Eksploracja danych

reguły asocjacyjne

Piotr Lipiński

Reguły asocjacyjne

- Badania dotyczące reguł asocjacyjnych zostały rozpoczęte na potrzeby analizy transakcji dokonywanych przez klientów w supermarketach (tzw. analiza koszyka zakupów).
- Rozpatrujemy pewien zbiór przedmiotów $\mathbf{I} = \{I_1, I_2, \ldots, I_d\}$ i pewien zbiór transakcji $\mathbf{T} = \{T_1, T_2, \ldots, T_N\}$, gdzie każda transakcja T_i to pewien podzbiór zbioru przedmiotów **I**.
- Co jest interesujące ?
  - jak często były kupowane poszczególne przedmioty
    - dokładniej: w ilu procentach transakcji występowały poszczególne przedmioty
      - odpowiedzi dostarcza prosta statystyka
  - jakie przedmioty były kupowane razem
    - dokładniej: jakie są częste zbiory przedmiotów występujące w transakcjach
      - odpowiedź wymaga nietrywialnej analizy danych
  - kupno jakiego przedmiotu pociąga za sobą kupno przedmiotu X

Reguły asocjacyjne

- Przykład:

1	plums, lettuce, tomatoes
2	celery, confectionery
3	confectionery
4	apples, carrots, tomatoes, potatoes, confectionery
5	apples, oranges, lettuce, tomatoes, confectionery
6	peaches, oranges, celety, potatoes
7	beans, lettuce, tomatoes
8	oranges, lettuce, carrots, tomatoes, confectionery
9	apples, bananas, plums, carrots, tomatoes, onions, confectionery
10	apples, potatoes

Reguły asocjacyjne

- Dla zbioru przedmiotów $X \subseteq \mathbf{I}$ określamy
  - support X (nośnik, wsparcie)

$$supp(X) = |\{T \in \mathbf{T}: X \subseteq T\}| / N.$$

- Częste zbiory przedmiotów
  - Zbiór przedmiotów $X \subseteq \mathbf{I}$ nazywamy częstym, jeśli występuje w dużej liczbie transakcji, czyli $supp(X) > \alpha$, dla pewnego ustalonego progu $\alpha > 0$.
  - Fakt 1: Jeśli zbiór przedmiotów $X \subseteq \mathbf{I}$ jest częsty, to każdy jego podzbiór $Y \subseteq X$ też jest częsty.
  - Fakt 2: Jeśli zbiór przedmiotów $X \subseteq \mathbf{I}$ zawiera podzbiór $Y \subseteq X$, który nie jest częsty, to sam zbiór X też nie jest częsty.
  - W związku z tym konstrukcję wszystkich zbiorów częstych można rozpocząć od zbiorów częstych jednoelementowych, a następnie łączyć je kolejno w zbiory częste dwuelementowe, trzyelementowe, itd.

Reguły asocjacyjne

- Reguła asocjacyjne $A \rightarrow C$ to para zbiorów przedmiotów $A \subseteq \mathbf{I}$ i $C \subseteq \mathbf{I}$.
  - Interpretacja: Jeśli w transakcji T znalazły się przedmioty ze zbioru A, tzn. $A \subseteq T$, to z dużym prawdopodobieństwem w transakcji T znalazły się przedmioty ze zbioru B, tzn. $B \subseteq T$.
- Dla reguły asocjacyjnej $A \rightarrow C$ określamy
  - support $A \rightarrow C$ (nośnik, wsparcie)

    $$supp(A \rightarrow C) = supp(A \cup C),$$
  - confidence $A \rightarrow C$ (wiarygodność, zaufanie)

    $$conf(A \rightarrow C) = supp(A \cup C) / supp(A).$$
- Jeśli $supp(A \rightarrow C)$ jest wystarczająco wysokie, a zarejestrowane transakcje reprezentują próbę losową z tego samego rozkładu co przyszłe transakcje, to $conf(A \rightarrow C)$ jest dość dobrym estymatorem prawdopodobieństwa, że dowolna przyszła transakcja zawierająca A będzie zawierać także B.
- Celem jest wyznaczenie zbioru wszystkich reguł asocjacyjnych o wystarczająco wysokim wsparciu i jak najwyższej wiarygodności.

Algorytm apriori

- IDEA ALGORYTMU:
  - jako parametr algorytmu określamy minimalne wsparcie min-supp,
  - zbiory przedmiotów i reguły asocjacyjne o wsparciu poniżej minimalnego uznajemy za nieinteresujące (niewiarygodne),
  - skoro $supp(A \rightarrow C) = supp(A \cup C)$, to interesujące są tylko takie reguły asocjacyjne $A \rightarrow C$, dla których $A \cup C$ jest zbiorem częstym (o minimalnym wsparciu min-supp),
  - generujemy wszystkie zbiory częste X,
  - dla każdego zbioru częstego X, rozpatrujemy wszystkie reguły asocjacyjne $A \rightarrow C$, dla których $A \cup C = X$, tzn. generujemy rozbicia zbioru X na podzbiory A i C,
  - ograniczamy się do rozbicia zbioru X na ROZŁĄCZNE podzbiory A i C.

Algorytm apriori

- ALGORYTM A PRIORI:

$$
\begin{array}{l}
k = 1;\ L_k = \text{zbiór jednoelementowych zbiorów częstych;} \\
k = 2; \\
\textbf{while } L_{k-1} \neq \varnothing \textbf{ do} \\
\quad C_k = \{\{x_1, x_2, \ldots, x_{k-2}, x_{k-1}, x_k\} : \\
\qquad\qquad \{x_1, x_2, \ldots, x_{k-2}, x_{k-1}\} \in L_{k-1} \wedge \\
\qquad\qquad \{x_1, x_2, \ldots, x_{k-2}, x_k\} \in L_{k-1}\}; \\
\quad \textbf{for each } t \in \mathbf{T} \textbf{ do} \\
\qquad \textbf{for each } c \in C_k \wedge c \subseteq t \textbf{ do} \\
\qquad\quad \mathbf{c.count{+}{+};} \\
\quad L_k = \{c \in C_k : c.count \geq \text{min-supp}\}; \\
\textbf{return } L_1 \cup L_2 \cup \ldots \cup L_k;
\end{array}
$$

Przykład

- Niech min-confidence = 0.75.
- Niech X = {lettuce, tomatoes}.
- Łatwo policzyć, że support(X) = 0.4.

- S_0 = {{lettuce, tomatoes} → ∅ }.
- Rozpatrujemy:
  - confidence({lettuce}→ {tomatoes}) = 0.4 / 0.4
  - confidence({tomatoes} → {lettuce}) = 0.4 / 0.6

- S_1 = {{lettuce}→ {tomatoes}}
- Rozpatrujemy:
  - confidence(∅ → {lettuce, tomatoes}) = 0.4

- S_2 = ∅

- Ostatecznie, wiarygodne reguły uzyskane ze zbioru częstego X, to:
  - {lettuce, tomatoes} → ∅
  - {lettuce}→ {tomatoes}

Algorytm apriori

•Generowanie reguł asocjacyjnych ze zbioru częstego X oparte jest na rozbijaniu tego zbioru w różny sposób na dwa rozłączne podzbiory Y i Z, takie że $X = Y \cup Z$, i rozpatrywaniu reguł $Y \to Z$. Zwracane są tylko wiarygodne reguły $Y \to Z$, czyli takie których confidence($Y \to Z$) >= min_confidence.

•**Spostrzeżenie 1**: confidence($A \to B \cup C$) <= confidence($A \cup B \to C$)
•Dowód: wprost z definicji confidence, bo support(A) >= support($A \cup B$).

•**Wniosek**: Jeśli reguła $A \to B \cup C$ jest wiarygodna, to reguła $A \cup B \to C$ tym bardziej jest wiarygodna.

•**Spostrzeżenie 2**: confidence($X \to \emptyset$) = 1 i żadna inna reguła powstała z X nie ma większego confidence.

•**Spostrzeżenie 3**: confidence($\emptyset \to X$) = support(X) i żadna inna reguła powstała z X nie ma mniejszego confidence.

•**Konstruowanie reguł:**
• Zaczynamy od reguły $X \to \emptyset$. Niech k = 0, zaś zbiór reguł wiarygodnych $S_k = \{X \to \emptyset\}$.
• Będziemy próbować przesuwać przedmioty z lewej strony na prawą. Niech $S_{k+1} = \emptyset$. Dla każdej reguły $A \to B$ z S_k, sprawdzamy wszystkie reguły postaci $A \setminus \{x\} \to B \cup \{x\}$ (dla każdego $x \in A$). Jeżeli reguła jest wiarygodna, to dodajemy ją do S_{k+1}. W tym kroku sprawdzamy więc $|S_k|(|X| - k)$ reguł.
• Niech k = k + 1. Jeśli S_k jest niepusty, to wracamy do kroku 1.
• Zwracamy wszystkie reguły ze zbiorów $S_0, S_1, \ldots, S_k$.

Które reguły są interesujące?

- Często otrzymujemy wiele reguł, z których nie wszystkie są interesujące.
- Często bada się różnicę między wsparciem reguły, a iloczynem wsparcia lewej i prawej strony reguły.
- Jeśli wsparcie reguły jest (w przybliżeniu) równy iloczynowi wsparcia lewej i prawej strony, to mówimy, że lewa i prawa strona są niezależne. Takie reguły są nieinteresujące, niezależnie od ich wiarygodności.
  - Dlaczego?
- Powstała więc idea stworzenia uniwersalnych miar oceniających jakość reguł asocjacyjnych.

Które reguły są interesujące?

- Dla reguły asocjacyjnej $A \to C$ określamy dodatkowe miary
  - lift $A \to C$

$$\text{lift}(A \to C) = \text{confidence}(A \to C) / \text{supp}(C)$$

  - leverage $A \to C$

$$\text{leverage}(A \to C) = \text{supp}(A \to C) - \text{supp}(A)\ \text{supp}(C).$$

Które reguły są interesujące?

- Przykład:

{pomidory}→{sałata}
support({sałata}) = 0.4
confidence({pomidory} →{sałata}) = 0.67
lift({pomidory}→{sałata}) = 0.67 / 0.4 = 1.675

{pomidory}→{słodycze}
support({słodycze}) = 0.6
confidence({pomidory}→{słodycze}) = 0.67
lift({pomidory}→{słodycze}) = 0.67 / 0.6 = 1.117

WNIOSEK: Pomidory mają większy wpływ na sałatę niż na słodycze.

Które reguły są interesujące?

- Przykład:

{marchew}→{pomidory}, confidence = 1, lift = 1.667
{sałata}→{pomidory}, confidence = 1, lift = 1.667

UWAGA: Jednak d*ruga reguła może być bardziej interesująca, bo dotyczy większej liczby transakcji.*

support({marchew}→{pomidory}) = 0.3
support({marchew}) = 0.3
support({pomidory}) = 0.6
leverage({marchew} →{pomidory})=0.3 – 0.3 * 0.6 = 0.12

support({sałata}→{pomidory}) = 0.4
support({sałata}) = 0.4
leverage({sałata} →{pomidory})=0.4 – 0.4 * 0.6 = 0.16

Reguły podobne

- Zobaczmy dwie wykryte reguły
  apples → confectionery, tomatoes
  apples, tomatoes → confectionery
  obie one mają takie samo wsparcie, równe 0.3, ale różną wiarygodność.

- Obie reguły są partycją tego samego częstego zbioru przedmiotów
  {apples, confectionery, tomatoes},
  więc np. przy ustawianiu produktów na sklepowych półkach przydatność obu reguł jest taka sama.

- Jak oceniać jakość wykrytych zbiorów przedmiotów?

Efektywność algorytmu

- Algorytm APRIORI jest dosyć efektywny w przypadku danych rzadkich (sparse data), kiedy każdy przedmiot występuje w niewielkiej liczbie transakcji.

- Istnieją modyfikacje tego algorytmu dla innych sytuacji (praca Bayardo 1997), opierają się na różnych sposobach konstrukcji częstych zbiorów przedmiotów.

- Istnieją też modyfikacje algorytmu konstruujące reguły asocjacyjne bez uprzedniego konstruowania częstych zbiorów przedmiotów.

Algorytm PrefixSpan

- Dotychczas interesowały nas transakcje będące zbiorami przedmiotów. Tak postrzegane transakcje, jako zbiory, nie zawierały informacji o kolejności występowania przedmiotów.

- Interesującym rozszerzeniem tradycyjnego podejścia jest określenie transakcji jako ciągów przedmiotów, co umożliwi określenie kolejności występowania przedmiotów.

- Potencjalne zastosowanie: śledzenie aktywności użytkowników w portalu internetowym (przedmiot = strona portalu, transakcja = ciąg odwiedzonych stron portalu).

Ciągi częste

- Przykład:

  I = {a, b, c, d}

  D = { <1, <abc>>, <2, <ac>>, <3, <bd>>, <4, <acd>> }

  min-supp = 3

  Ciągi częste to {<a>, <c>, <ac>}

- Interpretacja:
  - Portal zawiera cztery strony internetowe a, b, c, d.
  - Zarejestrowano następujące wizyty użytkowników: abc, ac, bd, acd.
  - Wówczas często odwiedzane strony portalu to a i c, a często wykonywana ścieżka w portalu to przejście ze strony a (niekoniecznie bezpośrednio) do strony c.

Ciągi częste

- Zamiast częstych zbiorów i budowanych na nich reguł asocjacyjnych zajmijmy się przypadkiem częstych ciągów (w których ważna jest kolejność elementów)
- Czym się różni to podejście od częstych zbiorów?
- Czy można zastosować algorytm a priori do konstrukcji ciągów częstych?
- Jednym z algorytmów do wyszukiwania ciągów częstych jest algorytm PrefixSpan.

Ciągi częste

- I = {a, b, c, d, e}
- D = { <1, <abc>>, <2, <acbed>>, <3, <bd>>, <4, <bcdad>>, <5, <becabc>> }
- min-supp = 3

- Krok 1: Znajdź zbiór 1-elementowych ciągów częstych.
  - Wystarczy jednokrotnie przejrzeć zbiór transakcji D i policzyć liczby wystąpień poszczególnych przedmiotów z I, a następnie wybrać te przedmioty, które występują co najmniej min-supp razy.
  - L_1 = {<a>, <b>, <c>, <d>}
- Krok 2: Podziel przestrzeń obliczeń.
  - Dłuższe ciągi częste muszą rozpoczynać się prefiksem będącym jednym ze znalezionych 1-elementowych ciągów częstych.
  - Szukamy ciągów częstych o prefiksie <a>, prefiksie <b>, prefiksie <c> i prefiksie <d>.
- Krok 3: Znajdź kolejne zbiory ciągów częstych.
  - Kolejne ciągi częste można wyznaczyć rzutując zbiór transakcji D na poszczególne 1-elementowe prefiksy z L_1.
  - D|<a> = { <1, <bc>>, <2, <cbed>>, <4, <d>>, <5, <bc>> }

Ciągi częste

- Krok 3a: Znajdź zbiór ciągów częstych o prefiksie <a>
  - Rzutujemy zbiór transakcji D na prefiks <a> uzyskując:

    D|<a> = { <1, <bc>>, <2, <cbed>>, <4, <d>>, <5, <bc>> }.
  - Wyznaczamy 1-elementowe ciągi częste w D|<a> uzyskując:

    <b> i <c>.
  - Zatem wszystkie 2-elementowe ciągi częste w D o prefiksie <a> to:

    <ab> i <ac>.
  - Dłuższe ciągi częste w D o prefiksie <a> muszą mieć więc prefiks <ab> lub <ac>, znajdujemy je rekurencyjnie

Ciągi częste

- Krok 3ab: Znajdź zbiór ciągów częstych o prefiksie <ab>
  - Rzutujemy zbiór transakcji D na prefiks <ab> uzyskując:

    D|<ab> = { <1, <c>>, <2, <ed>>, <5, <c>> }.
  - Wyznaczamy 1-elementowe ciągi częste w D|<ab> uzyskując zbiór pusty (nie ma żadnych ciągów częstych w D|<ab>).
  - Zatem nie ma żadnych 3-elementowych ciągów częste w D o prefiksie <ab>.

- Krok 3ac: Znajdź zbiór ciągów częstych o prefiksie <ac>
  - Rzutujemy zbiór transakcji D na prefiks <ab> uzyskując:

    D|<ac> = { <1, <ε>>, <2, <bed>>, <5, <ε>> }.
  - Wyznaczamy 1-elementowe ciągi częste w D|<ac> uzyskując zbiór pusty (nie ma żadnych ciągów częstych w D|<ac>).
  - Zatem nie ma żadnych 3-elementowych ciągów częste w D o prefiksie <ac>.

- Zatem wszystkie ciągi częste w D o prefiksie <a> to: <a>, <ab> i <ac>.

Ciągi częste

- Analogicznie wyznaczamy wszystkie ciągi częste w D o prefiksie <b>, <c> i <d>.
- Ostatecznie otrzymujemy:
  - <a>, <ab>, <ac>,
  - <b>, <bc>, <bd>,
  - <c>,
  - <d>.

PrefixSpan

- Algorytm PrefixSpan

```
PrefixSpan(p, D|p)
   F = {1-elementowe ciągi częste z D|p}
   R = ∅
   for each f ∈ F do
      q = p · f
      R = R ∪ {q}
      wyznacz D|q
      R = R ∪ PrefixSpan(q, D|q)
   return R
```