

Cezary Zawadka
Robert Krajewski
Marcin Gecow

Warszawa 2013.01.08

CPOO

Sprawozdanie

prowadzący projekt: mgr inż. Tomasz Olszewski

1. Opis projektu

Celem projektu jest porównanie ekstrakcji reprezentatywnych dla obrazu kolorów za pomocą samodzielnie zaimplementowanych algorytmów grupowania. Do realizacji zostały wybrane następujące algorytmy: k-means, DBSCAN oraz hierarchiczne.

Algorytmy grupowania znajdują zastosowanie w dziedzinie grafiki komputerowej. Wybrane metody mogą być wykorzystane np. przez grafików, projektantów stron internetowych do analizy aktualnej palety kolorów na stronie, aby odpowiednio dobrać dodatki lub wprowadzić zmiany, które nie będą odstawały od aktualnej wersji kolorystycznej. Analogiczne postępowanie można przeprowadzać dla innych zastosowań w grach, obrazach itp.

2. Opis algorytmów

K-Means

Algorytm *k-średnich* ma na celu pogrupowanie n obserwacji (x_1, x_2, \dots, x_n) na k zbiorów $S = \{S_1, S_2, \dots, S_k\}$, poprzez minimalizację funkcji celu. W naszej implementacji funkcją celu jest suma odległości próbek x_i od środków zbiorów S_i . Do wyznaczenia odległości może zostać użyta dowolna metryka.

$$E = \sum_{i=1}^k \sum_{x_i \in S_i} D(x_i, C_i)$$

gdzie C_i jest średnią wartością punktów w S_i , natomiast $D(x_j, C_i)$ oznacza odległość w zadanej metryce. W naszej implementacji zastosowaliśmy 2 metryki: euklidesową oraz Manhattan.

Początkowy zbiór k wartości średnich wyznaczany jest w sposób losowy z pikseli obrazu, następnie algorytm sprowadza się do dwóch kroków wykonywanych iteracyjnie:

Krok przypisania:

Polega na przypisaniu każdej obserwacji do klastra z najbliższą średnią.

$$S_i^{(t)} = \{x_p : D(x_p, C_i) \leq D(x_p, C_j) \forall 1 \leq j \leq k\}$$

Gdzie każdy x_p przypisywany jest dokładnie jednemu $S_i^{(t)}$, nawet jeżeli mógłby być przypisany do większej ich liczby.

Krok aktualizacji:

Polega na obliczeniu nowych wartości średnich mających być centroidami obserwacji w klastrze.

$$m_i^{(t+1)} = \frac{1}{|S_i^t|} \sum_{x_i \in S_i^t} x_i$$

Algorytm zostaje przerwany, kiedy jego kolejne iteracje nie zmieniają już więcej przypisania obserwacji do klastrów.

DBSCAN

Algorytm DBSCAN (ang. **D**ensity **B**ased **S**patial **C**lustering of **A**pplications with **N**oise) opiera się na gęstościowej definicji klastra. Potrafi znajdować grupy o dowolnym kształcie oraz wskazywać próbki stanowiące szum, w dodatku nie potrzebuje informacji o faktycznej liczbie klastrów.

Jedynymi dwoma parametrami wejściowymi algorytmu są ϵ oraz *minPts*. Epsilon oznacza maksymalną odległość pomiędzy punktami *gęstościowo osiągalnymi*. Z kolei *minPts* to minimalna liczba punktów w klastrze.

Algorytm rozpoczyna się od arbitralnego wybrania punktu początkowego, następnie sprawdzane jest jego ϵ -sąsiedztwo. Jeśli zawiera ono mniej niż *minPts* punktów, punkt ten uznaje się za szum i wybiera następny. W przeciwnym wypadku (ϵ -sąsiedztwo zawiera przynajmniej *minPts* punktów) tworzony jest nowy klaster. Do tego klastra dodawane są kolejne punkty leżące w ϵ -sąsiedztwach wyznaczanych dla wszystkich punktów z pierwotnego ϵ -sąsiedztwa. W momencie, gdy zostanie znaleziony cały *gęstościowo powiązany* klaster, wybierany jest kolejny dotąd nieodwiedzony punkt.

W najprostszej i najbardziej popularnej wersji odległości pomiędzy punktami wyznaczone są na podstawie prostej miary euklidesowej; dla danych wielowymiarowych oznaczać to będzie zazwyczaj stosunkowo słabą wydajność algorytmu (czas wykonania jest kwadratowy przy braku dodatkowych struktur usprawniających działanie).

Pseudokod algorytmu DBSCAN.

```

1.  DBSCAN(D, eps, MinPts)
2.  C = 0
3.  for each unvisited point P in dataset D
4.    mark P as visited
5.    NeighborPts = regionQuery(P, eps)
6.    if sizeof(NeighborPts) < MinPts
7.      mark P as NOISE
8.    else
9.      C = next cluster
10.     expandCluster(P, NeighborPts, C, eps, MinPts)
11.
12.  expandCluster(P, NeighborPts, C, eps, MinPts)
13.  add P to cluster C

```

```

14.     for each point P' in NeighborPts
15.         if P' is not visited
16.             mark P' as visited
17.             NeighborPts' = regionQuery(P', eps)
18.             if sizeof(NeighborPts') >= MinPts
19.                 NeighborPts = NeighborPts joined with NeighborPts'
20.             if P' is not yet member of any cluster
21.                 add P' to cluster C
22.
23.     regionQuery(P, eps)
24.     return all points within P's eps-neighborhood

```

Złożoność obliczeniowa algorytmu wynosi zależnie od metody indeksowania danych - $O(n^2)$ (jak w przygotowanej implementacji) lub $O(n \log n)$ (w przypadku zastosowania R-drzew). Złożoność pamięciową również można oszacować jako $O(n^2)$.

Parametry zastosowane w tym algorytmie nie pozwalają na zadanie żądanej liczby klastrów wyjściowych. Nie zawsze również otrzymanie na wyjściu ich małej liczby oznacza dobry wynik. Przykładowy wynik złego dobru parametrów dla obrazu 1:

| Obraz analizowany | Otrzymane grupy |
|--|--|
|  |  |

W powyższej tabeli jest przedstawiony wynik pracy algorytmu dla $\epsilon=1.0$ oraz $\text{minPts}=6$. W rezultacie otrzymano 13 klastrów narysowanych w kolejnych pionowych liniach. Można łatwo zaobserwować, że brakuje kolorów szarego oraz pomarańczowego. Dobór parametrów dla tego algorytmu jest nietrywialny. Dla osiągnięcia zadowalających wyników należy przeprowadzić wiele prób dla różnych wartości i ręcznie wybrać najlepsze. Zautomatyzowanie tego procesu wymagałoby na przykład przeprowadzenia procesu uczonego na wielu różnych przykładach co dopiero mogłoby przynieść odpowiednie rezultaty.

Grupowanie hierarchiczne

Rodzina algorytmów grupowania hierarchicznego charakteryzuje się specyficznym sposobem budowania grup. Wyróżnia się dwa główne podejścia: wstępujące i zstępujące. Pierwsze z nich polega na systematycznym tworzeniu drzewa (hierarchii) rozpoczynając od grup będących pojedynczymi przykładami i w każdej iteracji scalając dwie najbliższe położone grupy. Algorytm zatrzymuje się po osiągnięciu żądanej liczby klastrów lub kiedy nie da się już dalej łączyć. Druga metoda, algorytm zstępujący, wykonuje działania w odwrotnym kierunku. Początkowo występuje jedna grupa. W wyniku kolejnych iteracji najbardziej separowalna z grup

jest dzielona. Pętla powtarzana jest do momentu gdy każdy przykład będzie stanowił oddzielną grupę.

Odległości pomiędzy grupami są wyznaczone przy użyciu metryk odległości takich jak: euklidesowa, Manhattan, Canberra.

Asymptotyczną złożoność obliczeniową algorytmu oszacować można na $O(n^3)$ gdzie n to liczba pikseli. Można to zaobserwować w przedstawionym poniżej pseudokodzie, gdzie w liniach 6, 10 oraz 11 są zagnieżdżone pętle. Złożoność pamięciowa to $O(n)$.

Pseudokod algorytmu grupowania hierarchicznego.

```

1.   hclust(k, points)
2.   clusters <- ∅
3.   for i in 0..points.size
4.     clusters.add(points[i])
5.   end
6.   for it in 0..(points.size-k)
7.     clust_a <- -1
8.     clust_b <- -1
9.     min_dist <- ∞
10.    for i in 0..clusters.size
11.      for j in i+1..clusters.size
12.        dist <- distance(clusters[i], clusters[j])
13.        if dist < min_dist
14.          min_dist <- dist
15.          clust_a <- i
16.          clust_b <- j
17.        end
18.      end
19.    end
20.    connect(clust_a, clust_b)
21.  end
22. end

























```









































Funkcja *distance* znajduje odległość pomiędzy dwiema grupami A i B. Jest ona tutaj zdefiniowana jako odległość najbardziej oddalonych od siebie punktów a i b takich, że $a \in A$ i $b \in B$. Tak rozumianą odległość nazywa się *kompletnie łączącą* (z ang. *complete link*). Odległość między a i b jest dowolną metryką. Na potrzeby omawianego zagadnienia przyjęliśmy metrykę euklidesową co oznacza, że odległość między pikselami $a = (a_r, a_g, a_b)_i$, $b = (b_r, b_g, b_b)$ mierzona jest następująco: $d(a,b) = \sqrt{(a_r - b_r)^2 + (a_g - b_g)^2 + (a_b - b_b)^2}$

3. Badania

Do realizacji projektu został wybrany język programowania Java oraz środowisko NetBeans.

W procesie testowania, algorytm DBSCAN zastosowano dla różnych podzbiorów parametrów a następnie wyniki zostały ręcznie wybrane wedle subiektywnej oceny. Dla każdego obraz przedstawiony jest wynik z jednego przebiegu algorytmu, jednak w trzech przypadkach - dla 3, 5 oraz 7 głównych kolorów posegregowanych od lewej dla najczęściej występującego. Realnie algorytm zwykle kończył pracę z większą liczbą klastrów wynoszącą nClust przy zastosowaniu parametrów eps oraz minPts. Pozostałe kolory zostały pominięte w sprawozdaniu.

| | | KMEANS | DBSCAN | HIERARCHICZNE |
|--|---------|---|--|---|
|  obraz 1 | 3 grupy |  |  eps=3.5 minPts=6 nClust=15 |  |
| | 5 grupy |  |  |  |
| | 7 grup |  |  |  |
|  obraz 2 | 3 grupy |  |  eps=3.0 minPts=1 nClust=104 |  |
| | 5 grupy |  |  |  |
| | 7 grup |  |  |  |
|  obraz 3 | 3 grupy |  |  eps=6.0 minPts=4 nClust=16 |  |

| | | | | |
|--|---------|---|--|---|
| | 5 grupy |  |  |  |
| | 7 grup |  |  |  |
|  obraz 4 | 3 grupy |  |  eps=4.0 minPts=5 nClust=8 |  |
| | 5 grupy |  |  |  |
| | 7 grup |  |  |  |
|  obraz 5 | 3 grupy |  |  eps=20.0 minPts=5 nClust=9 |  |
| | 5 grupy |  |  |  |
| | 7 grup |  |  |  |
|  obraz 6 | 3 grupy |  |  eps=10.5 minPts=2 nClust=34 |  |
| | 5 grupy |  |  |  |
| | 7 grup |  |  |  |
|  obraz 7 | 3 grupy |  |  eps=10.5 minPts=1 nClust=7 |  |

| | | | | |
|-------------|---------|--|--------------------------------------|--|
| | 5 grupy | | | |
| | 7 grup | | | |
| obraz 8 | 3 grupy | | eps=10.0 minPts=8 nClust=8 | |
| | 5 grupy | | | |
| | 7 grup | | | |
| obraz 9 | 3 grupy | | eps=10.0 minPts=4 nClust=8 | |
| | 5 grupy | | | |
| | 7 grup | | | |

3.1 Zastosowanie różnych metryk odległości

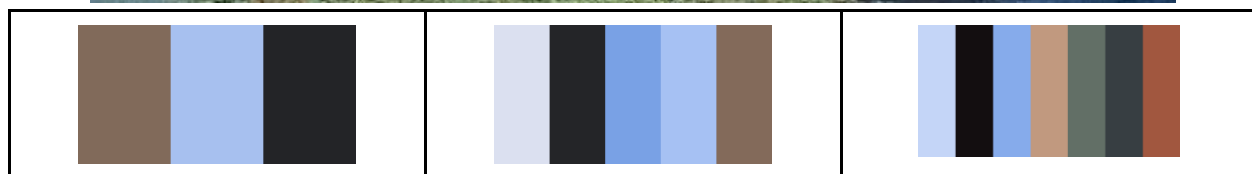
Dla algorytmu k-means zbadano wpływ wyboru metryki odległości na uzyskiwane kolory reprezentatywne. Uzyskane wyniki dla metryki euklidesowej i Manhattan nie różnią się znacząco, a widoczne różnice mogą wynikać również z losowego doboru wartości początkowych.

Poniższa tabela zawiera porównanie wyników dla metryki euklidesowej i metryki Manhattan.

| | Odległość euklidesowa | Odległość Manhattan |
|--|---|---|
|  obraz 1 |  |  |
|  obraz 2 |  |  |
|  obraz 3 |  |  |
|  obraz 4 |  |  |
|  obraz 5 |  |  |
|  obraz 6 |  |  |
|  obraz 7 |  |  |
|  obraz 8 |  |  |
|  obraz 9 |  |  |

3.2 Zastosowanie algorytmu k-means dla obrazów powyżej 250x250 pikseli.

Dla obrazów o rozmiarze przekraczającym kilka tysięcy pikseli jedynym możliwym do użycia algorytmem, ze względów wydajnościowych, okazał się k-means. Poniżej zamieszczono 3 obrazu o rozmiarze około 600x350 pikseli wraz z wyliczonymi kolorami reprezentatywnymi, odpowiednio dla 3, 5 i 7 grup.





4. Wyniki i wnioski

4.1 Czas działania algorytmów

Najszybszym okazał się być algorytm k-means. Jest to algorytm heurystyczny z elementami losowości, dlatego oszacowanie rzeczywistej złożoności obliczeniowej nie jest możliwe. Liczba operacji w jednej iteracji wynosi $\Theta(nk)$ gdzie n to liczba pikseli, k liczba grup. Zauważyliśmy, że liczba iteracji dla badanych obrazków jest wprost proporcjonalna do liczby grup k i nie zależy od rozmiaru zdjęcia. Złożoność algorytmu k-means szacujemy na $\Theta(nk^2)$.

Na średniej klasy komputerze stacjonarnym algorytm k-means potrzebował średnio poniżej 0,1 sekundy dla każdej próbki (obrazy 1-9), dla dużych obrazów z punkt 3.2 (rozmiar około 600x350 pikseli) czas wykonywania algorytmu wynosił około 2 sekund.

Wolniejszy okazał się algorytm DBSCAN. Zmierzony czas działania algorytmu dla jednej próbki to około 2 sekundy.

Najgorszy pod tym względem okazał się, zgodnie z przewidywaniami, algorytm grupowania hierarchicznego. Wykonywał się on dla jednej próbki około 30 sekund. Z tego powodu maksymalna wielkość obrazka została ograniczona do 50x50 pikseli (co i tak skutkuje ilością obliczeń w przybliżeniu równą $(50 * 50)^3 = 15\,625\,000\,000$).

4.2 Łatwość użycia

Dzięki możliwości podania jako parametr jedynie liczby pożądanych klastrów wyjściowych, algorytmy k-means oraz hierarchiczny są bardzo łatwe w użyciu. Stosowanie metody DBSCAN pozostaje znacznie trudniejsze ze względu na mało intuicyjne parametry, które muszą być odpowiednio dobierane w zależności od charakterystyki analizowanego obrazka - np. liczby kolorów oraz różnic między nimi.



4.3 Ocena wizualna

Badanie różnic w przeprowadzonych grupowaniach zostało dokonane w sposób subiektywny. Można ogólnie stwierdzić, że im większa złożoność obrazka (w sensie ilości kolorów i kształtów) tym większe rozbieżności poszczególnych algorytmach.

Największe różnice w wynikach są widoczne w przypadku obrazka 2. Otrzymane kolory wydają się najlepiej odzwierciedlać rzeczywisty rozkład kolorów w przypadku algorytmu k-means. Pozostałe algorytmy radzą sobie gorzej. Można zauważyć, że grupowanie hierarchiczne wybrało jak reprezentanta kolor czarny, który w rzeczywistości występuje rzadko.

Podobna sytuacja występuje w przypadku obrazka 6. Tutaj jednak najlepiej poradził sobie algorytm DBSCAN. Widać to dla każdej ilości grup. Pozostałe algorytmy są porównywalne i działają tym lepiej im większa jest ilość grup.

Najbardziej podobne wyniki uzyskano dla obrazków 1, 3, 4, 5 oraz 8. Warto zauważyć, że większość z tych obrazków (z wyjątkiem 4,5) składa się praktycznie tylko z 3 kolorów. W tabeli poniżej znajduje się porównanie wartości RGB kolorów wynikowych dla dwóch nietrywialnych obrazków (4,5). Kolory tła oznaczają subiektywne podobieństwa, tło białe oznacza, że kolory nie są do siebie zbliżone.

| | KMEANS | DBSCAN | HIERARCHICZNE |
|---|---|---|---|
|  obraz 4, 5 grup | r:210 g:181 b:151 r:079 g:071 b:069 r:126 g:107 b:098 r:039 g:035 b:033 r:021 g:019 b:019 | r:042 g:040 b:041 r:103 g:097 b:101 r:197 g:156 b:136 r:098 g:079 b:064 r:058 g:053 b:047 | r:030 g:029 b:027 r:187 g:154 b:135 r:230 g:214 b:162 r:123 g:105 b:093 r:077 g:067 b:066 |
|  obraz 5, 5 grup | r:198 g:215 b:222 r:240 g:243 b:241 r:030 g:113 b:190 r:003 g:005 b:017 r:187 g:148 b:106 | r:225 g:233 b:236 r:000 g:005 b:018 r:025 g:117 b:194 r:145 g:166 b:185 r:205 g:141 b:067 | r:080 g:066 b:102 r:000 g:004 b:015 r:225 g:233 b:236 r:209 g:156 b:090 r:052 g:128 b:190 |

Różnice metody DBSCAN względem pozostałych dwóch są również podyktowane faktem, że jako jedyny stosuje kategorię szumu oraz posiada zwykle na wyjściu więcej klastrów niż te prezentowane w porównaniu. W wyniku minimalizowania liczby klastrów w innych algorytmach może się zdarzyć, że mało liczny kolor zostanie sklasyfikowany jako jeden z głównych.

4.4 Podsumowanie

Spośród zastosowanych algorytmów najlepszy okazał się k-means. Podstawową jego zaletą jest jego czas działania. Pozwala na stosowanie go na dużych obrazach w rozsądnym czasie. Subiektywne wyniki oceny wizualnej wykazują, że jego jakość nie odstępuje od pozostałych testowanych metod. Jedyną wadą algorytmu k-means jest jego niedeterministyczne działania spowodowane losowym doбором początkowych wartości, powoduje to, że każde uruchomienie algorytmu, dla tego samego obrazu daje nieco różne wyniki.

Algorytm DBSCAN również działa w przystępnym czasie jednak utrudnienie w postaci konieczności doboru parametrów dla każdego obrazka osobno znacznie wydłuża faktyczny czas potrzebny na analizę. Ze względu na to, że parametry są dobierane ręcznie i wyniki subiektywnie oceniane przez użytkownika, to porównanie z pozostałymi algorytmami nie jest miarodajne.

Algorytm grupowania hierarchicznego okazał się działać najwolniej, co dyskwalifikuje go w praktycznym zastosowaniu. Wyniki wizualne nie ustępują pozostałym metodom.

Zmiana metryki odległości między pikselami nie wpływa znacząco na rezultaty, co zostało pokazane stosując najszybszy algorytm - k-means.