**WYŻSZA SZKOŁA INFORMATYKI**

**I ZARZĄDZANIA**

Z SIEDZIBĄ W RZESZOWIE

Sprawozdanie

Sztuczna inteligencja

***Regułowe systemy decyzyjne LERS***

Prowadzący: dr inż. Mariusz Wrzesień Wykonawca: Sylwia Babiarz 46741

6IID – GAK, SL04

Rzeszów 2014

# Opis problemu

Celem laboratorium jest utworzenie zestawu reguł decyzyjnych, dla danych umiejscowionych w tablicy decyzji, przy pomocy algorytmu LEM2.

# Badane zbiory danych

Dane, jakie należy przetworzyć, dotyczą owoców. Są one przedstawione za pomocą tablicy decyzji, której skład to sześć kolumn i dwadzieścia jeden wierszy. Wszystkie wiersze to przypadki, jakie są rozpatrywane. Kolumna pierwsza jest indeksowana atrybutem nominalnym o nazwie „color”. Zawiera cztery wartości atrybutów, tj. green, red, maroon, yellow. Kolumna druga indeksowana jest atrybutem porządkowym „size”. Posiada trzy wartości, medium, small, big. Kolejna kolumna ma nazwę atrybutu nominalnego „shape” i dysponuje dwoma wartościami, round i elongated. Kolumna numer 4 o atrybucie nominalnym ,,taste” posiada także dwie wartości sour oraz sweet. Przedostatnia z kolumn indeksowana jest atrybutem liczbowym „weight” wraz z pięcioma wartościami, 0.1000, 0.2000, 0.3000, 0.4000, 0.5000. Ostatnia kolumna ma nazwę atrybutu „fruit” i jest to kolumna decyzji. Składa się z sześciu klas: Lemon, Apple, Grapefruit, Grape, Banana, Cherry. Kolumna ,,LP” nie wchodzi w ciało tablicy decyzji, ukazuje tylko ile jest rozpatrywanych przypadków.

Tabela 1 Tablica decyzji z danymi o owocach

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***LP*** | ***COLOR*** | ***SIZE*** | ***SHAPE*** | ***TASTE*** | ***WEIGHT*** | ***FRUIT*** |
| **1** | green | medium | round | sour | 0.1000 | Lemon |
| **2** | red | medium | round | sweet | 0.3000 | Apple |
| **3** | green | medium | round | sour | 0.1000 | Lemon |
| **4** | red | small | round | sweet | 0.1000 | Cherry |
| **5** | green | medium | round | sour | 0.3000 | Apple |
| **6** | red | medium | round | sweet | 0.4000 | Apple |
| **7** | green | big | round | sour | 0.5000 | Grapefruit |
| **8** | green | small | round | sweet | 0.1000 | Grape |
| **9** | red | medium | round | sour | 0.4000 | Apple |
| **10** | maroon | small | round | sweet | 0.2000 | Grape |
| **11** | green | big | elongated | sweet | 0.3000 | Banana |
| **12** | yellow | big | round | sweet | 0.5000 | Grapefruit |
| **13** | green | small | round | sour | 0.1000 | Grape |
| **14** | green | medium | elongated | sweet | 0.2000 | Banana |
| **15** | yellow | medium | elongated | sweet | 0.3000 | Banana |
| **16** | yellow | big | elongated | sweet | 0.3000 | Banana |
| **17** | green | big | round | sweet | 0.4000 | Apple |
| **18** | green | medium | round | sour | 0.1000 | Lemon |
| **19** | yellow | medium | round | sour | 0.2000 | Lemon |
| **20** | yellow | small | round | sour | 0.1000 | Lemon |
| **21** | red | big | round | sweet | 0.5000 | Apple |

# Obliczenia

Na przykładzie jednej z klas tablicy decyzji o owocach można opisać działanie algorytmu LEM2. Do generowania reguł został wykorzystany program Microsoft Excel z funkcją filtrowania. Poniższy opis przedstawia sposób generowania reguł dla klasy Lemon.

* Pierwszym krokiem jest wyznaczenie zbioru B, który zawiera wszystkie przypadki klasy Lemon:

***B= {1, 3, 18, 19, 20}.***

* Następnie zostaje wyznaczony zbiór T(G) wszystkich możliwych par atrybut– wartość spośród przypadków ze zbioru B:

***T(G) = {(COLOR, green), (COLOR, yellow), (SIZE, medium), (SIZE, small),***

***(SHAPE, round), (TASTE, sour), (WEIGHT, 0.1), (WEIGHT, 0.2)}.***

* Wstępnie założono, że zbiór ***B = G = {1, 3, 18, 19, 20} oraz n = Ø.***
* Następnie wypisano wszystkie przypadki z tablicy decyzji, które należą do danej kombinacji atrybut – wartość. Przypadki dotyczące klasy Lemon zostały wyszczególnione kolorem zielonym.

***[(Color, green)] = {1, 3, 18, 5, 7, 8, 11, 13, 14, 17}***

***[(Color, yellow)] = {19, 20, 12, 15, 16}***

***[(Size, medium)] = {1, 3, 18, 19, 2, 5, 6, 9, 14, 15}***

***[(Size, small)] = { 20, 4, 8, 10, 13}***

***[(Shape, round)] = { 1, 3, 18, 19, 20,2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 17, 21}***

***[(Taste, sour)] = { 1, 3, 18, 19, 20, 5, 7, 9, 13}***

***[(Weight,0.1)] = {1, 3, 18, 20, 4, 8, 13}***

***[(Weight, 0.2)] = { 19, 10,14}***

* Spośród wszystkich par zostają wybrane te, które mają największą część wspólną ze zbiorem G oraz mają jak największą moc, czyli pokrywają jak najmniejszą liczbę przypadków z innych klas.

* Najwięcej przypadków z klasy Lemon zawierają kombinacje [(Shape, round)] oraz [(Taste, sour)] (po cztery przypadki). Algorytm w pierwszej kolejności wybiera zestawienie ***[(Taste, sour)]***, ponieważ pokrywa wszystkie przypadki klasy Lemon i ma większą moc. Warunek pokrywa także cztery elementy z innych klas- obiekty {5, 7, 9, 13}, dlatego w kolejnym kroku należy dobrać dodatkowy warunek, który wyeliminuje przypadki niechciane.
* Poszukując kolejnej kombinacji, nie uwzględniamy juz w zbiorze T(G) warunku [(Taste, sour)] i dobieramy kolejną parę, która ma najwięcej przypadków klasy Lemon i największą moc. Jest to para ***[(Shape, round)]***, ponieważ jako jedyna ma aż pięć przypadków rozpatrywanej klasy. Dodanie jej do warunku nie utworzy reguły, ponieważ nadal są pokryte, te same co poprzednio, obiekty innych klas.
* Następnymi parami branymi pod uwagę są [(Size, medium)] oraz [(Weight, 0.1)], ponieważ obie mają po cztery przypadki z badanej klasy. Algorytm wybiera parę **[(Weight, 0.1)]** ponieważ ma większą moc.
* Dzięki połączeniu trzech par: [(Taste, sour)], [(Shape, round)] i [(Weight, 0.1)] (T = {(Taste, sour), (Shape, round), (Weight, 0.1000)}, więc [T] = {1, 3, 18}) oraz [T] ⊂ B można utworzyć pierwszą, następującą regułę:

***JEŻELI taste = sour ORAZ shape = round ORAZ weight = 0.1 ORAZ size = medium TO fruit = lemon***

n = n ∪ [T] = Ø + {1, 3, 18} = {1, 3, 18}

G = B - [T] = {1, 3, 18, 19, 20} - {1, 3 ,18} = {19, 20}

* Klasa Lemon składa się z pięciu przypadków, natomiast powyższa reguła wyznaczyła tylko trzy z nich. Dlatego następnym krokiem jest wyliczenie reguły pokrywającej pozostałe dwa, postępując w ten sam sposób jak wyżej.
* Na nowo zostają wyszukane w zbiorze T(G) kombinacje z największą ilością obiektów należących do zbioru G= {19, 20}:

***[(Color, yellow)] = {19, 20, 12, 15, 16}***

***[(Size, medium)] = {19, 2, 5, 6, 9, 14, 15, 1, 3 , 18}***

***[(Size, small)] = { 20, 4, 8, 10, 13}***

***[(Shape, round)] = {19, 20, 4, 5, 6, 7, 8, 9, 10, 12, 13, 17, 21, 2, 1, 3 , 18}***

***[(Taste, sour)] = {19, 20, 5, 7, 9, 13, 1, 3 ,18}***

***[(Weight,0.1)] = {20, 4, 8, 13, 1 ,3 ,18}***

***[(Weight, 0.2)] = { 19, 10,14}***

Są to pary: [(Color, yellow)], [(Shape, round)] oraz [(Taste, sour)]. Każde z tych zestawień pokrywa oba przypadki. Największą moc ma jednak para **[(Color, yellow)]** W tym kroku przypadki klasy Lemon wyznaczone przez poprzednią regułę były rozpatrywane jak obiekty należące do innej klasy.

* Do warunku [(Color, yellow)] algorytm musi dodać następny, ponieważ są nim pokryte obiekty innych klas (przypadki {12, 15, 16}). Tak więc kolejną parą będzie **[(Taste, sour)]**, ponieważ ma większą moc jak [(Shape, round)].
* W wyniku połączenia warunków otrzymano:

T= {(Color, yellow), (Taste, sour)} [T] = {19, 20}

Ponieważ spełniony jest warunek [T] ⊂ B można utworzyć regułę 2:

***JEŻELI color = yellow ORAZ taste = sour TO fruit = lemon***

n = n ∪ [T] = {1, 3, 18} + {19, 20} = {1, 3, 18, 19, 20}

G = B - [T] = {1, 3, 18, 19, 20} - {1, 3, 18, 19, 20} = Ø

* G = Ø zatem wszystkie przypadki z kategorii Lemon zostały pokryte przez reguły. Algorytm kończy działanie dla tej klasy.

# Wyniki

W wyniku opisanego powyżej algorytmu powstało dwanaście reguł pokrywające wszystkie przypadki:

1. JEŻELI taste = sour

ORAZ shape = round

ORAZ weight = 0.1

ORAZ size = medium

TO fruit = lemon

Przypadki {1, 3, 18}

1. JEŻELI color = yellow

ORAZ taste = sour

TO fruit = lemon

Przypadki: {19, 20}

1. JEŻELI shape = round

ORAZ color = red

ORAZ size = medium

TO fruit = Apple

Przypadki: {2, 6, 9}

1. JEŻELI shape = round

ORAZ size = big

ORAZ color = Green

ORAZ taste = sweet

TO fruit = apple

Przypadek {17}

1. JEŻELI shape = round

ORAZ weight = 0.5

ORAZ color = red

TO fruit = apple

Przypadek {21}

1. JEŻELI weight = 0.3

ORAZ taste = sour

TO fruit = apple

Przypadek {5}

1. JEŻELI color = red

ORAZ size = small

TO fruit = cherry

Przypadek {4}

1. JEŻELI weight = 0.5

ORAZ size = big

ORAZ shape = round

ORAZ color = yellow

TO fruit = grapefruit

Przypadek {12}

1. JEŻELI weight = 0.5

ORAZ size = big

ORAZ taste = sour

TO fruit = grapefruit

Przypadek {7}

1. JEŻELI size = small

ORAZ shape = round

ORAZ weight = 0.1

ORAZ color = green

TO fruit = grape

Przypadki {8, 13}

1. JEŻELI color = maroon

TO fruit = grape

Przypadek {10}

1. JEŻELI shape = elongated

TO fruit = banana

Przypadki {11, 14, 15, 16}

# Wnioski

Algorytm LEM2 może analizować przypadki zamieszczone w niesprzecznej, jak i w sprzecznej bazie informacyjnej przy pomocy teorii zbiorów przybliżonych niezależnie od ich kolejności. Został stworzony w systemie uczenia maszynowego LERS. Reguły wyszukiwane są w sposób iteracyjny, a wynik to zbiór reguł, które pokrywają wszystkie przypadki ze zbioru uczącego. Jest algorytmem lokalnym, ponieważ pracuje na poziomie kombinacji atrybut – wartość dla zbioru przypadków o tej samej decyzji z tablicy decyzji. Algorytm nie zakończy swojej pracy, dopóki każdy przypadek klasy Ki nie jest pokryty regułą.

Z powyższej bazy wiedzy, przy pomocy algorytmu LEM2, powstało dwanaście reguł pokrywających wszystkie przypadki. Reguła dwunasta, dotycząca klasy Banana pokrywa wszystkie przypadki swojej kategorii, natomiast, na przykład, aby pokryć klasę Apple algorytm musiał stworzyć aż cztery reguły. Dzieje się tak, ponieważ algorytm wykorzystuje zawsze parę, która ma najwięcej wspólnych elementów ze zbiorem G i dopiero później sprawdza moc kombinacji. W przykładzie jest para [(Shape, round]), która zawiera aż 80% przypadków. W trakcie tworzenia reguł algorytm bierze tą parę pod uwagę, choć tak naprawdę nic ona nie wnosi do reguły (przykład Reguła 1). Właśnie dodawanie w trakcie tworzenia reguł zbędnych warunków jest największą wadą LEM2. Żaden z przypadków nie jest opisanych przez kilka reguł, co jest zaletą algorytmu.