Grupowanie

1 k-means

1. Losowo wybieramy k centroidów.
2. Powtarzamy, dopóki żaden przykład nie zmieni grupy w dwóch kolejnych iteracjach:
   (a) Dla każdego przykładu znajdujemy najbliższy centroid i przypisujemy go do jego grupy.
   (b) Dla każdej grupy wyliczamy nowy centroid, będący uśrednieniem wszystkich punktów z grupy.

2 Hierarchiczne grupowanie aglomeracyjne

1. Zaczynamy z każdym przykładem w swojej własnej grupie.
2. Identyfikujemy dwie najbliższe grupy zgodnie z przyjętą metryką i łączymy je ze sobą. Licząc dystans między grupami więcej niż jednym elementem, możemy np. porównać ich centroidy.
3. Powtarzamy krok 2, aż wszystkie przykłady będą w jednej grupie.

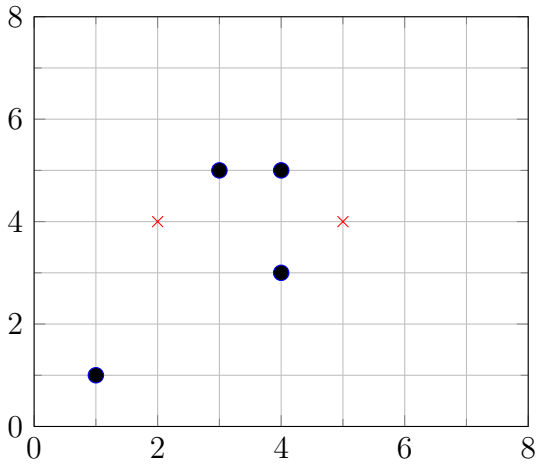
Zadania

Zadanie 1.

Dane są następujące wektory:

- A(1, 1),
- B(3, 5),
- C(4, 3),
- D(4, 5).

Zaczynając z centroidami $c_1(2, 4)$ i $c_2(5, 4)$, grupuj wektory algorytmem k-means.



Zadanie 2.

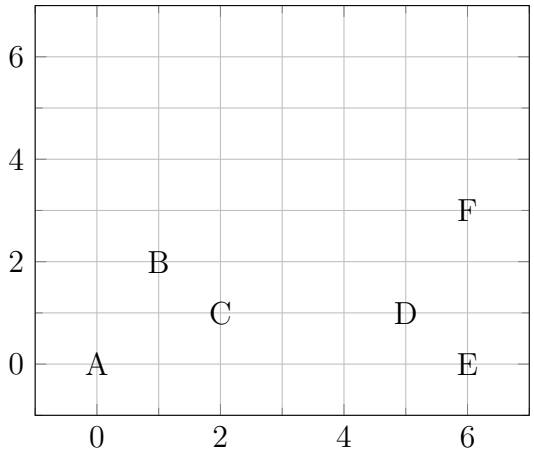
Grupuj następujący zbiór algorytmem k-means, zaczynając z centroidami $c_1(0, 1, 0, 1)$, $c_2(0, 2, 1, 0)$:

- A(0, 1, 0, 1),
- B(0, 0, 2, 0),
- C(3, 2, -1, 1),
- D(0, 2, 1, 0).

Zadanie 3.

Grupuj następujące wektory metodą hierarchicznego grupowania aglomeracyjnego, a następnie narysuj dendrogram:

- A(0,0),
- B(1,2),
- C(2,1),
- D(5,1),
- E(6,0),
- F(6,3).



Mini-projekt: k-means

Implementuj algorytm k-means. Grupuj dane z pliku `iris.data` (podczas grupowania ignorujemy atrybut decyzyjny).

Program powinien:

- Umożliwiać wybór k.
- Po każdej iteracji: wypisywać sumę odległości przykładów od ich centroidów. Ta wartość powinna zmniejszać się z każdą iteracją. Uwaga: wypisujemy sumę dla wszystkich przykładów, a nie każdej grupy osobno.
- Na końcu: wyświetlać składy grup.
- Dodatkowo (opcjonalnie): wyświetlać miary czystości grup, np. procentowe zawartości każdej z klas Iris, lub entropię.