

Clustering



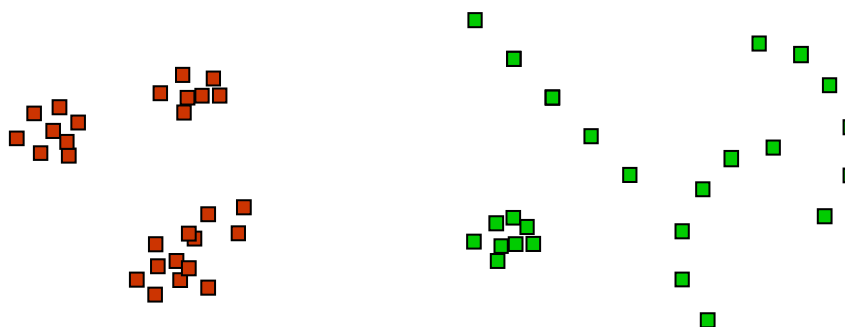
Efektywne metody grupowania danych

Plan wykładu



- Wstęp:
 - Motywacja i zastosowania
- Metody grupowania danych
 - Algorytmy oparte na podziałach (partitioning algorithms)
 - PAM
 - Ulepszanie: CLARA, CLARANS
 - Hierarchiczne algorytmy (hierarchical algorithms)
 - BIRCH
 - Algorytmy oparte na gęstościach (density-based algorithms)
 - DBSCAN

Motywacja i zastosowania



- **Problem:** wyznaczyć grupy podobnych obiektów na podstawie podanej z góry miary podobieństwa lub funkcji odległości
- **Cel:**
 - **Wyznaczyć grupy** obiektów podobnych
 - **Redukcja zbioru danych:** znaleźć obiekty reprezentatywne w „homogenicznych” grupach
 - **Odkrywanie nowych cech:** szukanie „naturalnych” grup obiektów. Dla każdej grupy wyznaczyć (nieznana) cechę opisującą obiekty w grupie
 - **Odkrywanie szumu (okłamywania) w danych:** Szukanie obiektów, które nie należą do żadnej grupy

Zastosowania



- WWW:
 - Klasyfikacja dokumentów
 - Grupowanie danych Weblog w celu odkrywaniu wzorców dostępu do sieci użytkownika
- Marketing:
 - Odkrywanie grup klientów o podobnych zachowaniach
- Przetwarzanie obrazów:
 - Kompresja obrazów
- Automatyczne uporządkowanie książek, dokumentów
- Odkrywanie oszust

Problem grupowania danych



- **Dany jest:**
 - Zbiór N obiektów, każdy jest określony wektorem wartości o takiej samej długości
 - Funkcja odległości (macierz odległości)
 - Funkcja oceny jakości grupowania
- **Problem:** Podzielić zbiór obiektów na grupy, które optymalizują funkcję jakości.

Główne metody grupowania



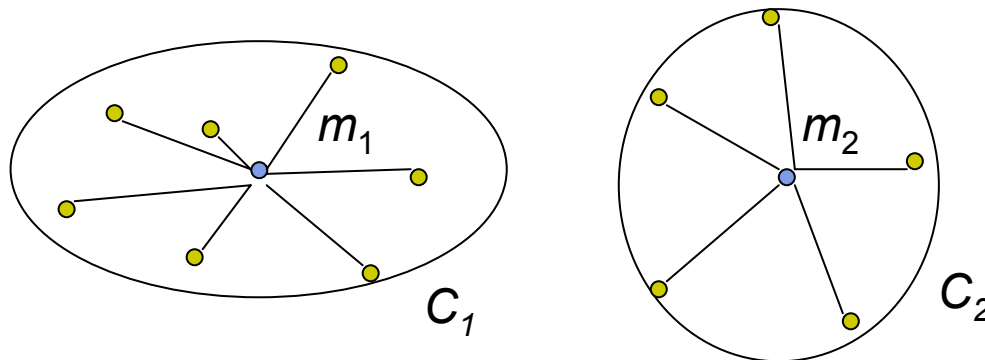
- Algorytmy oparte na podziałach (partitioning algorithms):
Konstrukcja różnych podziałów i potem ocenia się je za pomocą kryterium jakości
- Hierarchiczne algorytmy (hierarchical algorithms): Utworzy hierarchiczną dekompozycję zbioru danych
- Algorytmy oparte na gęstościach (density-based algorithms):
oparte na funkcji gęstości i na lokalnych połączeniach.

Algorytm oparty na podziałach- Problem optymalizacji



- Dane są zbiór obiektów D i parametr k . Podziel zbiór D na k grup takich, że optymalizują one wybrane kryterium optymalizacji.
- Suma kwadratów błędu (square error criterion):

$$E = \sum_{i=1}^k \sum_{p \in C_i} \text{dist}(p, m_i)^2$$



Algorytm oparty na podziałach – Znane algorytmy



- Algorytm optymalny
- Algorytmy aproksymacyjne:
 - *k-means* (MacQueen): Każda grupa jest reprezentowana przez środek ciężkości obiektów w grupie
 - *k-medoids*: Każda grupa jest reprezentowana przez jeden obiekt w grupie.
 - PAM (**P**artition **a**round **m**edoids) (Kaufman i Rouseeuw)
 - CLARA (**C**lustering **L**arge **A**pplications)
 - CLARANS (**C**lustering **L**arge **A**pplications based on **R**andomized **S**earch) (Raymont T. Ng i Jiawei Han)

Algorytm k – centroidów - Idea



Krok 1. Podziel zbiór danych na dowolnych k rozłącznych zbiorów

Krok 2. Dla każdej grupy wyznacz centroid. Centroid jest środkiem ciężkości grupy.

Krok 3. Podziel obiekty do najbliższego centroida.

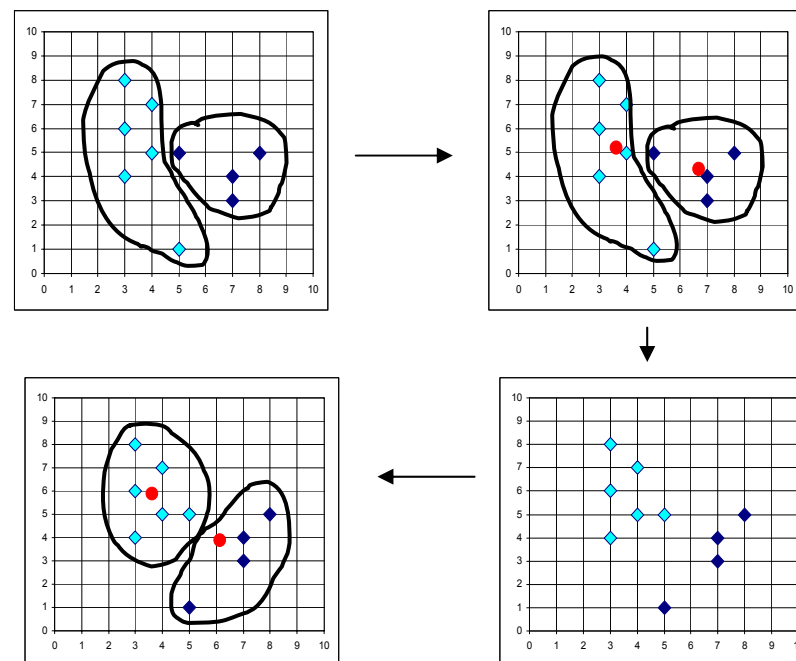
Krok 4. Jeśli grupy się nie zmieniają **stop**, wpp. **Goto** krok 2.

Zaleta:

- Efektywny: $O(tkn)$, gdzie n : liczba obiektów, k : liczba grup, t : liczba iteracji

Wada:

- Stosowalny tylko dla danych rzeczywistych
- Liczba grup k musi być podana
- Nie wykrywa szumu
- Nie pasuje, jeśli grupy nie są wypukłe



Algorytm k -medoidów - PAM



Cel: wyznaczyć k reprezentatywnych obiektów.

Krok 1. Wybierz dowolnie k reprezentatywnych obiektów

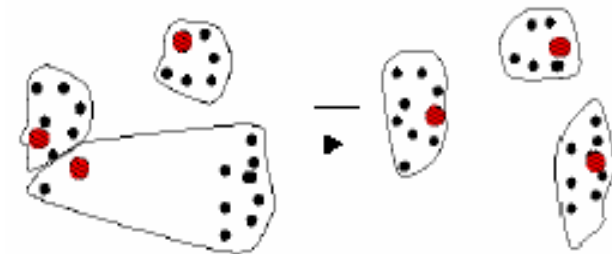
Krok 2. Dla każdej pary: (obiekt zwykły h ,
obiekt reprezentatywny i)
wyznacz całkowity koszt zamiany TC_{ih}

Krok 3. Dla pary (i, h) o najmniejszym koszcie TC_{ih}
zamień zwykły obiekt h
na medoid i i medoid i na zwykły obiekt,
jeśli $TC_{ih} < 0$

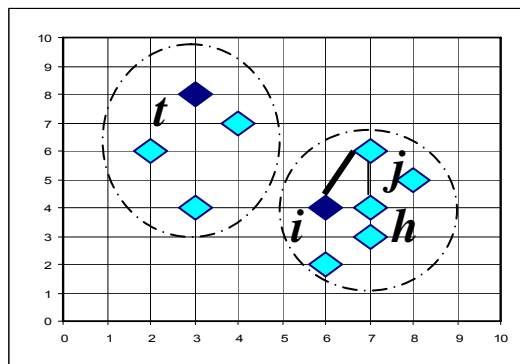
Krok 4. Podziel objekty do grup względem nowego
układu medoidów, goto **Krok 2**

Złożoność: $O(k(n-k)^2 \cdot t)$

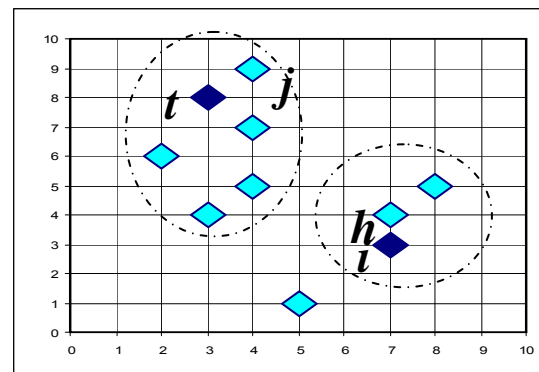
n -liczba obiektów, k -liczba grup, t -liczba iteracji



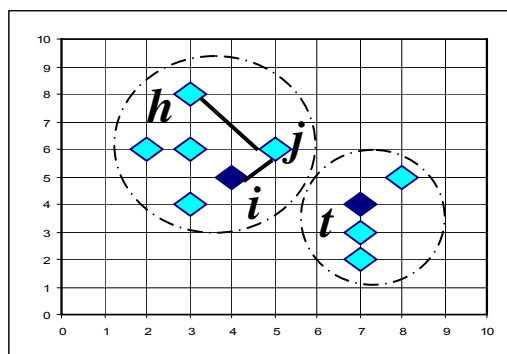
Wyznaczanie $TC_{ih} = \sum_j C_{jih}$



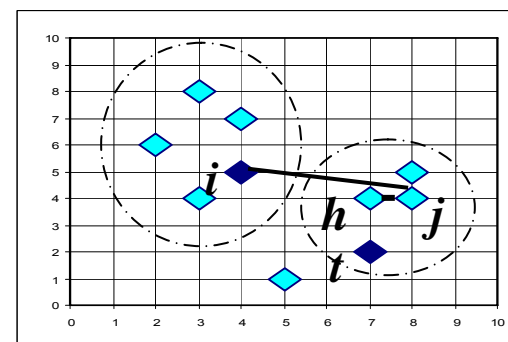
$$C_{jih} = d(j, h) - d(j, i)$$



$$C_{jih} = 0$$



$$C_{jih} = d(j, t) - d(j, i)$$



$$C_{jih} = d(j, h) - d(j, t)$$

Algorytm CLARA (Clustering Large Applications)



- **Algorytm CLARA**
 - Próbuj wiele razy zbiór danych
 - Dla każdej próbki szukaj obiektów reprezentatywnych (za pomocą *PAM*)
 - Zwracaj najlepszy zbiór *medoidów*
- **Zaleta:** stosowalna dla dużego zbioru danych
- **Wada:** jakość zależy od wyniku próbkowania
- **Modyfikacja CLARA:** Próbkiwanie za pomocą R^* -drzewa



Algorytm CLARANS (Randomized CLARA)

CLARANS (A Clustering Algorithm based on Randomized Search) (Ng and Han'94)

Interpretacja przestrzeni wyszukiwania:

- **Graf pełny $G_{n,k}$:**
 - wierzchołkami są zbiory k medoidów
 - Dwa wierzchołki są połączone, jeśli odpowiednie zbiory medoidów mają $(k-1)$ wspólnych elementów
- Zmiana jednego medoidu odpowiada przejściu z jednego wężła do wężła sąsiedniego.

Algorytm CLARANS (Randomized CLARA) (c.d.)



- **Algorytm PAM:** przeszukiwanie całego grafu
- **Algorytm CLARANS:** ma dwa parametry *max_neighbor* i *num_local*
 - Startuje od dowolnego wężła
 - Dla bieżącego wężła *t*, szukaj lepszego wężła wśród (*max_neighbor*) sąsiadów *t* i przejdź do lepszego sąsiada
 - Powtórz proces wyszukiwania aż liczba cykli osiągnie *num_local*
 - Jeśli lokalne minimum jest osiągalne, to algorytm losuje dowolny nowy wężel i szuka innego minimum lokalnego zaczynając od tego wężła.

Algorytm CLARANS



Wejście: $max_neighbor$, num_local , $G_{n,k}$

Wyjście: zbiór k mediodów

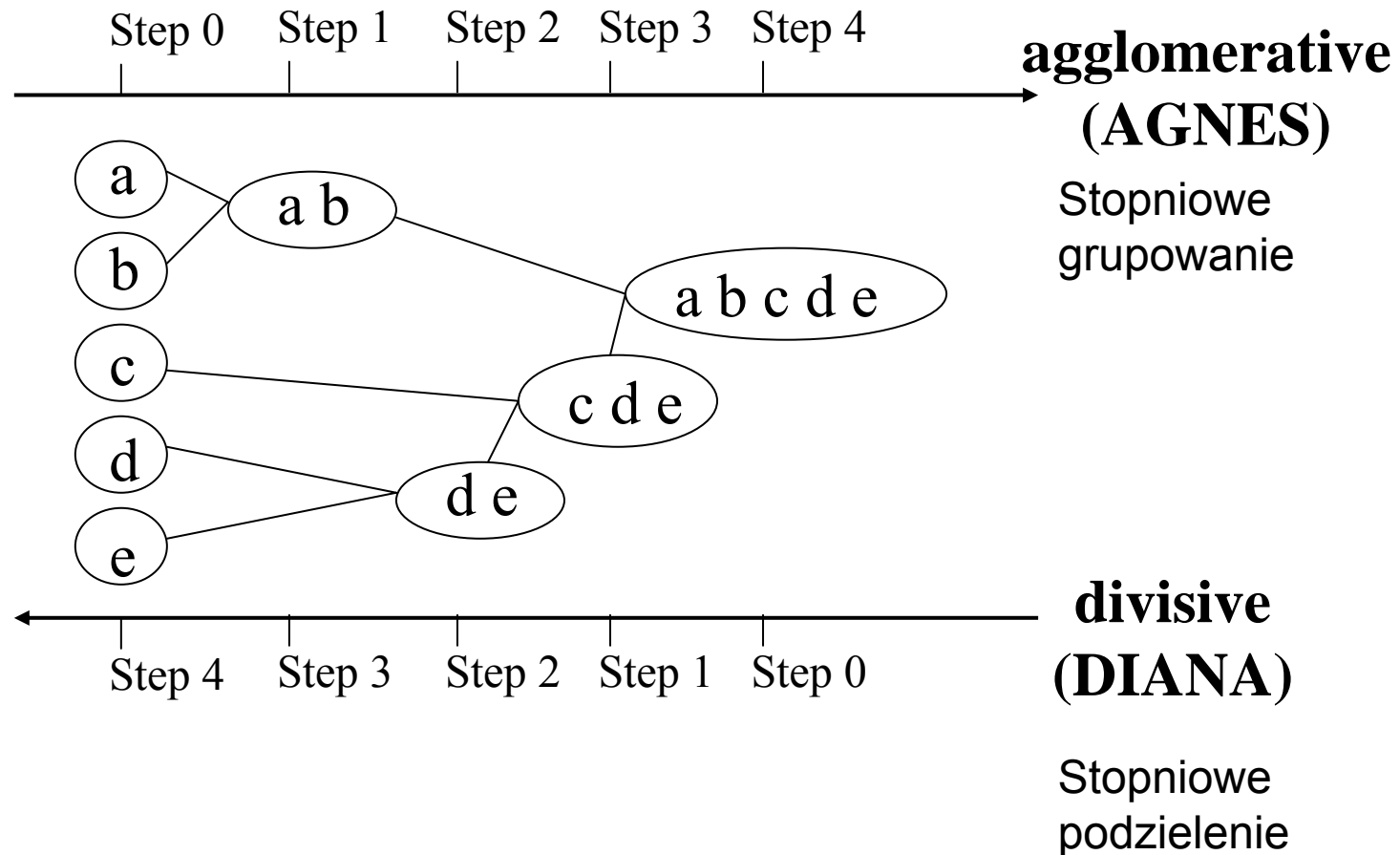
Krok 1. $current$ = dowolny węzeł w grafie $G_{n,k}$;
 $j = 1$; $best_node = current$;

Krok 2. Wybierz losowego sąsiada s i oceń jakość s (za pomocą funkcję TC)

Krok 3. **if** (sąsiad s jest lepszy niż $current$) **then** $current = s$,
else $j = j + 1$;
if ($j < max_neighbor$) **goto** **Krok 2**
else if ($current$ jest lepszy niż $best_node$) **then**
 $best_node = current$;

Krok 4. $i = i + 1$;
if ($i > num_local$) **return** $best_node$;
else goto **Krok 1**

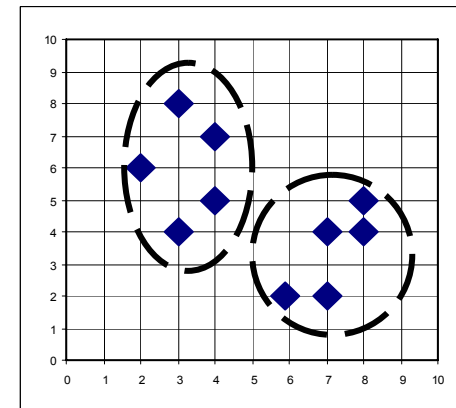
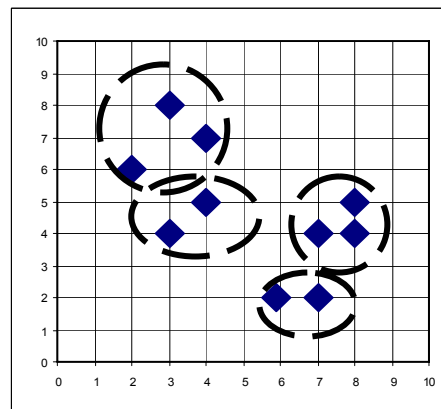
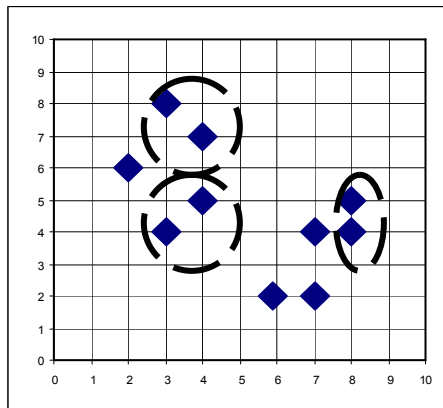
Hierarchiczne grupowanie



AGNES (Agglomerative Nesting)



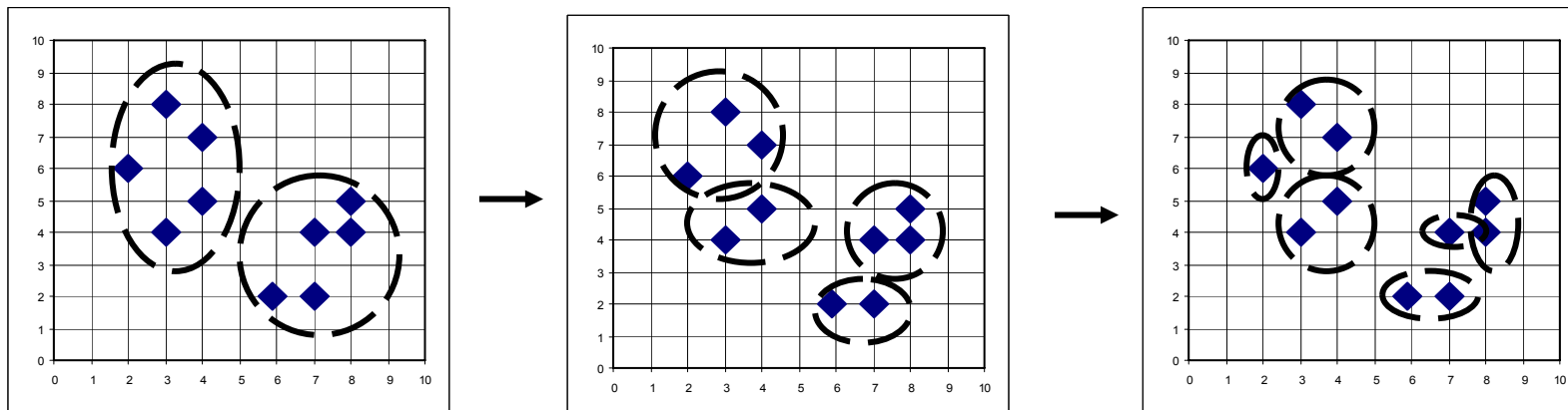
- Algorytm **bottom-up**
- Każdy obiekt należy do jednej grupy (do jednego liścia)
- Połącz węzły, które są bardziej podobne (odległość najmniejsza)
- Kontynuuj, aż wszystkie obiekty należą do jednej grupy (do korzenia)



DIANA (Divisive Analysis)



- Algorytm **top-down**
- Podziel węzeł na dwa węzły
- Kontynuuj, aż każdy obiekt znajduje się w jednym liściu



Hierarchiczne grupowanie- Obserwacja



- **Zalety:**
 - Liczba grup k nie musi być znana
 - Wielkość grupy można dobrać
- **Wady:**
 - Nie jest efektywne: $O(n^2)$, gdzie n jest liczbą obiektów
 - Struktura jest statyczna

Ulepszony algorytm hierarchiczny- BIRCH



- **BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies):**
 - Działa efektywnie: decyzja dla jednej grupy (dzielenie czy połączenie z inną grupą) nie wymaga przeglądania całego zbioru danych
 - I/O koszt jest liniowy względem rozmiaru danych: przeglądanie zbioru danych raz
 - Algorytm działa dla danych dynamicznie zmienionych
 - Wykrywa szumy w danych

BIRCH – Struktura CF drzewa



- CF (**C**lustering **F**eature) – drzewo:
 - Zrównoważone drzewo
 - Ma trzy parametry:
 - B – maksymalna liczba rozgałęzień,
 - L – maksymalna liczba obiektów w liściach
 - T – maksymalny promień (grup w liściach)
 - Węzeł wewnętrzny: $[CF_i, child_i]$ $i= 1,2,\dots,B$
 - Węzeł zewnętrzny (liść): $[CF_i]$ $i=1,2,\dots,L$

Opis grupy



- Niech grupa G zawiera N punktów $\vec{x}_i \in R^d$
- **Środek**, **promień** (R) i **średnica** (D) grupy są zdefiniowane:

$$\vec{x}_0 = \frac{\sum_{i=1}^N \vec{x}_i}{N}$$

$$R = \left(\frac{\sum_{i=1}^N (\vec{x}_i - \vec{x}_0)^2}{N} \right)^{\frac{1}{2}}$$

$$D = \left(\frac{\sum_{i=1}^N \sum_{j=1}^N (\vec{x}_i - \vec{x}_j)^2}{N(N-1)} \right)^{\frac{1}{2}}$$

- Parametry \vec{x}_0 , R i D opisują grupę obiektów G .



Opis grupy – Wektor CF

$$CF = (5, (16,30), (54,190))$$

Opis grupy : $CF = (N, \vec{LS}, SS)$

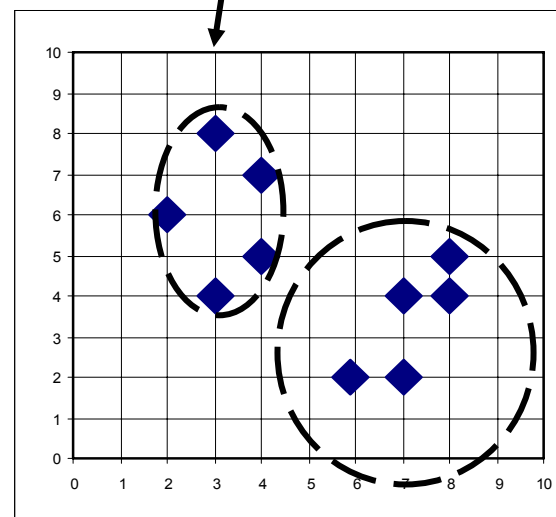
N : liczba punktów w grupie

$$\vec{LS}: \sum_{i=1}^N \vec{X}_i$$

$$SS: \sum_{i=1}^N X_i^2$$

Twierdzenie:

Niech $CF_1 = (N_1, \vec{LS}_1, SS_1)$ i $CF_2 = (N_2, \vec{LS}_2, SS_2)$
będą opisami dwóch grup G_1 i G_2 , to $CF = (N_1+N_2,$
 $\vec{LS}_1 + \vec{LS}_2, SS_1+SS_2)$ będzie opisem grupy, która jest
połączeniem G_1 i G_2



(3,4)

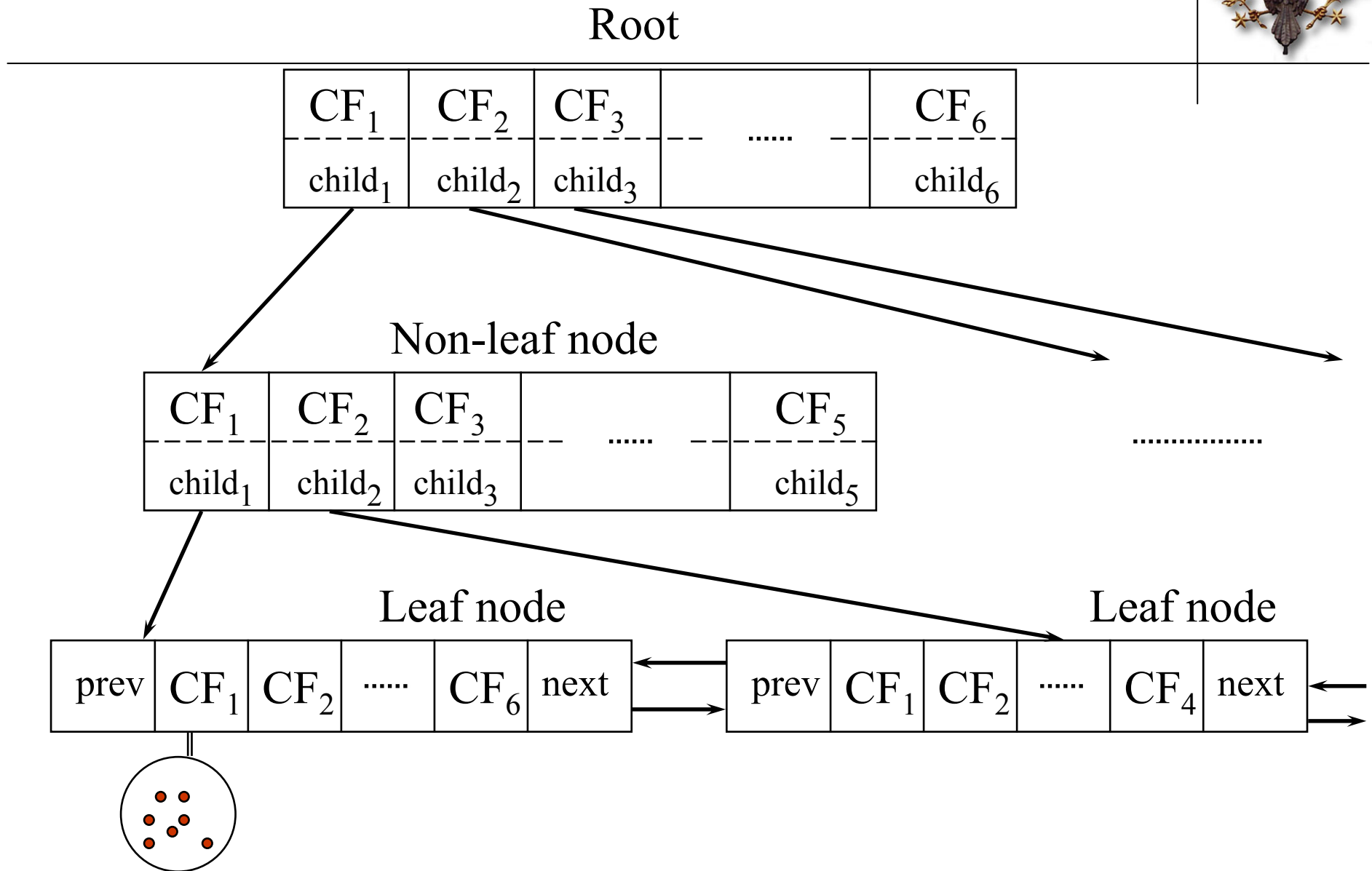
(2,6)

(4,5)

(4,7)

(3,8)

CF - drzewo



Wstawianie obiektu do drzewa



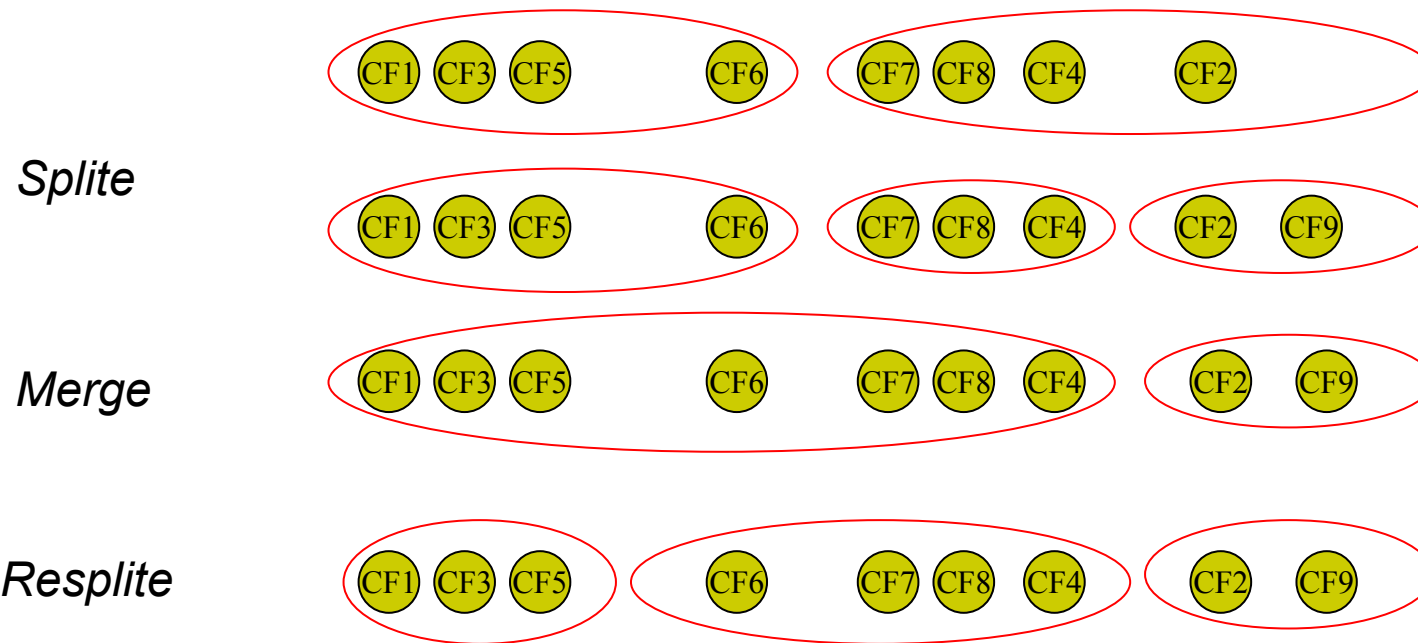
Krok 1. Wybierz liść l do wstawiania. Użyj jednej z funkcji odległości D do wyznaczenia najbliższej grupy do badanego punktu

Krok 2. Jeśli w l jest miejsce to wstaw x ,

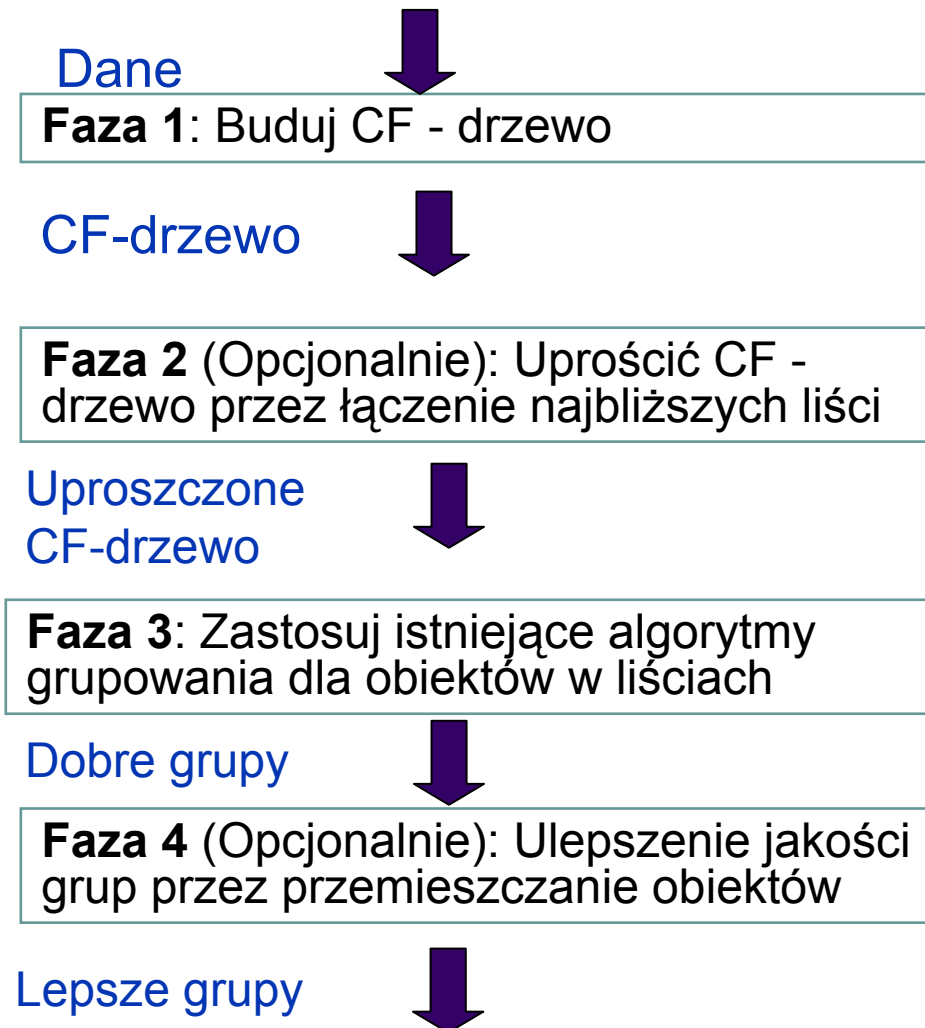
wpp. Podziel liść l na dwa liście i popraw ścieżkę od l do korzenia.

Krok 3. Rekonstrukcja drzewo przez połączenie dwa najbliższe węzły i podzielić na dwa (w razie potrzeby): *merge i resplite*

Efekt *splitte*, *merge* i *resplite*



Algorytm BIRCH – Schemat blokowy



Algorytm BIRCH –Faza 3,4



- **Faza 3:**
 - Każda grupa w liściu jest reprezentowana przez środek ciężkości. Zastosuj dowolny algorytm grupowania dla zbioru środków
 - Zastosuj dowolny algorytm grupowania bezpośrednio na obiektach w grupie.
- **Faza 4 (ulepszenie jakości grup):**
 - Wyznaczaj środki grup generowanych przez fazę 3
 - Dla każdego obiektu o , przemieszczaj go do grupy, której środek jest najbliżej o .

BIRCH - Ocena



- **Zalety:**

- Wyznacza grupy przez jedno przeglądanie zbiorów danych.
- Proces wstępny dla wielu algorytmów grupowania

- **Wady:**

- Działa tylko dla danych numerycznych
- Wrażliwy na kolejność obiektów

Ograniczenie algorytmu grupowania oparte na odległości



- Każda grupa jest reprezentowana przez jeden obiekt lub środek ciężkości
- Grupy są wypukłymi figurami.

Grupowanie oparte na gęstości



- Grupa składa się z punktów sąsiednich o wysokiej gęstości w otoczeniu
- Regiony pokrywające grupy mają wyższą gęstość niż regiony na zewnątrz

Grupowanie oparte na gęstości

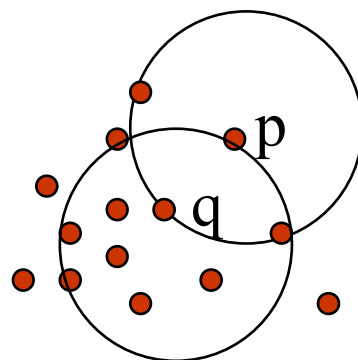


- Główne zalety:
 - Odkrywa grupy o dowolnym kształcie
 - Odkrywa szumy (okłamywanie)
 - Jedno przeglądanie zbioru danych
- Interesujące algorytmy:
 - DBSCAN: Ester, et al. (KDD'96)
 - OPTICS: Ankerst, et al (SIGMOD'99).
 - DENCLUE: Hinneburg & D. Keim (KDD'98)
 - CLIQUE: Agrawal, et al. (SIGMOD'98)

Pojęcia podstawowe



- Dwa parametry:
 - ε : promień definiujący otoczenie obiektu
 - **MinPts**: minimalna liczba punktów w ε -otoczeniu
- **Rdzeń**: obiekt, który ma co najmniej MinPts w ε -otoczeniu
- **Brzegowy obiekt**: obiekt, który ma mniej niż *MinPts* w ε -otoczeniu.



$$\text{MinPts} = 5$$

$$\varepsilon = 1 \text{ cm}$$

Pojęcia podstawowe (c.d.)



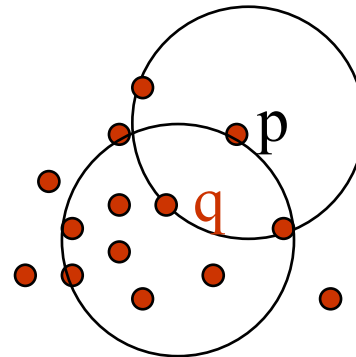
- ε -otoczenie:

$$N_\varepsilon(p): \{q \in D \mid \text{dist}(p,q) \leq \varepsilon\}$$

- Dane są parametry ε i $MinPts$. Punkt p jest bezpośrednio wyprowadzony z punktu q jeśli

1) $p \in N_\varepsilon(q)$

2) $|N_\varepsilon(q)| \geq MinPts$



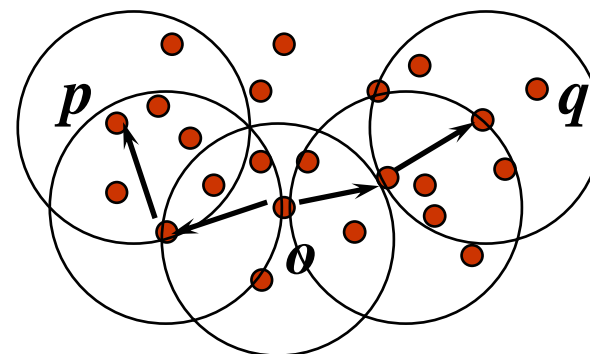
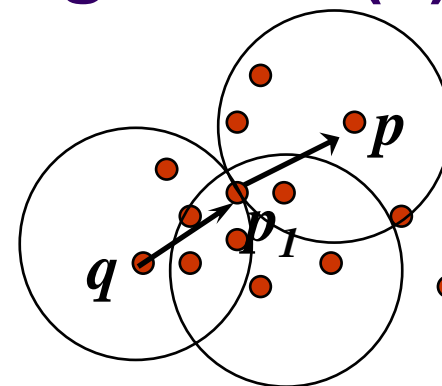
$$MinPts = 5$$

$$\varepsilon = 1 \text{ cm}$$



Density-Based Clustering: Background (II)

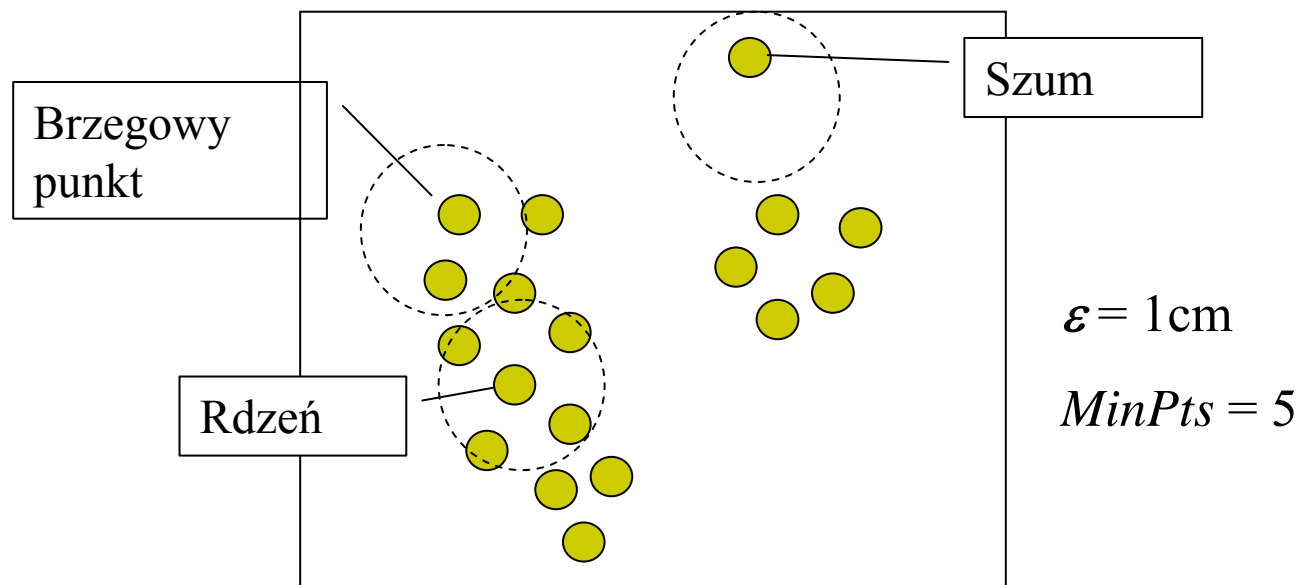
- Punkt p jest wyprowadzony z punktu q jeśli istnieje ciąg punktów p_1, \dots, p_n taki, że $p_1 = q$, $p_n = p$ i p_{i+1} jest bezpośrednio osiągalny z p_i
- Punkt p i q są połączone jeśli istnieje punkt o taki, że p i q są wyprowadzone z o



DBSCAN: Density Based Spatial Clustering of Applications with Noise



- **Grupa:** Maksymalny zbiór punktów połączonych



Algorytm DBSCAN



Krok 1. Wybierz dowolny punkt p

Krok 2. Wyszukiwać zbiór G wszystkich punktów osiągalnych z punktu p w sensie ε i $MinPts$.

Krok 3. Jeśli p jest rdzeniem, **return** G (grupa była utworzona) .

Krok 4. jeśli p jest punktem brzegowym (żaden punkt nie jest osiągalny z p) to sprawdź następny nieodwiedzony punkt

Krok 5. Kontynuuj aż wszystkie punkty są odwiedzone

Ulepszanie



- Użycie R^* -drzewa do indeksowania zbioru danych
- Wyznaczanie sąsiadów w ε -otoczeniu danego punktu p :
 - Przeglądaj drzewo od korzenia do liścia
 - Dla węzła t , wybierz poddrzewo, którego prostokąt ma niepuste przecięcie z okręgiem o środku w p i promieniu ε .

Złożoność czasowa	Czas wyszukiwania sąsiadów	Złożoność DBSCAN
Bez indeksowania	$O(n)$	$O(n^2)$
R^* -drzewo	$O(\log n)$	$O(n \cdot \log n)$