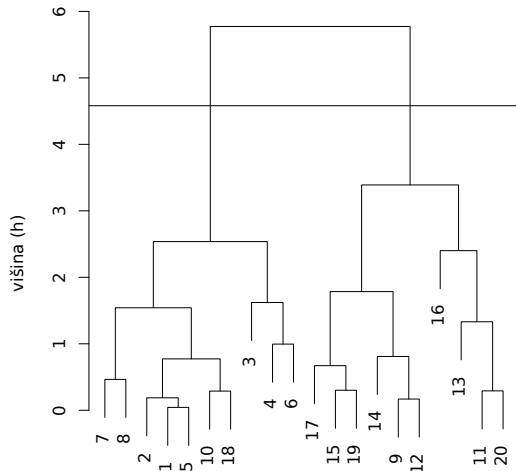
Primer podatkov za razvrščanje: dva kroga



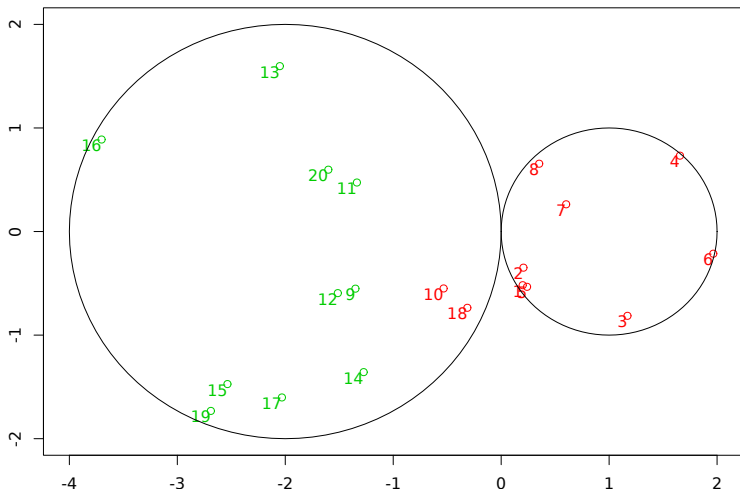
Primer dendrograma: MAX, *complete*



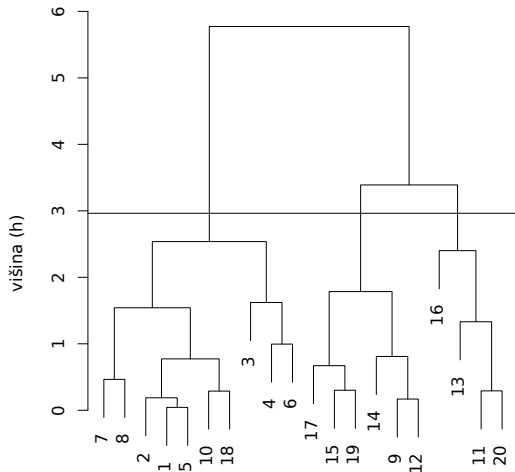
Od dendrograma do skupin (*complete*): dve skupini



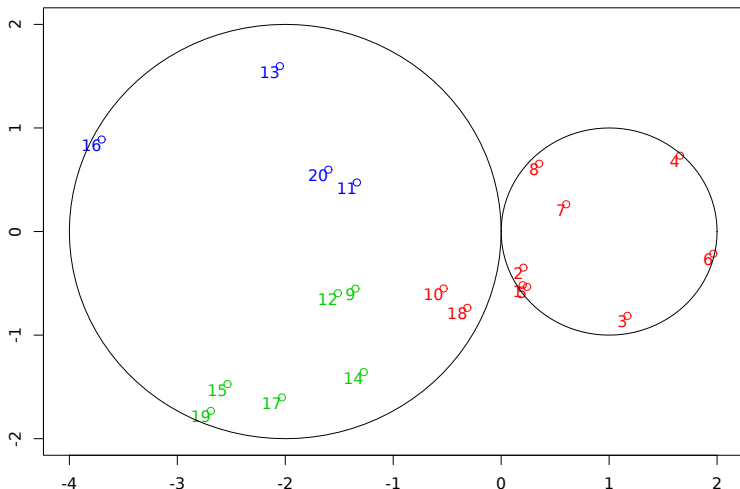
Dve skupini (*complete*): podatki



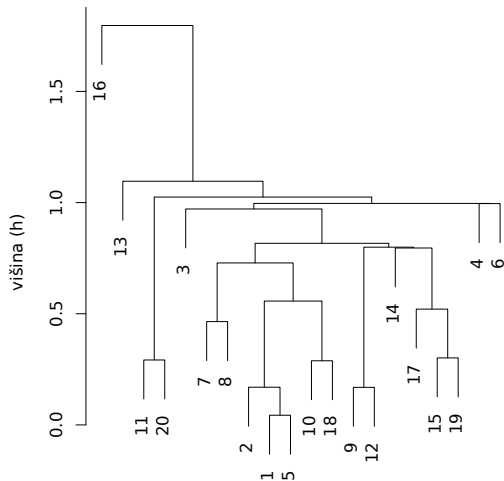
Od dendrograma do skupin (*complete*): tri skupine



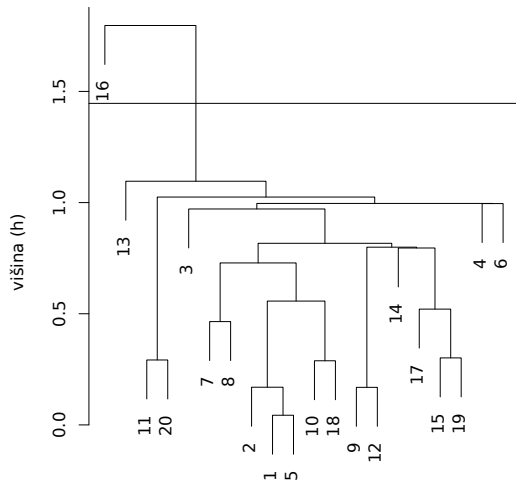
Tri skupine (*complete*): podatki



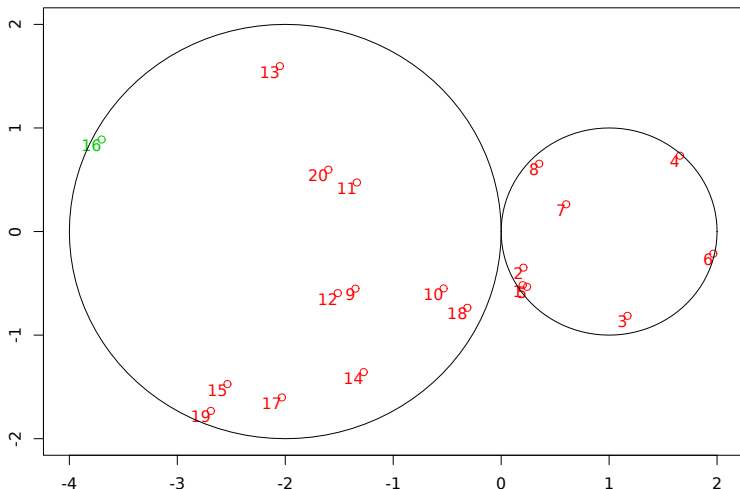
Primer dendrograma: MIN, *single*



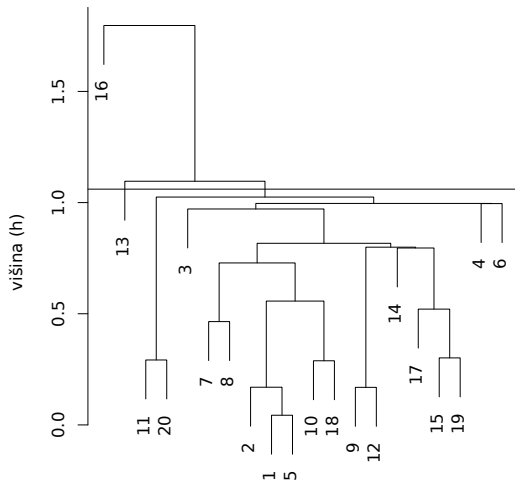
Od dendrograma do skupin (*single*): dve skupini



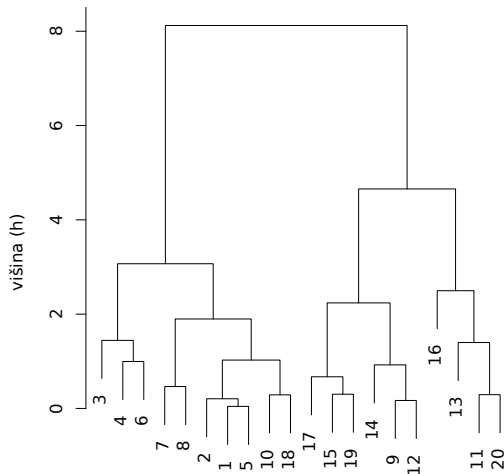
Dve skupini (*single*): podatki



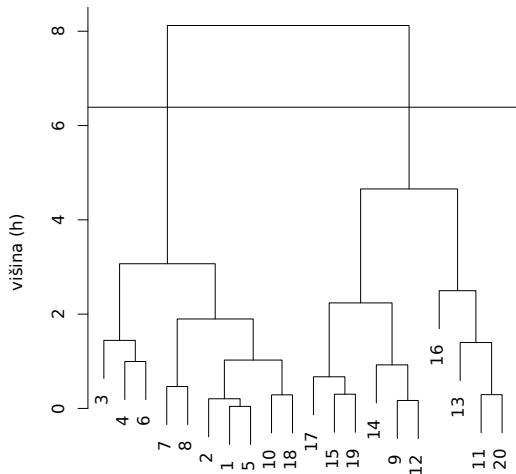
Od dendrograma do skupin (*single*): tri skupine



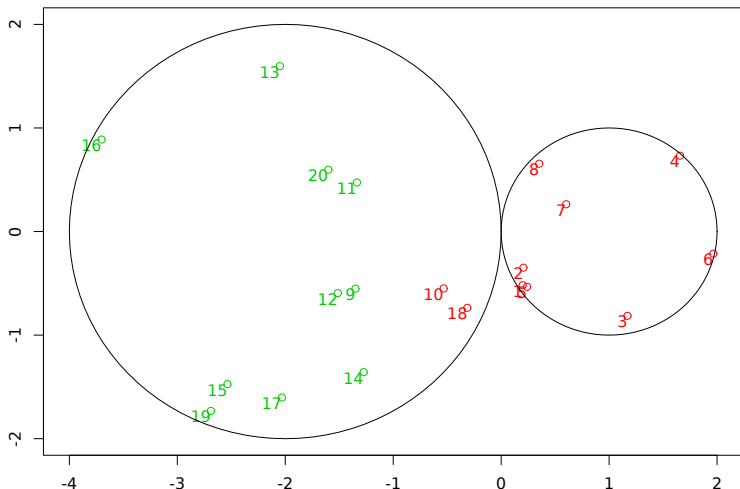
Primer dendrograma: *Ward*



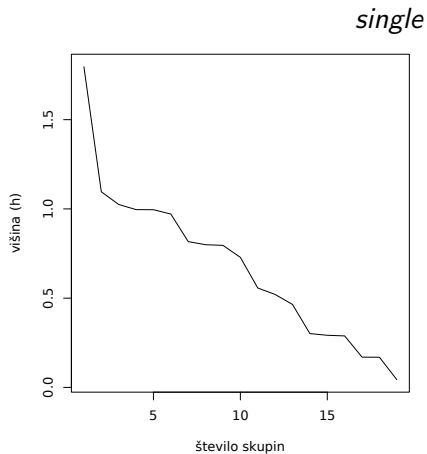
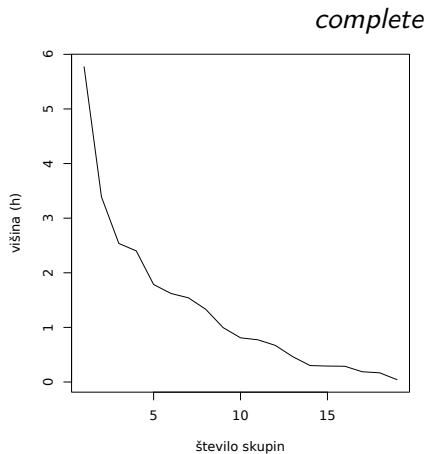
Od dendrograma do skupin (*Ward*): dve skupini



Dve skupini (*Ward*): podatki



Primerjava združitvenih višin



Razvrščanje v skupine kot optimizacijski problem

Najdi razbitje učne množice S na skupine S_1, S_2, \dots, S_k

$$\arg \min_{S_1, S_2, \dots, S_k} \sum_{i=1}^k |S_i| \text{Var}(S_i)$$

Minimiziramo torej varianco primerov znotraj skupin, hkrati (ker je celotna varianca v podatkih konstantna) maksimiramo varianco primerov iz različnih skupin.

Iterativni algoritem za optimizacijo (Lloyd)

Začetni pogoj

Primere iz S naključno razdelimo v k skupin, za vsako skupino S_i izračunamo centroid

$$\mu_i = \text{Centroid}(S_i) = \frac{1}{|S_i|} \sum_{e \in S_i} e$$

Iteracija iz dveh zaporednih korakov

- 1 Dodelitev (*assignment*) primerov skupinam
- 2 Posodabljanje (*update*) centroidov: izračun po zgornji formuli

Ustavitvena pogoja (lahko je izpolnjen le eden)

- V zadnji iteraciji se skupine niso spremenile
- Maksimalno število iteracij

Korak dodelitve (*assignment*)

Vsakemu primeru poiščemo najbližji centroid

In nato primer uvrstimo v skupino tega centroida:

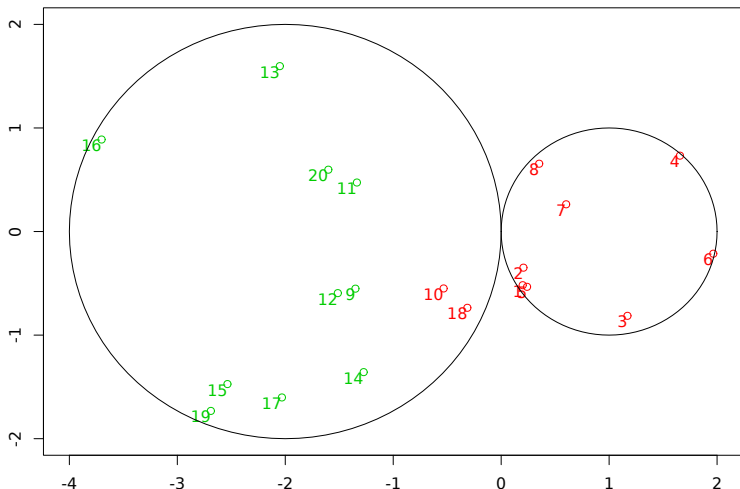
$$S_i = \{e : d(e, \mu_i) \leq d(e, \mu_j), \forall j : 1 \leq j \leq k\}$$

Če je primer enako oddaljen od več centroidov, naključno izberemo enega.

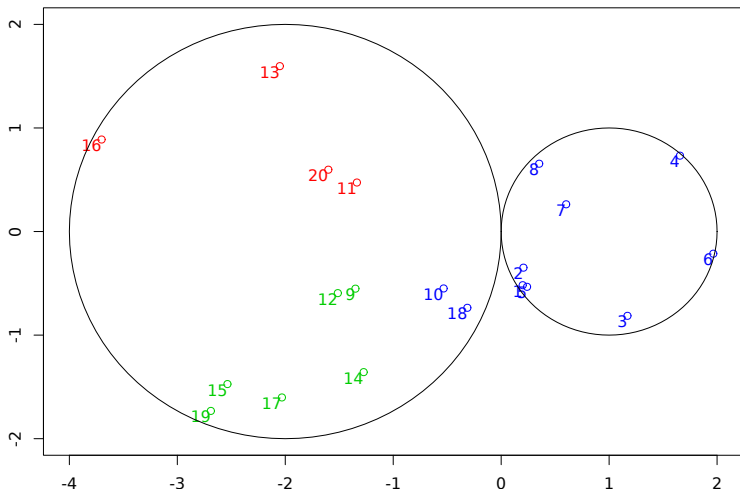
Običajna predpostavka za uporabo algoritam Lloyd

Evklidski prostor in razdalje: formula za centroid je prilagojena temu.

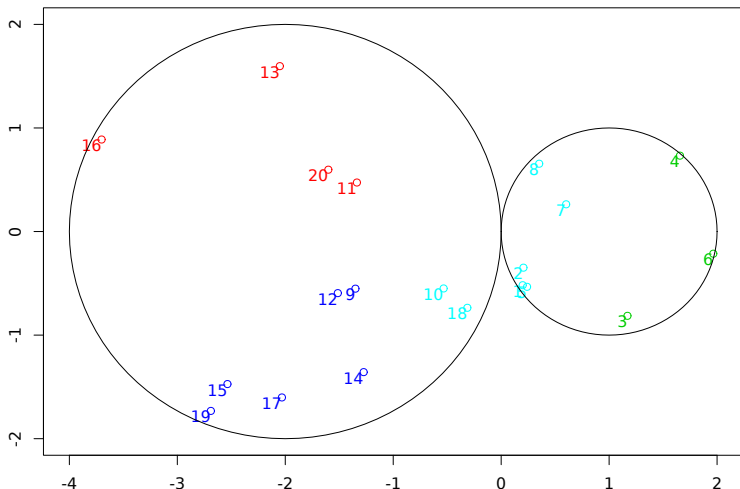
Dve skupini (k-means): podatki



Tri skupine (k-means): podatki



Štiri skupine (k-means): podatki



Osnovne definicije

Množica vseh postavk $I = \{i_1, i_2, \dots, i_p\}$

Predstavlja množico vseh postavk iz katerih so sestavljene transakcije.

Transakcijska baza podatkov T na množici postavk I

- Je množica transakcij $T = \{t_1, t_2, \dots, t_n\}$
- Vsaka transakcija je množica postavk $t_i \subseteq I$

Običajni primer

- Postavke so produkti $I = \{jabolka, kruh, mleko, pivo, plenice\}$
- Transakcije so nakupi

Transakcijska baza podatkov in učna množica

Transakcijska baza podatkov T na I je pravzaprav učna množica S :

- Vsaka postavka $i \in I$ ustreza Boolovi spremenljivki X_i iz S ,
 $D_i = \{0, 1\}$
- Vsaka transakcija $t \in T$ ustreza enemu primeru iz S
- Vrednost spremenljivke X_i za primer t je $I(i \in t)$

Primer transakcijske baze devetih transakcij v trgovini

	<i>transakcija</i>	<i>jabolka</i>	<i>kruh</i>	<i>mleko</i>	<i>pivo</i>	<i>plenice</i>
t_1	<i>jabolka, kruh, plenice</i>	1	1	0	0	1
t_2	<i>kruh, pivo</i>	0	1	0	1	0
t_3	<i>kruh, mleko</i>	0	1	1	0	0
t_4	<i>jabolka, kruh, pivo</i>	1	1	0	1	0
t_5	<i>jabolka, mleko</i>	1	0	1	0	0
t_6	<i>kruh, mleko</i>	0	1	1	0	0
t_7	<i>jabolka, mleko</i>	1	0	1	0	0
t_8	<i>jabolka, kruh, mleko, plenice</i>	1	1	1	0	1
t_9	<i>jabolka, kruh, mleko</i>	1	1	1	0	0

Algoritem *APRIORI*: vhodi in rezultat

Vhodi T, I in ϵ

- Transakcijska baza T na množici postavk I
- Parameter ϵ , ki določa minimalno pogostost množice postavk

Rezultat *FrequentItemsets*

Množica pogostih množic postavk, ki imajo pogostost $\geq \epsilon$.

Pogostost množice postavk s (tudi podpora, *support*) v T

$$\text{support}(s) = \frac{|\{t \in T : s \subseteq t\}|}{|T|}$$

Pogostost je ocena verjetnosti $p(s)$, da transakcija iz T vsebuje s .

Algoritem $APRIORI(T, I, \epsilon) = FrequentItemsets$

- L_k je množica pogostih postavk velikosti k
- C_k je množica kandidatov za pogoste postavke velikosti k

```

function  $APRIORI(T, I, \epsilon)$ 
1    $L_1 = \{pogoste\ postavke\ iz\ I\}$ 
2   for  $k = 2$  to  $|I|$  do
3        $C_k = Join(L_{k-1}, L_{k-1})$ 
4        $C_k = C_k \setminus \{t : (\exists s : s \subset t \wedge |s| = k - 1 \wedge s \notin L_{k-1})\}$ 
5       for  $t \in T$  do
6           for  $s \in \{s : s \in C_k \wedge s \subseteq t\}$  do
7                $support[s] = support[s] + 1/|T|$ 
8            $L_k = \{s : s \in C_k \wedge support[s] \geq \epsilon\}$ 
9           if  $L_k = \emptyset$  then break
10  return  $FrequentItemsets = \cup_k L_k$ 

```

Algoritem *APRIORI*: vrstica 1

Pogoste postavke $L_1 = \{\{i\} : i \in I \wedge \text{support}(\{i\}) \geq \epsilon\}$

- Za $\epsilon = 20\%$ je $L_1 = \{\{jabolka\}, \{kruh\}, \{mleko\}, \{pivo\}, \{plenice\}\}$
- Saj velja

<i>postavka</i> <i>i</i>	<i>jabolka</i>	<i>kruh</i>	<i>mleko</i>	<i>pivo</i>	<i>plenice</i>
<i>support(i)</i>	6/9	7/9	6/9	2/9	2/9

Algoritem *APRIORI*: vrstica 3 ($k = 2$)

$$C_k = \text{Join}(L_{k-1}, L_{k-1})$$

$$C_k = \{t \cup \{i\} : t \in L_{k-1} \wedge i \notin t \wedge \exists s \in L_{k-1} : i \in s\}$$

- Množicam pogostih postavk t iz L_{k-1} dodajamo po eno postavko i
- Postavka i ne sme biti element množice, kamor jo dodamo, $i \notin t$
- Postavka i mora biti element ene od množic L_{k-1}

Elementi $\text{Join}(L_1, L_1)$ za primer trgovine

$\{jabolka, kruh\}, \{jabolka, mleko\}, \{jabolka, pivo\}, \{jabolka, plenice\},$
 $\{kruh, mleko\}, \{kruh, pivo\}, \{kruh, plenice\}, \{mleko, pivo\},$
 $\{mleko, plenice\}, \{pivo, plenice\}$

Algoritem *APRIORI*: vrstica 4 ($k = 2$)

$$C_k = C_k \setminus \{t : (\exists s : s \subset t \wedge |s| = k - 1 \wedge s \notin L_{k-1})\}$$

Pobrišemo vse množice postavk iz C_k , ki imajo vsaj eno podmnožico velikosti $k - 1$, ki ni iz L_{k-1} . ZAKAJ?

Anti-monotonost pogostosti: $s \subset t \implies \text{support}(t) \leq \text{support}(s)$

Torej: če s ni pogosta množica postavk, potem tudi t ni.

Elementi C_2 za primer trgovine so vsi elementi $\text{Join}(L_1, L_1)$

$\{\text{jabolka}, \text{kruh}\}, \{\text{jabolka}, \text{mleko}\}, \{\text{jabolka}, \text{pivo}\}, \{\text{jabolka}, \text{plenice}\},$
 $\{\text{kruh}, \text{mleko}\}, \{\text{kruh}, \text{pivo}\}, \{\text{kruh}, \text{plenice}\}, \{\text{mleko}, \text{pivo}\},$
 $\{\text{mleko}, \text{plenice}\}, \{\text{pivo}, \text{plenice}\}$

Algoritem *APRIORI*: vrstice 5-8 ($k = 2$)

$t \in C_2$	transakcije iz T	$support(t)$
$\{jabolka, kruh\}$	t_1, t_4, t_8, t_9	4/9
$\{jabolka, mleko\}$	t_5, t_7, t_8, t_9	4/9
$\{jabolka, pivo\}$	t_4	1/9
$\{jabolka, plenice\}$	t_4, t_8	2/9
$\{kruh, mleko\}$	t_3, t_6, t_8, t_9	4/9
$\{kruh, pivo\}$	t_2, t_4	2/9
$\{kruh, plenice\}$	t_1, t_8	2/9
$\{mleko, pivo\}$		0/9
$\{mleko, plenice\}$	t_8	1/9
$\{pivo, plenice\}$		0/9

$$L_2 = \{\{jabolka, kruh\}, \{jabolka, mleko\}, \{jabolka, plenice\}, \{kruh, mleko\}, \{kruh, pivo\}, \{kruh, plenice\}\}$$

Algoritem *APRIORI*: vrstica 3 ($k = 3$)

Elementi $Join(L_2, L_2)$ za primer trgovine

- $\{jabolka, kruh, mleko\} = \{jabolka, kruh\} \cup \{mleko\}$
- $\{jabolka, kruh, plenice\} = \{jabolka, kruh\} \cup \{plenice\}$
- $\{jabolka, mleko, plenice\} = \{jabolka, mleko\} \cup \{plenice\}$
- $\{kruh, mleko, pivo\} = \{kruh, mleko\} \cup \{pivo\}$
- $\{kruh, mleko, plenice\} = \{kruh, mleko\} \cup \{plenice\}$
- $\{kruh, pivo, plenice\} = \{kruh, pivo\} \cup \{plenice\}$

Algoritem *APRIORI*: vrstica 4 ($k = 3$)

Elementi C_3 za primer trgovine

- $\{\text{jabolka, kruh, mleko}\}$ je, ker so vse podmnožice velikosti 2 v L_2
- $\{\text{jabolka, kruh, plenice}\}$ je; enako kot zgoraj
- $\{\text{jabolka, mleko, plenice}\}$ ni, ker $\{\text{mleko, plenice}\} \notin L_2$
- $\{\text{kruh, mleko, pivo}\}$ ni, ker $\{\text{mleko, pivo}\} \notin L_2$
- $\{\text{kruh, mleko, plenice}\}$ ni, ker $\{\text{mleko, plenice}\} \notin L_2$
- $\{\text{kruh, pivo, plenice}\}$ ni, ker $\{\text{pivo, plenice}\} \notin L_2$

$$C_3 = \{\{\text{jabolka, kruh, mleko}\}, \{\text{jabolka, kruh, plenice}\}\}$$

Algoritem *APRIORI*: vrstice 5-8 ($k = 3$)

$t \in C_3$	transakcije iz T	$support(t)$
$\{jabolka, kruh, mleko\}$	t_8, t_9	2/9
$\{jabolka, kruh, plenice\}$	t_1, t_8	2/9

$$L_3 = \{\{jabolka, kruh, mleko\}, \{jabolka, kruh, plenice\}\}$$

Algoritem *APRIORI*: vrstici 3 in 4 ($k = 4$)

Element $Join(L_3, L_3)$ za primer trgovine

$$\{jabolka, kruh, mleko, plenice\} = \{jabolka, kruh, mleko\} \cup \{plenice\}$$

Elementi C_4 za primer trgovine

$\{jabolka, kruh, mleko, plenice\}$ ni, ker $\{kruh, mleko, plenice\} \notin L_3$

$$C_4 = \emptyset, L_4 = \emptyset$$

Algoritem *APRIORI*: vrstica 9

$$\begin{aligned} \text{FrequentItemsets} &= L_1 \cup L_2 \cup L_3 \\ &= \{\{jabolka\}, \{kruh\}, \{mleko\}, \{pivo\}, \{plenice\}, \\ &\quad \{jabolka, kruh\}, \{jabolka, mleko\}, \{jabolka, plenice\}, \\ &\quad \{kruh, mleko\}, \{kruh, pivo\}, \{kruh, plenice\}, \\ &\quad \{jabolka, kruh, mleko\}, \{jabolka, kruh, plenice\}\} \end{aligned}$$

Oblika povezovalnih pravil

if B then H

- B je pogoj (telo, *body*)
- H je posledica (glava, *head*)
- B in H sta pogosti množici postavk, $B \cap H = \emptyset$

Zaupanje povezovalnega pravila r : if B then H

$$\text{confidence}(H, B) = \frac{\text{support}(H \cup B)}{\text{support}(B)}$$

Zaupanje je ocena pogojne verjetnosti $p(H|B)$, to je da transakcija iz T vsebuje H , pod pogojem, da vsebuje B .

Rudarjenje povezovalnih pravil: vhodi in rezultat

Vhodi T , I , ϵ in μ

- Transakcijska baza T na množici postavk I
- Parameter ϵ , ki določa minimalno pogostost množice postavk
- Parameter μ , ki določa minimalno zaupanje pravila

Rezultat *AssociationRules*

Množica povezovalnih pravil, ki imajo zaupanje $\geq \mu$.

Algoritem za rudarjenje povezovalnih pravil *MAR*

MAR = *MiningAssociationRules*

```
function MAR( $T, I, \epsilon, \mu$ )  
   $F = \text{APRIORI}(T, I, \epsilon)$   
   $R = \emptyset$   
  for  $f$  in  $F$  do  
    for  $(H, B)$  in  $\{(H, B) : H \subseteq f \wedge B \subseteq f \wedge H \cap B = \emptyset\}$  do  
      if  $\text{confidence}(H, B) \geq \mu$  then  
         $R = R \cup \{r : \text{if } H \text{ then } B\}$   
  return  $\text{AssociationRules} = R$ 
```


Algoritem *MAR*: $f = \{jabolka, kruh, plenice\}$

povezovalno pravilo r	$confidence(r)$
if $\{jabolka\}$ then $\{kruh, plenice\}$	$2/6 = 33\%$
if $\{kruh\}$ then $\{jabolka, plenice\}$	$2/7 = 29\%$
if $\{plenice\}$ then $\{jabolka, kruh\}$	$2/2 = 100\%$
if $\{jabolka, kruh\}$ then $\{plenice\}$	$2/4 = 50\%$
if $\{jabolka, plenice\}$ then $\{kruh\}$	$2/2 = 100\%$
if $\{kruh, plenice\}$ then $\{jabolka\}$	$2/2 = 100\%$

Vzpon povezovalnih pravil

Vzpon (*lift*) povezovalnega pravila r : if B then H

$$\text{lift}(H, B) = \frac{\text{support}(H \cup B)}{\text{support}(H) \cdot \text{support}(B)}$$

Vzpon je mera (ne)odvisnosti med dogodkoma "transakcija vključuje H " in "transakcija vključuje B "; vzpon 1 nakazuje neodvisnost, višje vrednosti pa odvisnost.

povezovalno pravilo r	$\text{confidence}(r)$	$\text{lift}(r)$
if $\{plenice\}$ then $\{jabolka, kruh\}$	$2/2 = 100\%$	$9/4 = 2.25$
if $\{jabolka, plenice\}$ then $\{kruh\}$	$2/2 = 100\%$	$9/7 = 1.29$
if $\{kruh, plenice\}$ then $\{jabolka\}$	$2/2 = 100\%$	$9/6 = 1.50$

Povezovalna in odločitvena pravila: podobnost in razlike

Podobnost

Oboje enake oblike: if *Pogoj* then *Posledica*.

Razlike

- Odločitvena pravila imajo posledico, ki omogoča napoved $Y = v$
- Množica (urejena ali ne) odločitvenih pravil je en **napovedni model**
- Povezovalna pravila v posledici se lahko sklicujejo na poljubni X
- Povezovalna pravila tolmačimo vsakega posebej kot **vzorec**, ki pojasni podatke oz. razkrije prej neznane podatkovne povezave
- Obstajajo algoritmi za učenje odločitvenih pravil iz povezovalnih

Znani algoritmi in implementacije

Hierarhično razvrščanje v skupine (Florek in ost 1951)

Vgrajena funkcija v R.

Algoritem k-means (Lloyd 1957)

Vgrajena funkcija v R.

APRIORI (Agrawal in ost 1993), a pravzaprav GUHA (Hájek 1966)

Pogoste množice postavk in rudarjenje povezovalnih pravil, tudi implementacija v R.