ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Marta K. Smolińska, Zenon A. Sosnowski[1]

PODSUMOWANIA LINGWISTYCZNE Z GRUPOWANIEM ROZMYTYM

Streszczenie: W pracy przedstawiono zastosowanie podsumowania lingwistycznego jako predykatu rozmytego do wyznaczania obiektów z typową wartością atrybutu lub zbioru atrybutów. W rozmytym algorytmie grupującym wykorzystana jest populacja z wyznaczoną ze względu na dany atrybut typowością obiektów. Wyniki działania tego algorytmu oraz jego zmodyfikowanej postaci zostały przedstawione na przykładzie populacji, której obiektami są piksele obrazu.

Słowa kluczowe: podsumowania lingwistyczne, grupowanie rozmyte

1. Wstęp

W [1] został opisany język SummarySQL, który umożliwia definiowanie zapytań z podsumowaniami lingwistycznymi. Tam też, miedzy innymi, opisano algorytm wyznaczania typowych wartości (ang. *Typical Values*) oraz typowych klastrów (ang. *Typical Clusters*). W [2] przedstawiono algorytm znajdowania przybliżonych wartości typowych.

W niniejszej pracy analizowaną bazą obiektów był obraz. Obraz potraktowany jako zbiór obiektów, z których każdy reprezentowany jest przez położenie oraz kolor, jest bazą rzeczywistych danych, przy czym efekt klasteryzacji jest widoczny i łatwy w ocenie przez człowieka. Poza tym atrybut kolor może być traktowany jako pojedyncza wartość lub jako obiekt składający się z trzech atrybutów składowych RGB, co daje możliwość zastosowania różnych funkcji podobieństwa.

W rozdziale drugim opisaliśmy podsumowanie lingwistyczne, a w rozdziale trzecim algorytm znajdowania typowych wartości. W rozdziale czwartym przedstawiono algorytm grupowania typowych klastrów oraz jego modyfikacji dającej lepsze efekty grupujące w zastosowaniu do obrazów. W rozdziale piątym opisano

[1] Wydział Informatyki, Politechnika Białostocka, Białystok

algorytm znajdowania przybliżonych wartości typowych oraz jego postać zmodyfikowaną w zastosowaniu do kilku atrybutów.

2. Podsumowania lingwistyczne

Podsumowanie lingwistyczne jest zkwantyfikowanym rozmytym wyrażeniem postaci *„większość ludzi jest wysoka”* lub *„niewielu wysokich ludzi jest lekkich”*, gdzie *(wysocy)* ludzie są związani ze specyficzną populacją obiektów.

W zapisie ogólnym:

$$Q\ (obiektów)\ w\ (populacji)\ C_f\ \ jest\ S \tag{1}$$

C_f jest populacją rozmytą, czyli rozmytym zbiorem obiektów (każdemu obiektowi przypisany jest stopień przynależności do zbioru).

S jest rozmytym wyrażeniem lingwistycznym, zwanym sumatorem.

Q jest kwantyfikatorem lingwistycznym, np. *większość, niewiele, około połowa*.

Z podsumowaniem lingwistycznym związana jest wartość prawdy $\tau \in [0;1]$ nazywana miarą istotności podsumowania. Jest ona stopniem przynależności proporcji obiektów należących do populacji C_f, które spełniają S ($r \in [0;1]$), do zbioru rozmytego Q. Informuje ona, w jakim stopniu podsumowanie lingwistyczne zgodne jest z populacją C_f.

$$\tau = \mu_Q(r) \tag{2}$$

gdzie

$$r = \frac{card_f\left(S \cap C_f\right)}{card_f\left(C_f\right)} \tag{3}$$

a $card_f$ jest mocą zbioru rozmytego, zdefiniowaną jako:

$$card_f(B) = \sum_{o_i \in B} \mu_B(o_i) \tag{4}$$

Natomiast przecięcie $\cap$ jest zdefiniowane jako agregacja minimum lub inna dowolna t-norma.

Złożoność obliczenia stopnia prawdy τ podsumowania lingwistycznego w wypadku prostego zbioru rozmytego jest złożonością rzędu $O(|C_f|)$, gdzie $|C_f|$ jest liczbą obiektów w C_f.

Formalnie podsumowanie lingwistyczne zapisywane będzie jako predykat rozmyty:

$$\sum_Q \left(\mu_{C_f}(o_j) \middle| \mu_S(o_j)\right) \tag{5}$$

lub krócej

$$\sum_Q \left(\mu_{C_f} \middle| \mu_S\right) \tag{6}$$

gdzie $\sum_Q$ reprezentuje podsumowanie z kwantyfikatorem rozmytym Q. Predykat μ_{C_f} jest funkcją przynależności klasy rozmytej C_f, którą chcemy podsumować, a μ_S jest funkcją przynależności sumatora S.

3. Typowe wartości

Podsumowanie lingwistyczne, tak jak predykat rozmyty, związane jest z wartością prawdy z przedziału jednostkowego $[0;1]$. Na podstawie tej obserwacji możemy używać podsumowań lingwistycznych jako predykatów rozmytych, co możemy wykorzystać w wyznaczaniu typowej wartości.

Jeśli chcemy zapytać o *„osoby z typową wagą”*, to myślimy o osobach z populacji *C*, z wagą podobną do wagi u większości osób. Musimy, więc zdefiniować funkcję $\approx_w$ podobieństwa wagi, np.: $x \approx_w y = \max\left(1 - \frac{|x-y|}{\delta}, 0\right)$, gdzie δ jest stałą definiowaną przez użytkownika (np. 10% dziedziny atrybutu). Nową populację $C_{typical}$ zawierającą osoby z typową wagą wraz ze stopniem przynależności możemy opisać jako:

$$\mu_{C_{typical}}(o_i) = \mu_C(o_i) \wedge \sum_{most} \left(\mu_C(o_j) \middle| o_j.Weight \approx_w o_i.Weight\right) \tag{7}$$

Wartość prawdy podsumowania $\sum_{most} \left(\mu_C(o_j) \middle| o_j.Weight \approx_w o_i.Weight\right)$ definiuje rozmyty predykat dla *typowej wagi* w populacji C, a predykat $\mu_C(o_i)$ zapewnia, że typowy obiekt o_i nie może być bardziej typowy niż jego stopień przynależności do C.

3.1. Algorytm znajdowania typowych wartości

Każdy obiekt o_i z populacji C staje się prototypem obiektu, z typową wartością rozpatrywanego atrybutu a. Dla każdego z tych prototypów obliczane jest podsumowanie „większość obiektów w populacji ma wartość atrybutu a podobną do wartości tego atrybutu w obiekcie o_i". Wynikiem jest populacja $C_{typical}$, zawierająca obiekty wraz ze stopniem przynależności. Algorytm może znajdować typowe obiekty ze względu na kilka atrybutów, wtedy stosujemy iloczyn stopnia podobieństwa poszczególnych atrybutów. W [1] wynikiem tego algorytmu jest populacja zawierająca wartości atrybutów wraz z ich stopniem typowości. W obiektowych bazach danych [3] danymi wejściowymi algorytmu są: rozmyta populacja C oraz funkcja sim – podobieństwa między obiektami, ze względu na atrybut a (lub zbiór atrybutów). Jako wynik natomiast otrzymujemy populację $C_{typical}$, zawierającą obiekty wraz z ich stopniem typowości. Pseudokod tego algorytmu przedstawiono na rysunku 1.

1) **znajdowanie_typowych_obiektow**
2) foreach (object $o_i \in C$)
3) $sum = \sum_{most} \left(\mu_C(o_j) \middle| sim(o_j.a, o_i.a) \right)$
4) $C_{typical}.Add(o_i)\ with\ \mu_{C_{typical}}(o_i) = \min(sum, \mu_C(o_i))$;
5) end
6) return $C_{typical}$

Rys. 1. Algorytm znajdowania typowych obiektów ze względu na atrybut *a*.

4. Typowe klastry

Rysunek 2 przedstawia algorytm grupowania rozmytego przy użyciu typowej wartości. Przy zastosowaniu tego algorytmu w obrazie nie dawał on zbyt dobrych rezultatów, gdyż do każdego kolejnego klastra trafiały obiekty mniej do siebie podobne.

Lepsze efekty klasteryzacji uzyskaliśmy po zmodyfikowaniu punktu 4 do postaci przedstawionej na rysunku 3.

Tak zmodyfikowany algorytm umieszcza w klastrach elementy bardziej podobne niż algorytm oryginalny, ma jednak bardzo dużą wadę – jego złożoność obliczeniowa jest bardzo duża.

1. Znajdujemy typowe obiekty w populacji C ze względu na kryteria klastra. W wyniku tej operacji otrzymujemy populację $C_{typical}$. Kryteria klastra w wypadku obrazu to podobieństwo między dwoma kolorami.
2. Znajdujemy najbardziej typowy obiekt o'. Jest to obiekt z najwyższym stopniem typowości (najwyższy stopień przynależności w $C_{typical}$).

   $$\forall o \in C_{typical} : o'.\mu \geq o.\mu$$

   Zapisujemy ten obiekt w nowej populacji będącej *i*-tym klastrem U_i i usuwamy go z $C_{typical}$.
3. Poszukiwanie typowego klastra:
   Wśród obiektów, z $C_{typical}$, których stopień typowości jest większy niż α, znajdujemy obiekt podobny do przynajmniej jednego obiektu w klastrze U_i, dodajemy go do tego klastra i usuwamy z $C_{typical}$. Znajdowanie to kontynuujemy do momentu, w którym żaden obiekt nie zostanie dodany do klastra.
   Obiekty traktujemy jako podobne, jeśli podobieństwo między nimi jest większe niż α, która jest stałą ustaloną przez użytkownika. Na przykład:
   α - 75% z najbardziej typowej wartości ($\alpha = o'.\mu \cdot 75\%$).
   Użycie parametru α powoduje, ze klaster zawiera obiekty z $\alpha - przekroju$. Dzięki temu do klastra nie trafiają obiekty ze zbyt małym stopniem typowości (mniejszym niż α) lub zbyt mocno oddalone od elementów klastra.
4. Powtarzać od punktu 2 do osiągnięcia kryterium stopu, np. dopóki najbardziej typowy obiekt z $C_{typical}$ będzie miał stopień typowości większy lub równy β ($o'.\mu \geq \beta$). β jest stałą definiowaną przez użytkownika, np. 50% typowości najbardziej typowej wartości znalezionej w pierwszej iteracji.
   Kryterium stopu można też zdefiniować jako osiągnięcie zadanej liczby klastrów lub opróżnienie całej populacji $C_{typical}$.

Rys. 2. Algorytm grupowania rozmytego przy użyciu typowych wartości – typowe klastry.

Przykład 4.1.

W celu dokonania prezentacji została utworzona syntetyczna populacja prostych obiektów składających się z nazwy (indeksu) oraz wartości całkowitej z zakresu $[0;250]$. Każdy z obiektów należy do populacji ze stopniem przynależności 1. Tabela 1 przedstawia populację oraz efekt klasteryzacji algorytmem „ty-

powe klastry" z rysunku 2 oraz wersją zmodyfikowaną (rys. 3). W eksperymencie tym wartość $\alpha = 75\% \cdot o'.\mu$, przy czym $o'.\mu$ jest najwyższym stopniem przynależności w populacji $C_{typical}$. Natomiast relacja podobieństwa między dwoma obiektami została zdefiniowana jako podobieństwo między wartościami całkowitymi (wartościami drugiego atrybutu), opisane wzorem 8.

$$s(x, y) = \max\left(1 - \frac{|x - y|}{50}, 0\right) \tag{8}$$
$$x, y \in [0;250]$$

4. Zmodyfikować populację $C_{typical}$ w sposób następujący: stopień przynależności każdego elementu będącego w $C_{typical}$ zamienić na stopień przynależności, jaki ten obiekt miał w oryginalnej populacji, i ponownie obliczyć typowość dla wszystkich elementów w zmodyfikowanej $C_{typical}$.
Powtarzać od punktu 2 do osiągnięcia kryterium stopu.

Rys. 3. Modyfikacja punktu czwartego w algorytmie grupującym.

Przy takich parametrach oba algorytmy znalazły 6 klastrów, z tym że pierwszy algorytm umieścił w drugim klastrze więcej obiektów, które są mniej do siebie podobne. Klastry uzyskane przy użyciu drugiego algorytmu wydają się bardziej spójne, tzn. bardziej zwarte pod względem wartości atrybutu, na którego dziedzinie określona jest funkcja podobieństwa. Elementy do klastra dodawane są ze stopniem przynależności równym stopniowi typowości obiektu w populacji $C_{typical}$.

Tabela 1

Przykładowa populacja oraz klastry uzyskane algorytmem typowe klastry i wersją zmodyfikowaną

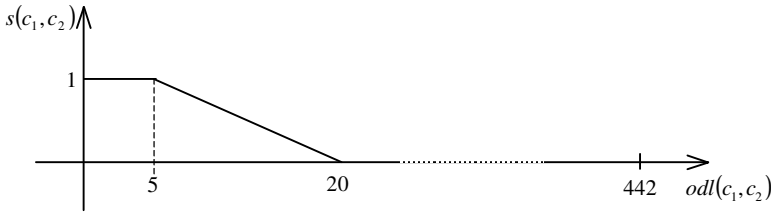
Populacja			Klastry			Klastry uzyskane algorytmem zmodyfikowanym		
1	200	1	1	200	0,26	1	200	0,26
2	220	1	2	220	0,24	2	220	0,24
3	10	1	5	190	0,24	5	190	0,24
4	20	1	8	210	0,26	8	210	0,26
5	190	1	9	200	0,26	9	200	0,26
6	100	1	19	190	0,24	19	190	0,24

7	150	1	14	30	0,21	11	70	0,55
8	210	1	3	10	0,18	10	50	0,48
9	200	1	4	20	0,2	12	50	0,48
10	50	1	10	50	0,19	14	30	0,54
11	70	1	11	70	0,17	3	10	0,47
12	50	1	12	50	0,19	4	20	0,47
13	250	1	16	110	0,18			
14	30	1	20	100	0,18			
15	240	1	6	100	0,18			
16	110	1	15	240	0,16	16	110	0,75
17	130	1	13	250	0,12	6	100	0,67
18	0	1				17	130	0,6
19	190	1				20	100	0,67
20	100	1	17	130	0,15	7	150	0,48
			18	0	0,14	13	250	0,75
						15	240	0,75
			7	150	0,11	18	0	1

Na rysunku 5 przedstawiono efekt działania obu algorytmów na obrazie. Rysunek 5.b przestawia obraz po grupowaniu bez wyznaczania typowości po znalezieniu każdego klastra. Przy zastosowaniu miary podobieństwa między kolorami, opartej na odległości euklidesowej, zdefiniowanej wzorem 9, z reprezentacją graficzną przedstawioną na rysunku 4.

$$s(c_1,c_2)=\begin{cases}1, & odl <= 5\\ -\dfrac{odl-5}{15}+1, & 5 < odl <= 20\\ 0, & odl > 20\end{cases} \tag{9}$$

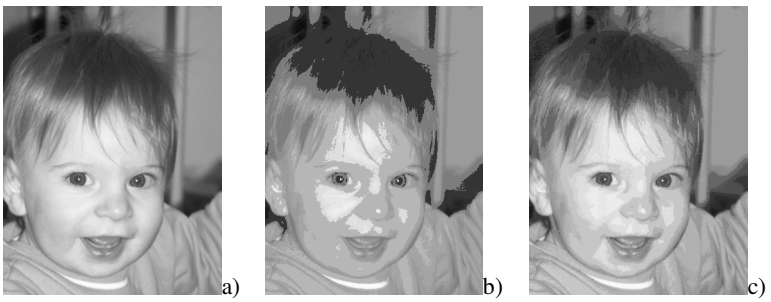
$$odl(c_1,c_2)=\sqrt{(c_1.R-c_2.R)^2+(c_1.G-c_2.G)^2+(c_1.B-c_2.B)^2} \tag{10}$$



Rys. 4. Funkcja podobieństwa między dwoma kolorami, zależna od odległości między tymi kolorami

Algorytm znalazł 30 klastrów z 247 oryginalnych kolorów, i $340 \times 449 = 152660$ obiektów (pikseli). Na rysunku klaster to jeden kolor. Do reprezentacji klastra wybierany jest kolor, z najwyższym stopniem przynależności w klastrze

Zmodyfikowany algorytm znalazł 56 klastrów, przy czym obraz wynikowy jest bardziej podobny do oryginału (rys. 5.c).



a) b) c)

Rys. 5. Wynik działania algorytmu typowego grupowania a) obraz oryginalny, b) algorytm typowej klasteryzacji, c) zmodyfikowany algorytm typowego grupowania.

5. Algorytm znajdowania przybliżonych wartości typowych

Algorytm typowej klasteryzacji ma bardzo dużą złożoność, bo sam algorytm znajdowania typowych wartości ma złożoność $O\left(|C|^2\right)$, gdzie $|C|$ jest liczbą obiektów w populacji C. W [2] autorzy przedstawili algorytm znajdowania przybliżonych wartości typowych. Pseudokod tego algorytmu przedstawiliśmy na rysunku 6.

Danymi wejściowymi dla tego algorytmu są: rozmyty zbiór obiektów O, A — atrybut, dla którego poszukiwana jest typowa wartość oraz zbiór rozłącznych przedziałów $L = \{L_1, ..., L_n\}$ zdefiniowanych w dziedzinie atrybutu A, takich że

$L_j \cap L_k = \varnothing$ jeśli $j \neq k$

i suma przedziałów definiuje całą dziedzinę atrybutu A, a więc

$L_1 \cup ... \cup L_n = range(A)$

1) proc findTypicalValapprox
2) cardArray[n] = [0,...,0]
3) forall object $o_i \in O$ do
4) j = findIntervalIndex($o_i.A$, *L)*
5) cardArray[j] = cardArray[j]+ $\mu_o(a_i)$
6) end
7) forall intervals $L_j \in L$ do
8) $r_j = \sum_k \left(mean(L_j) \approx mean(L_k)\right) * cardArray[k],\ k \in [1,n]$
9) $\mu_j = \dfrac{r_j}{fcard(O)}$
10) extend $typical_A$ with $\{\mu_j / mean(L_j)\}$
11) end
12) return $typical_A$
13) end

Rys. 6. Algorytm znajdowania przybliżonych wartości typowych

Funkcja $mean(L_j)$ zwraca medianę (element, który w uporządkowanej próbce znajduje się na jej środku) przedziału o indeksie j.

Złożoność tego algorytmu jest złożonością rzędu $O(|O|+n^2)$ $|O|$ - liczba obiektów w *O, n* – liczba przedziałów.

Złożoność tę można zmniejszyć do, $O(|O|+n\cdot \max_j(K_j))$, jeśli r_j obliczamy tylko dla indeksu $k \in K_j$, gdzie $mean(L_{jk}) \in [mean(L_j)-\delta, mean(L_j)+\delta]$ i δ jest odległością, dla której relacja podobieństwa jest różna od 0.

Algorytm ten zmodyfikowany na nasz użytek przedstawiono na rysunku 7. Danymi wejściowymi są: populacja C zawierająca piksele opisane przez położenie w obrazie $(p.x, p.y)$ oraz kolor składający się z 3 składowych (R,G,B). Każda ze składowych ma wartość z zakresu $[0;255]$. Jako zbiór przedziałów będziemy traktować zbiór sześcianów wyodrębnionych z sześcianu RGB, poprzez podział każdego z boków na x równych części. Liczba powstałych w ten sposób przedziałów to y^3 ($y = 255/x$). W wyniku otrzymujemy populację $C_{typicalIntervals}$, zawierającą przedziały wraz z ich stopniem typowości.

1. **TypicalValue_Approximated**
2. cardArray[y,y,y] = {0…0; 0…0; 0…0}
3. foreach (object $o_i \in C$)
4. i = $o_i.R / x$; j = $o_i.G / x$; k = $o_i.B / x$;
5. cardArray[i, j, k] += $\mu_C(o_i)$;
6. if ($L_{i,j,k} \notin C_{typicalIntervals}$)
7. $C_{typicalIntervals}.Add(L_{i,j,k})$;
8. $L_{i,j,k}.Add(o_i)$;
9. end
10. foreach (interval $L_{i,j,k} \in C_{typicalIntervals}$)
11. $r_{i,j,k} = 0$;
12. foreach(interval $L_{ii,jj,kk} \in C_{typicalIntervals}$)
13. $r_{i,j,k} += sim(L_{i,j,k}, L_{ii,jj,kk}) \cdot cardArray[i, j, k]$
14. $L_{i,j,k}.\mu = \frac{r_{i,j,k}}{fcard(C)}$;
15. end
16. end
17. return $C_{typicalIntervals}$

Rys. 7. Algorytmu znajdowania przybliżonych wartości kolorów w obrazie

Grupowanie przebiega tak samo jak przy algorytmie z rysunku 2 lub 3, z tym że w tym wypadku grupowane są przedziały. Grupowaniu poddajemy tylko te przedziały, do których trafiły jakieś obiekty, bo zależy nam na znalezieniu grup obiektów. Kolor w obrazie wynikowym jest kolorem przedziału z najwyższym stopniem przynależności w klastrze, do którego należy przedział zawierający obiekt. W przedstawionych obrazach długość przedziału wynosiła 1. Maksymalna liczba niepustych przedziałów dla obrazów w odcieniach szarości wynosi 256. Jednak w wypadku obrazów kolorowych długość przedziału musi być większa, gdyż przy przedziale jednostkowym ich liczba może wynieść $256^3 = 16777216$. Bardzo dobre rezultaty uzyskaliśmy przy długości boku równej 8 i 16.

a) b) c)

Rys. 8. Wynik działania algorytmów grupujących z przybliżonym wyznaczaniem typowości i funkcją podobieństwa opartą na euklidesowej odległości między kolorami a) obraz oryginalny (213 kolorów) b) algorytm grupujący (34 klastry) c) zmodyfikowany algorytm grupujący (18 klastrów)[2]

Wykonane zostały eksperymenty z zastosowaniem dwóch różnych relacji podobieństwa. Podobieństwo określane dla kolorów będących środkami przedziałów według wzoru 9 (rysunek 8) oraz funkcja podobieństwa oparta na reprezentacji wartości atrybutu jako liczby rozmytej.

5.1. Miara podobieństwa oparta na liczbach rozmytych

Jeśli każdą ze składowych koloru potraktujemy jako trójkątną liczbę rozmytą, reprezentowaną jako $A = (m_A, \alpha_A, \beta_A)$

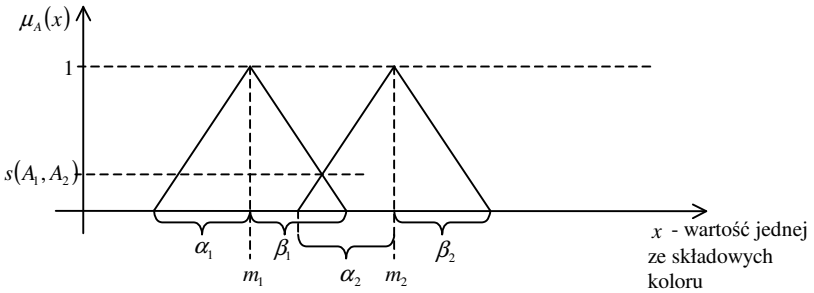
$$\mu_A(x) = \begin{cases} 0 & x \le m_A - \alpha_A \\ \dfrac{x - (m_A - \alpha_A)}{\alpha_A} & m_A - \alpha_A < x \le m_A \\ \dfrac{m_A + \beta_A - x}{\beta_A} & m_A < x < m_A + \beta_A \\ 0 & x \ge m_A + \beta_A \end{cases} \tag{11}$$

to podobieństwo między liczbami rozmytymi $A_1 = (m_1, \alpha_1, \beta_1)$ i $A_2 = (m_2, \alpha_2, \beta_2)$ możemy określić wzorem (12). Graficzną reprezentację tego podobieństwa ilustruje rysunek 9.

[2] Obrazy przestawione na rysunku 8 i 11 pochodzą z bazy [4]

$$s(A_1, A_2) = \begin{cases} 1, & m_1 = m_2 \\ \dfrac{\beta_1 + m_1 - m_2 + \alpha_2}{\alpha_2 + \beta_1}, & (m_1 < m_2) \wedge (m_2 - m_1 \le \beta_1 + \alpha_2) \\ \dfrac{\beta_2 + m_2 - m_1 + \alpha_1}{\alpha_1 + \beta_2}, & (m_2 < m_1) \wedge (m_1 - m_2 \le \beta_2 + \alpha_1) \\ 0, & ((m_1 < m_2) \wedge (m_2 - m_1 > \beta_1 + \alpha_2)) \vee \\ & \vee ((m_2 < m_1) \wedge (m_1 - m_2 > \beta_2 + \alpha_1)) \end{cases} \quad (12)$$

$$s(c_1, c_2) = s(c_1.R, c_2.R) \cdot s(c_1.G, c_2.G) \cdot s(c_1.B, c_2.B) \quad (13)$$



Rys. 9. Podobieństwo między dwiema liczbami rozmytymi

Dla tak zdefiniowanej miary podobieństwa z wartościami $\alpha = \beta = 8$ algorytm grupowania z przybliżonym znajdowaniem typowej wartości dał rezultaty przedstawione na rysunku 10 i 11. W obrazie na rysunku 10 kolor został potraktowany jako trzy oddzielne atrybuty, a miarą podobieństwa obiektów był iloczyn podobieństwa trzech składowych R, G i B, obliczonych wg wzoru 13. Takie potraktowanie koloru zdaje się bardziej zasadne przy obrazach kolorowych (których ze względu na czarno-biały druk nie możemy zaprezentować). Przy odcieniach szarości można sprawdzać tylko jedną ze składowych i obliczać podobieństwo dla tej jednej składowej, bo wszystkie mają taką samą wartość dla danego obiektu. Wyniki z uwzględnieniem takiego podobieństwa prezentuje rys. 11.

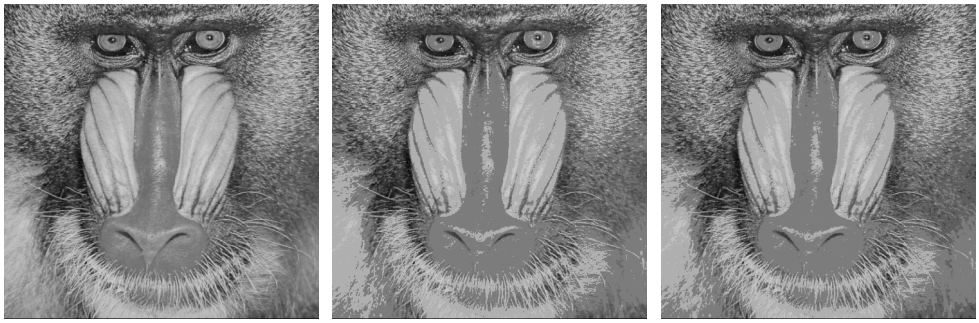
6. Podsumowanie

We wszystkich przedstawionych eksperymentach kryterium stopu algorytmu grupującego było opróżnienie populacji $C_{typical}$, tak aby pogrupowane były wszystkie obiekty. Algorytm zmodyfikowany daje lepsze efekty grupowania. W obrazach

czarno-białych tworzy mniej klastrów, przy czym obraz jest bardziej podobny do oryginału. W eksperymentach prowadzonych na obrazach kolorowych algorytm zmodyfikowany przeważnie tworzył większą liczbę klastrów, cały czas jednak lepiej odwzorowując oryginał. Oczywistą wadą tej modyfikacji jest złożoność, która przy dużych populacjach lub też (w przypadku algorytmu wyznaczającego przybliżone wartości typowe) szerokich dziedzinach atrybutów, których podział na przedziały nie daje wcale mniejszej ich liczby niż liczebność populacji, jest nie do przyjęcia. W przeciwnym wypadku użycie algorytmu wydaje się zasadne w celu uzyskania lepszych efektów.

a) b) c)

Rys. 10. Wynik działania algorytmu grupującego z przybliżonym znajdowaniem typowości a) obraz oryginalny (247 kolorów), b)algorytm grupujący 30 klastrów, c) zmodyfikowany algorytm grupujący 21 klastrów.



a) b) c)

Rys. 11. Wynik grupowania z przybliżonym wyznaczaniem typowości z funkcją podobieństwa opartą na liczbach rozmytych. a) obraz oryginalny kolorów 218, b) algorytm grupowania (34 klastry), c) zmodyfikowany algorytm grupowania (16 klastrów)

Literatura

[1] Rasmussen D.: *Application of the Fuzzy Query Language – Summary SQL*, DATALOGISKE SKRIFTER, Roskilde University, 1997
[2] Rasmussen D., Yager R. R.: *Introdution of Fuzzy Characteristic Rules by Typical Values*, DATALOGISKE SKRIFTER, Roskilde University, 1997
[3] Smolińska M. K., Sosnowski Z. A.: *Podsumowania lingwistyczne w obiektowych bazach danych*, XIII Warsztaty Naukowe PTSK, Symulacja w Badaniach i Rozwoju, Wrzesień 2006
[4] *The USC_SIPI Image Database,* [http://sipi.usc.edu/database/index.html] University of Southern California

LINGUISTIC SUMMARIES WITH FUZZY CLUSTERING

Abstract: This paper presents linguistic summary as a fuzzy predicate, which is used to find, objects with typical values of an attribute or a set of attributes. In the fuzzy clustering algorithm we use population with given typicality of objects for selected attribute. We present the results of this algorithm and its modification basing on an example with population of pixels in image.

Keywords: linguistic summaries, fuzzy clustering

Artykuł zrealizowano w ramach prac badawczych S/WI/2/03 i 3T11F01130.