Matematyczne aspekty analizy danych
semestr zimowy 2025/2026

Dr Anna Muranova
UWM w Olsztynie

Wykład 14

Rozdział 12. Analiza skupień

Wprowadzenie do analizy skupień

Grupowanie (analiza skupień, klasteryzacja) (ang. data clustering) – metoda tzw. klasyfikacji bez nadzoru (ang. unsupervised learning). Jest to metoda dokonująca grupowania elementów według jednorodnych klasów. Podstawą grupowania w większości algorytmów jest podobieństwo pomiędzy elementami – wyrażone przy pomocy funkcji (metryki) podobieństwa.
Poprzez grupowanie można również rozwiązać problemy z gatunku odkrywania struktury w danych oraz dokonywanie uogólniania. Grupowanie polega na wyodrębnianiu grup (klasów, podzbiorów).

Wybrane cele dokonywania grupowania są następujące:

- uzyskanie jednorodnych przedmiotów badania, ułatwiających wyodrębnienie ich zasadniczych cech,
- zredukowanie dużej liczby danych pierwotnych do kilku podstawowych kategorii, które mogą być traktowane jako przedmioty dalszej analizy,
- zmniejszenie nakładu pracy i czasu analiz, których przedmiotem będzie uzyskanie klasyfikacji obiektów typowych,
- odkrycie nieznanej struktury analizowanych danych, porównywanie obiektów wielocechowych.

Grupowanie jako jedna z metod pozyskiwania wiedzy, a tym samym eksploracji danych, jest ściśle uwarunkowana źródłem danych oraz oczekiwaną postacią rezultatów.

Algorytmy analizy skupień dzieli się na kilka podstawowych kategorii:

- metody hierarchiczne – algorytm tworzy dla zbioru obiektów hierarchię klasyfikacji, zaczynając od takiego podziału, w którym każdy obiekt stanowi samodzielne skupienie, a kończąc na podziale, w którym wszystkie obiekty należą do jednego skupienia.
- grupa metod *k*-średnich (ang. k-means), w której grupowanie polega na wstępnym podzieleniu populacji na z góry założoną liczbę klas (tzw. skupień). Następnie uzyskany podział jest poprawiany w ten sposób, że niektóre elementy są przenoszone do innych klas, tak, aby uzyskać minimalną wariancję wewnątrz każdej z nich – dąży się do zapewnienia jak największego podobieństwa elementów w ramach każdego ze skupień, przy jednoczesnej maksymalnej różnicy pomiędzy samymi klasami (skupieniami).
- metody rozmytej analizy skupień (ang. fuzzy clustering). Metody rozmytej analizy skupień mogą przydzielać element do więcej niż jednej kategorii. Z tego powodu algorytmy rozmytej analizy skupień są stosowane w zadaniu kategoryzacji (przydziału jednostek do jednej lub wielu kategorii).

Analiza skupień używa się nie tylko w eksploracji danych, ale tez ma inne zastosowania:

- wstępna analiza danych, polegająca na wyodrębnieniu jednorodnych grup (subpopulacji), które podlegają osobnej dalszej analizie statystycznej lub ekonometrycznej;
- wyszukiwanie informacji (ang. information retrieval), mająca za zadanie uporządkowanie i uproszczenie dostępu do informacji. Do klasycznych zastosowań należy klasyfikacja dokumentów tekstowych: książek, czy stron internetowych;
  - np.: w wyszukiwarkach internetowych – wykorzystywane do automatycznego wyodrębnienia jednolitych grup i przydziału elementów wyników do poszczególnych z nich;
- segmentacja obrazu (ang. image segmentation), czyli podział obrazu na regiony homogeniczne pod względem pewnej własności obrazu (kolor, tekstura, intensywność). Taki uproszczony obraz jest prostszy do obróbki np. przez algorytmy rozpoznawania obrazu;
- grupowanie zadań w problemie harmonogramowania tak, by zadania intensywnie ze sobą komunikujące się trafiły do tej samej grupy. Taka grupa zostanie w następnym kroku przypisana do wykonania na jednym procesorze (bądź kilku procesorach połączonych szybkimi kanałami komunikacyjnymi).

Grupowanie hierarchiczne

Grupowanie hierarchiczne (hierarchiczna analiza skupień, klasteryzacja hierarchiczna, klastrowanie hierarchiczne) – metoda analizy skupień, która ma na celu zbudowanie hierarchii klastrów. Służy do dzielenia obserwacji na grupy (klastry) bazując na podobieństwach między nimi. W przeciwieństwie do wielu algorytmów służących do klastrowania w tym wypadku nie jest konieczne wstępne określenie liczby tworzonych klastrów. Strategie tworzenia klastrów hierarchicznych dzielą się zasadniczo na dwa typy:

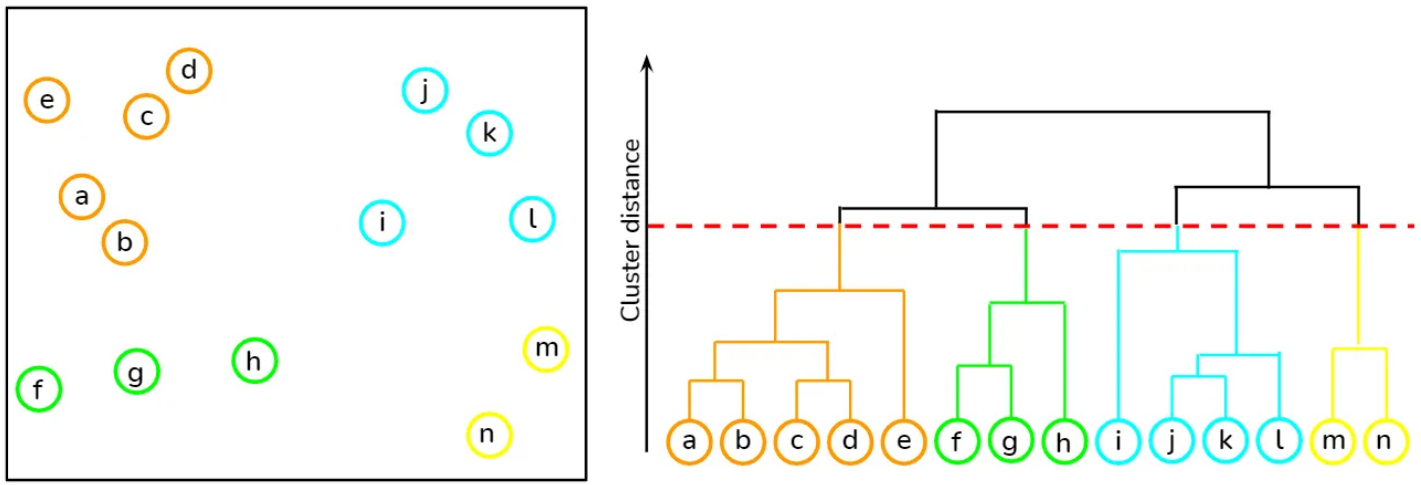
- metody aglomeracyjne (ang. agglomerative) – każda obserwacja tworzy na początku jednoelementowy klaster. Następnie pary klastrów są scalane, w każdej iteracji algorytmu łączone są z sobą dwa najbardziej zbliżone klastry. Tworzone są tak zwane aglomeracje. W tym typie podczas tworzenia klastrów idzie się w górę hierarchii.
- metody deglomeracyjne (ang. divisive) – początkowo wszystkie obserwacje znajdują się w jednym klastrze. W następnych krokach klastry dzielone są na mniejsze i bardziej jednorodne. Podziały wykonywane są rekursywnie. W czasie tworzenia klastrów idzie się w dół hierarchii.

Algorytmy grupowania hierarchicznego charakteryzują się złożonością obliczeniową $O(n^3)$ oraz wymagają $O(n^2)$ pamięci, co czyni je mało efektywnymi. Wyniki hierarchicznego grupowania stanowią zestaw zagnieżdżonych klastrów, które są zwykle prezentowane w dendrogramie.

Dendrogram

Definition

Dendrogram – diagram w kształcie drzewa ukazujący związki pomiędzy wybranymi elementami na podstawie przyjętego kryterium.



Rysunek: Dane i dendrogram. Źródło: `https://towardsdatascience.com/hierarchical-clustering-explained-e59b13846da8`

Odległość

Stosując algorytmy grupowania hierarchicznego, konieczne jest dokonanie pomiaru odległości między punktami. Głównym celem jest to, aby odległości między obserwacjami tego samego klastra były możliwie jak najmniejsze, natomiast odległości między klastrami były jak największe. W hierarchicznym grupowaniu istnieją dwa bardzo ważne parametry: metryka odległości i metoda połączenia.
Zdefiniowanie sposobu określania odległości między obserwacjami jest jednym z najważniejszych aspektów tego algorytmu. Może zdarzyć się, że niedostosowanie odpowiedniej metryki do danych spowoduje otrzymanie bezsensownych wyników klastrowania.

Główne miary odległości są następujące:

- Odległość Euklidesowa: $\|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
- Kwadratowa odległość Euklidesowa: $\|a - b\|_2^2 = \sum_i (a_i - b_i)^2$
- Odległość Manhattan: $\|a - b\|_1 = \sum_i |a_i - b_i|$
- Maksymalna odległość : $\|a - b\|_\infty = \max_i |a_i - b_i|$

Metody połączenia

Metody połączenia określają, jak definiowana jest odległość między dwoma klastrami. Ważne jest, aby w danym eksperymencie wypróbować kilka metod łączenia oraz porównać ich wyniki. W zależności od zbioru danych, niektóre metody mogą działać lepiej. Poniżej znajduje się lista najczęściej występujących metod połączeni (dla klastrów A i B):

Metody połączenia 1

- Pojedyncze połączenie – odległość między dwoma klastrami jest minimalną odległością między obserwacją w jednym klastrze a obserwacją w innym klastrze. Sprawdza się, gdy klastry są wyraźnie oddzielone:

$$\min_{a \in A,\, b \in B} d(a, b)$$

.

- Kompletne połączenie – odległość między dwoma klastrami jest maksymalną odległością między obserwacją w jednym klastrze a obserwacją w innym klastrze. Może być wrażliwy na występowanie outlier'ów.

$$\max_{a \in A,\, b \in B} d(a, b).$$

- Średnie połączenie – odległość między dwoma klastrami jest średnią odległością między obserwacją w jednym klastrze a obserwacją w innym klastrze:

$$\frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b).,$$

- Min-max połączenie – odległość między dwoma klastrami jest:

$$\min_{x \in A \cup B} \max_{y \in A \cup B} d(x, y)$$

Metody połączenia 2

- Połączenie centroidalne – odległość między dwoma klastrami jest odległością pomiędzy centroidami klastra:

$$\|\mu_A - \mu_B\|^2,$$

gdzie μ_A oraz μ_B są centroidami klasterów A i B (średnią arytmetyczną pozycji wszystkich elementów odpowiedniego klastra)

- Połączenie Ward'a – odległość między dwoma klastrami jest sumą kwadratów odchyleń od punktów do centroidów. Ten sposób dąży do zminimalizowania sumy kwadratów wewnątrz klastra:

$$\frac{|A| \cdot |B|}{|A \cup B|}\|\mu_A - \mu_B\|^2 = \sum_{x \in A \cup B}\|x - \mu_{A \cup B}\|^2 - \sum_{x \in A}\|x - \mu_A\|^2 - \sum_{x \in B}\|x - \mu_B\|^2$$

- Połączenie według minimalnego błędu sumy kwadratów – odległość między dwoma klastrami jest sumą kwadratów odchyleń od punktów do wspólnego centroida:

$$\sum_{x \in A \cup B}\|x - \mu_{A \cup B}\|^2$$

Metody połączenia 3

- Połączenie według minimalnego wzrost wariancje – odległość między dwoma klastrami jest różnica pomiędzy wspólną wariancje a wariancjami klastrów:

$$\frac{1}{|A \cup B|} \sum_{x \in A \cup B} \|x - \mu_{A \cup B}\|^2 - \frac{1}{|A|} \sum_{x \in A} \|x - \mu_A\|^2 - \frac{1}{|B|} \sum_{x \in B} \|x - \mu_B\|^2$$
$$= \mathrm{Var}(A \cup B) - \mathrm{Var}(A) - \mathrm{Var}(B)$$

- Połączenie według minimalnej wariancji – odległość między dwoma klastrami jest wspólną wariancją:

$$\frac{1}{|A \cup B|} \sum_{x \in A \cup B} \|x - \mu_{A \cup B}\|^2 = \mathrm{Var}(A \cup B)$$

Metoda aglomeracyjna

Aglomeracyjne grupowanie działa w sposób „oddolny” (ang. bottom-up). Na początku algorytmu każda obserwacja jest traktowana jako pojedynczy klaster. Następnie pary klastrów są sukcesywnie łączone, aż do momentu gdy wszystkie klastry zostaną scalone w jeden duży klaster zawierający wszystkie obiekty.

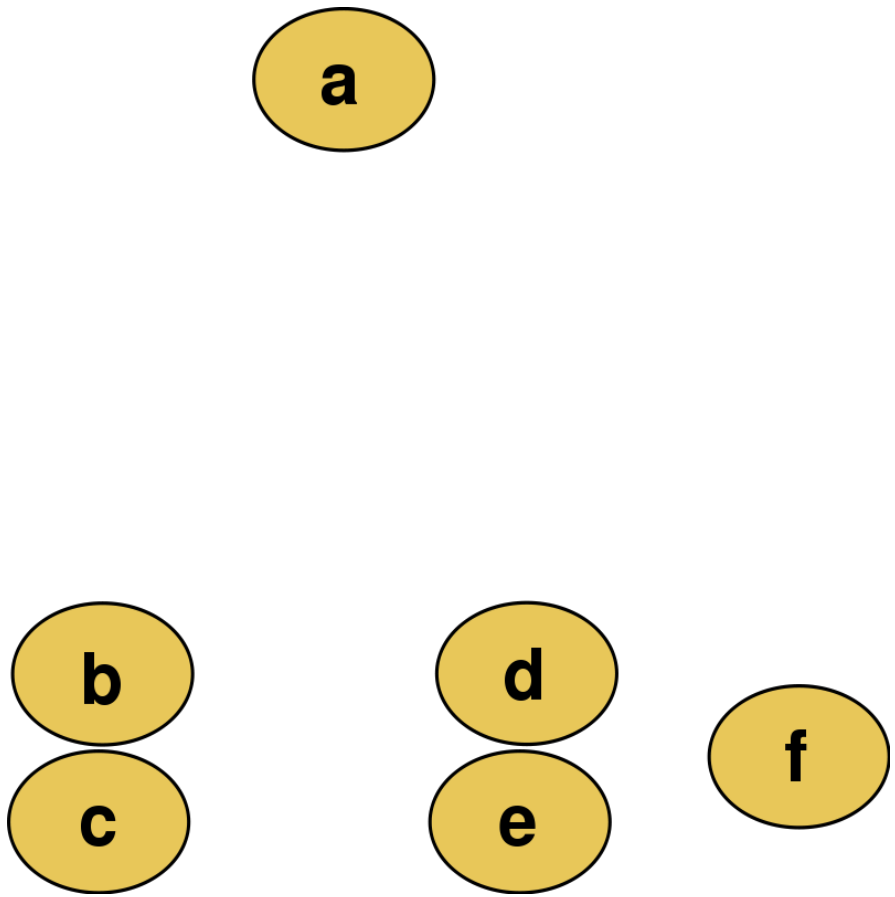
Przebieg algorytmu metody aglomeracyjnej można przedstawić następująco:

1. Wyszukiwane są dwa najbliższe punkty w zbiorze danych
2. Znalezione punkty są łączone – od tego momentu będą traktowane jako jeden punkt
3. Proces rozpoczyna się od nowa. Od teraz wykorzystywany jest nowy zbiór obserwacji utworzony w poprzednich krokach

Decyzja o połączeniu dwóch klastrów w jedne jest nieodwracalna – tego klastra nie można rozdzielić już w następnej iteracji algorytmu.

Przykład

(Źródło przykładu i obrazków: Wikipedia `https://pl.wikipedia.org/wiki/Grupowanie_hierarchiczne`) Na rysunkach przedstawiono przebieg procesu grupowania aglomeracyjnego. Na rysunku 2 jest zbiór danych, który zostanie poddany klasteryzacji.



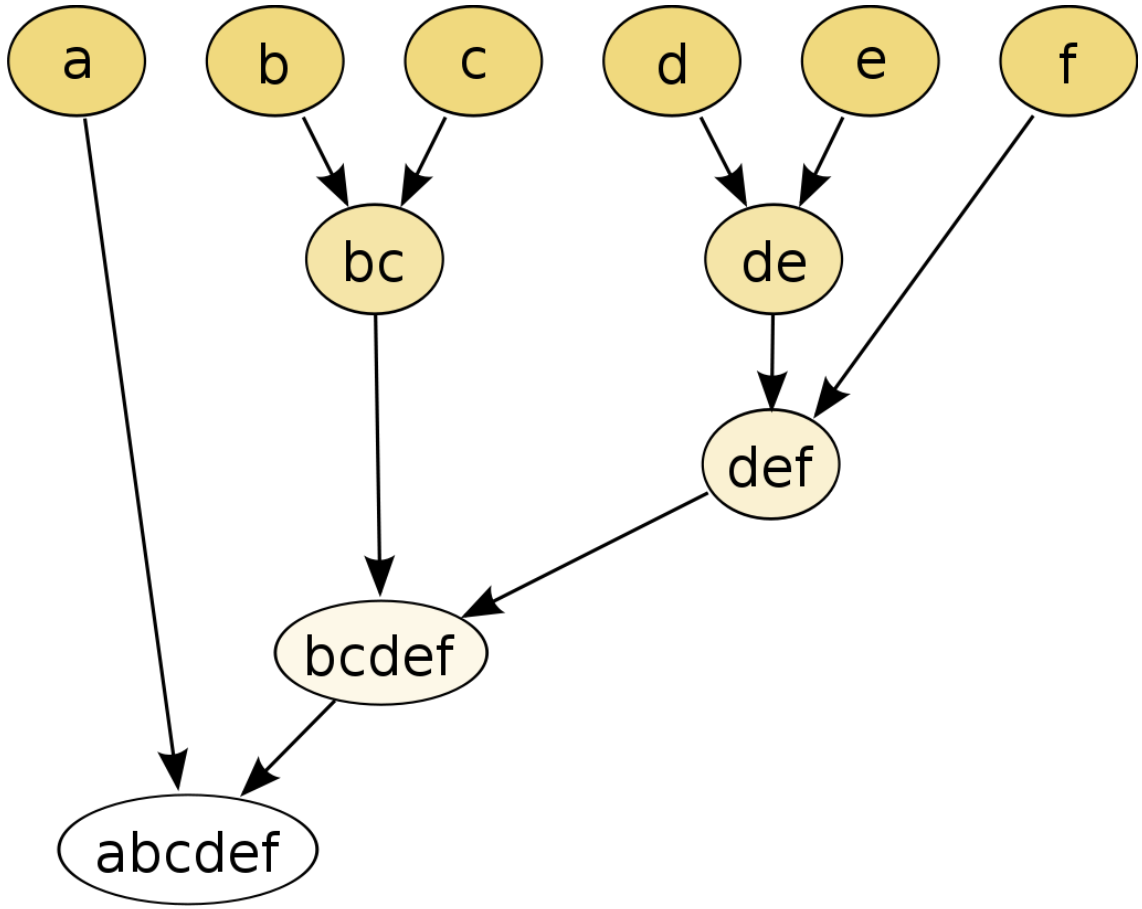
Rysunek: Przykładowe dane do klasteryzacji.

Przykład

Takim czynem, mamy sześć elementów a, b, c, d, e, f. Pierwszym krokiem działania algorytmu jest określenie, które elementy należy połączyć do jednego w klastra. Zwykle chcemy wziąć dwa najbliższe sobie elementy, zgodnie z wybraną odległością. W przykładzie wybór podejmowany jest na podstawie odległości Euklidesowej, najbardziej intuicyjnej. Jedną z możliwości porównywania odległości między sobą jest zbudowanie macierzy odległości na tym etapie, gdzie liczba w i-tym wierszu j-tej kolumny jest odległością między i-tym i j-tym elementem. Następnie, w miarę rozwoju klastrów, wiersze i kolumny macierzy są scalane, w momencie gdy scalane są klastry. Odległości powinny być wtedy aktualizowane.
Łatwo zauważyć, że najbliżej siebie znajdują się elementy b i c oraz d i e. W pierwszej fazie działania algorytmu połączyliśmy dwa najbliższe elementy b i c. Od teraz mamy następujące klastry $\{a\}$, $\{b, c\}$, $\{d\}$, $\{e\}$ i $\{f\}$. Naszym celem jest ich dalsze scalanie. Aby to zrobić, musimy określić odległość między klastrem $\{b, c\}$, a pozostałymi klastrami (które stanowią na razie pojedyncze obserwacje). Następnym etapem będzie połączenie elementów $\{d\}$ i $\{e\}$. Kolejne iteracje będą wykonywane aż do momentu gdy wszystkie sześć elementów znajdzie się w jednym klastrze.

Przykład

Na rysunku 3 przedstawiono dendrogram, który przedstawia rezultat działania algorytmu grupowania hierarchicznego.



Rysunek: Dendrogram po klasteryzacji.

Metoda deglomeracyjna

Podstawową zasadę klasteryzacji deglomeracyjnej opublikowano jako algorytm DIANA (ang. DIvisive ANAlysis clustering, [**?**]). DIANA wybiera obiekt o maksymalnej średniej odmienności, a następnie przenosi do tego skupienia wszystkie obiekty, które są bardziej podobne do nowego skupienia niż do pozostałych.
Nieformalnie DIANA to nie tyle proces „dzieleni", co „wydrążania": w każdej iteracji wybierany jest istniejący klaster (np. początkowy klaster całego zbioru danych), aby utworzyć w nim nowy klaster. Obiekty stopniowo przenoszą się do tego zagnieżdżonego klastra i opróżniają istniejący klaster. Ostatecznie wszystko, co pozostaje w klastrze, to zagnieżdżone klastry, które tam wyrosły, same w sobie nie posiadające żadnych luźnych obiektów.

Formalnie algorytm DIANA może być opisany jako kolejność następujących kroków:

1. Niech $I = \{1 \dots n\}$ będzie zbiorem wszystkich n indeksów obiektów oraz $\mathcal{C}$ – zbiorem wszystkich utworzonych do tej pory klastrów.
2. Następne kroki trzeba powtarzać dopóki $|\mathcal{C}| \neq n$.
   - 2.1 Znajdź bieżący klaster zawierający 2 lub więcej obiektów o największej średnicy:
   $$A_* = \arg\max_{A \in \mathcal{C}} \max_{a,b \in A} d(a,b).$$
   - 2.2 Znajdź obiekt w tym klastrze najbardziej różniący się od reszty klastra:
   $$a_* = \arg\max_{a \in A_*} \frac{1}{|A_*| - 1} \sum_{b \in A_* \setminus \{a\}} (a,b)$$
   - 2.3 Wyjąć a_* ze starego klastra i zrobić dla niego nowy klaster (splinter group/grupa odłamków) $A_{new} = \{a_*\}$.
   - 2.4 Dopóki A_* nie jest pusty, kontynuuj migrację obiektów z A_* do A_{new}. Aby wybrać obiekty do migracji, nie tylko weź pod uwagę różnicą od A_*, ale także dostosuj pod kątem odmienności od A_*: niech
   $$a_* = \arg\max_{a \in A} D(a),$$
   gdzie
   $$D(a) = \frac{1}{|A_*| - 1} \sum_{b \in A_* \setminus \{a\}} d(a,b) - \frac{1}{|A_{new}|} \sum_{b \in A_{new}} d(a,b)$$
   - 2.5 Dodaj A_{new} do $\mathcal{C}$.

Algorytm DIANA

Intuicyjnie, $D(a)$ mierze, jak bardzo obiekt chce opuścić swój obecny klaster. Jeżeli obiekt nie pasuje tak samo do nowego klastra, to prawdopodobnie za kilka kroków on założę własne klaster.
Dendrogram DIANA można skonstruować, pozwalając za każdym razem nowemy klastru być dzieckiem klastru A_*. Konstruuje to drzewo z C_0 jako korzeniem i n unikalnymi klastrami pojedynczych obiektów jako liśćmi.

Przykład

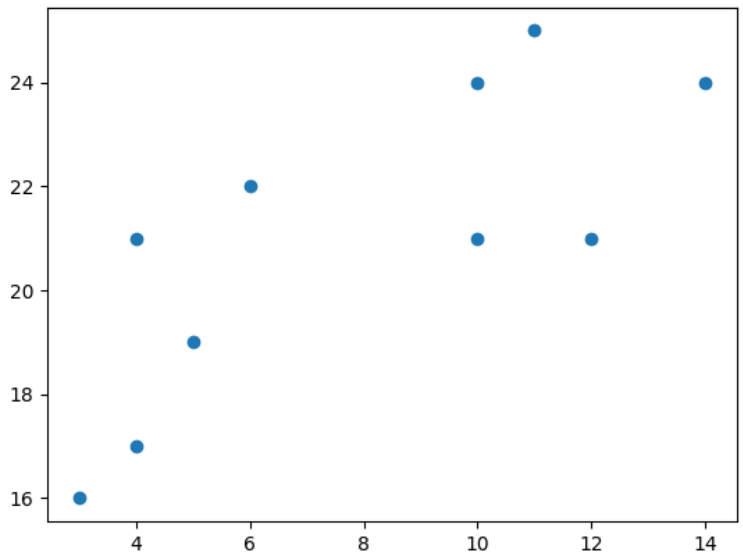
Niech mamy obiekty z rysunku 2 do klasteryzacje. Na początku mamy jeden klaster $A = \{a, b, c, d, e, f\}$.

- $A_* = \{a, b, c, d, e, f\}$, $a_* = a$, $A_{new} = \{a\}$. Wszystkie elementy są bliżej do pozostałych niż do a, dlatego na końcu tej iteracji mamy klastry $\{a\}$ i $\{b, c, d, e, f\}$
- $A_* = \{b, c, d, e, f\}$, $a_* = f$, $A_{new} = \{f\}$. Przynosimy do A_{new} elementy d i e i na końcu tej iteracji mamy klastry $\{a\}$, $\{b, c\}$, $\{d, e, f\}$.
- $A_* = \{d, e, f\}$, $a_* = f$, $A_{new} = \{f\}$. Do A_{new} nic nie przynosimy i na końcu tej iteracji mamy klastry $\{a\}$, $\{b, c\}$, $\{d, e\}$, $\{f\}$.
- $A_* = \{b, c\}$, $a_* = b$, $A_{new} = \{b\}$. Do A_{new} nic nie przynosimy i na końcu tej iteracji mamy klastry $\{a\}$, $\{b\}$, $\{c\}$, $\{d, e\}$, $\{f\}$.
- $A_* = \{d, e\}$, $a_* = d$, $A_{new} = \{d\}$. Do A_{new} nic nie przynosimy i na końcu tej iteracji mamy klastry $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$, $\{f\}$.

Dendrogram będzie taki sam jak na rysunku 3.

Grupowanie hierarchiczne w Pythonie

```
import matplotlib.pyplot as plt

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

plt.scatter(x, y)
plt.show()
```

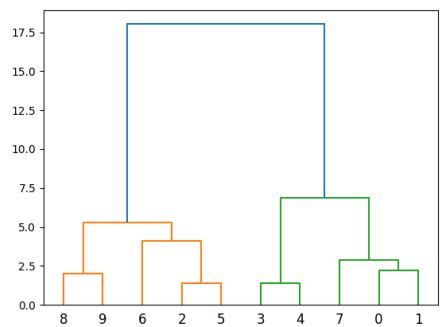


Źródło https://www.w3schools.com/python/python_ml_hierarchial_clustering.asp

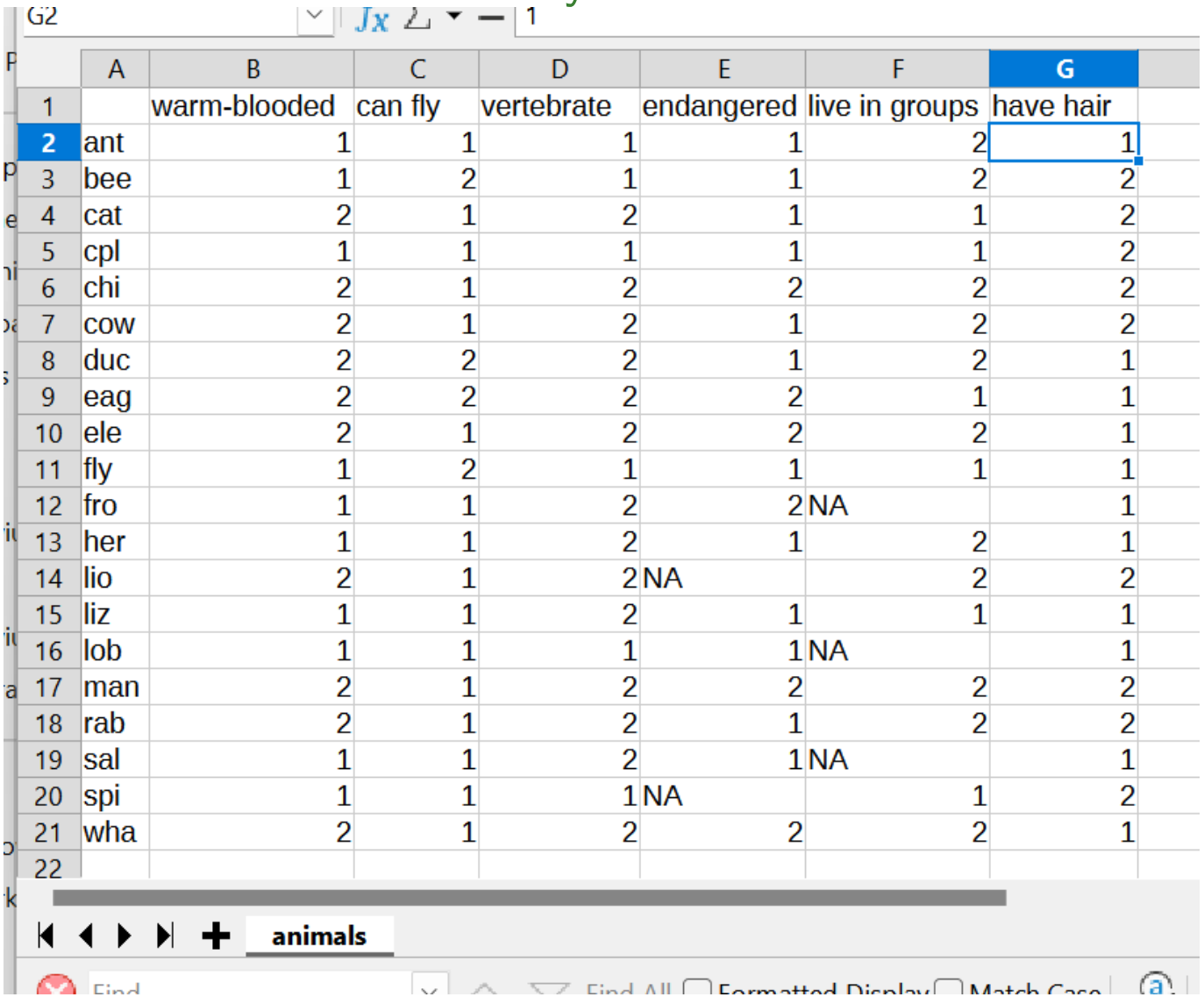
Grupowanie hierarchiczne w Pythonie

Metoda aglomeracyjna: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

```
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

data = list(zip(x, y))

linkage_data = linkage(data, method='ward', metric='euclidean')
#method decyduje jak obliczamy dystans pomiędzy klastrami
dendrogram(linkage_data)
plt.show()
```



Grupowanie hierarchiczne w Pythonie



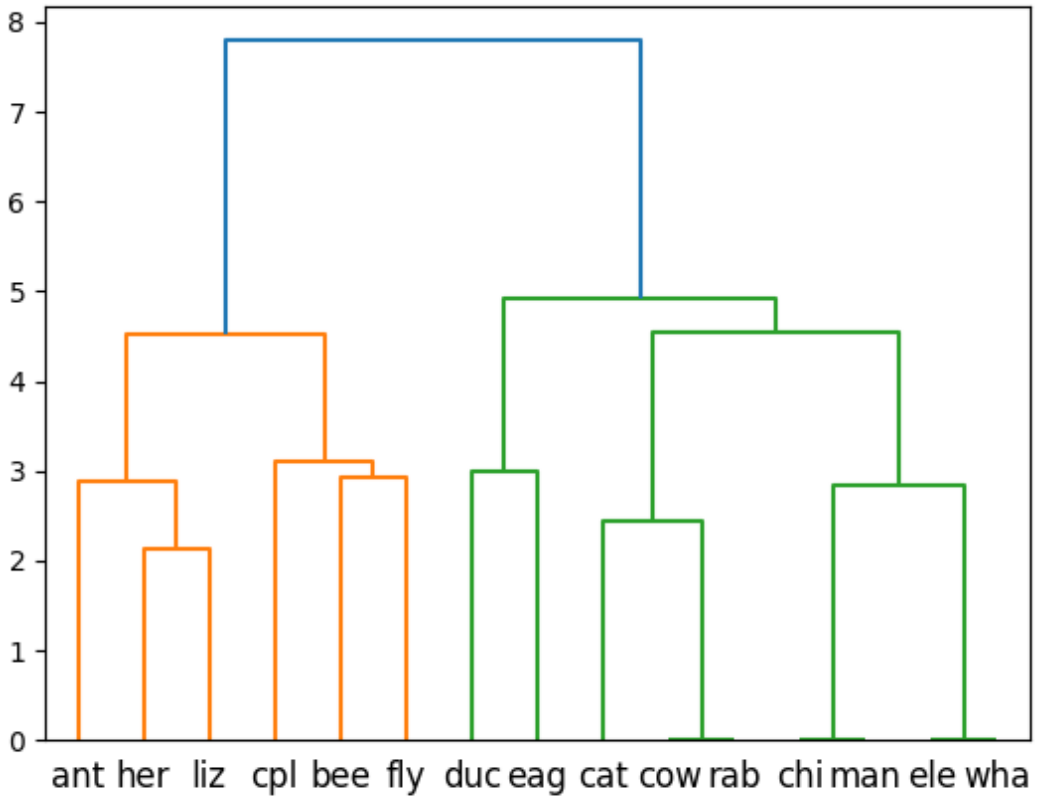
	A	B	C	D	E	F	G
1		warm-blooded	can fly	vertebrate	endangered	live in groups	have hair
2	ant	1	1	1	1	2	1
3	bee	1	2	1	1	2	2
4	cat	2	1	2	1	1	2
5	cpl	1	1	1	1	1	2
6	chi	2	1	2	2	2	2
7	cow	2	1	2	1	2	2
8	duc	2	2	2	1	2	1
9	eag	2	2	2	2	1	1
10	ele	2	1	2	2	2	1
11	fly	1	2	1	1	1	1
12	fro	1	1	2	2	NA	1
13	her	1	1	2	1	2	1
14	lio	2	1	2	NA	2	2
15	liz	1	1	2	1	1	1
16	lob	1	1	1	1	NA	1
17	man	2	1	2	2	2	2
18	rab	2	1	2	1	2	2
19	sal	1	1	2	1	NA	1
20	spi	1	1	1	NA	1	2
21	wha	2	1	2	2	2	1
22							

Źródło danych:
`https://pmagunia.com/dataset/r-dataset-package-cluster-animals`

Grupowanie hierarchiczne w Pythonie: kod

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import scale

data = pd.read_csv("animals.csv").dropna()

newdata = pd.DataFrame(scale(data), index=data.index,
columns=data.columns)
linkage_data = linkage(newdata, method='ward', metric='euclidean')
dendrogram(linkage_data,labels = newdata.index)

plt.show()
```

Grupowanie hierarchiczne w Pythonie: obrazek



Grupowanie hierarchiczne w Pythonie: cut_tree

Jednym z problemów związanych z hierarchicznym grupowaniem jest to, że wynikiem nie jest informacja na ile klastrów należy podzielić obserwacje lub gdzie można przeciąć dendrogram w celu utworzenia klastrów. Możliwe jest przecięcie drzewa na określonej wysokości służy do tego funkcja `cut_tree()`. Zwraca ona wektor zawierający numer klastra każdej z obserwacji.

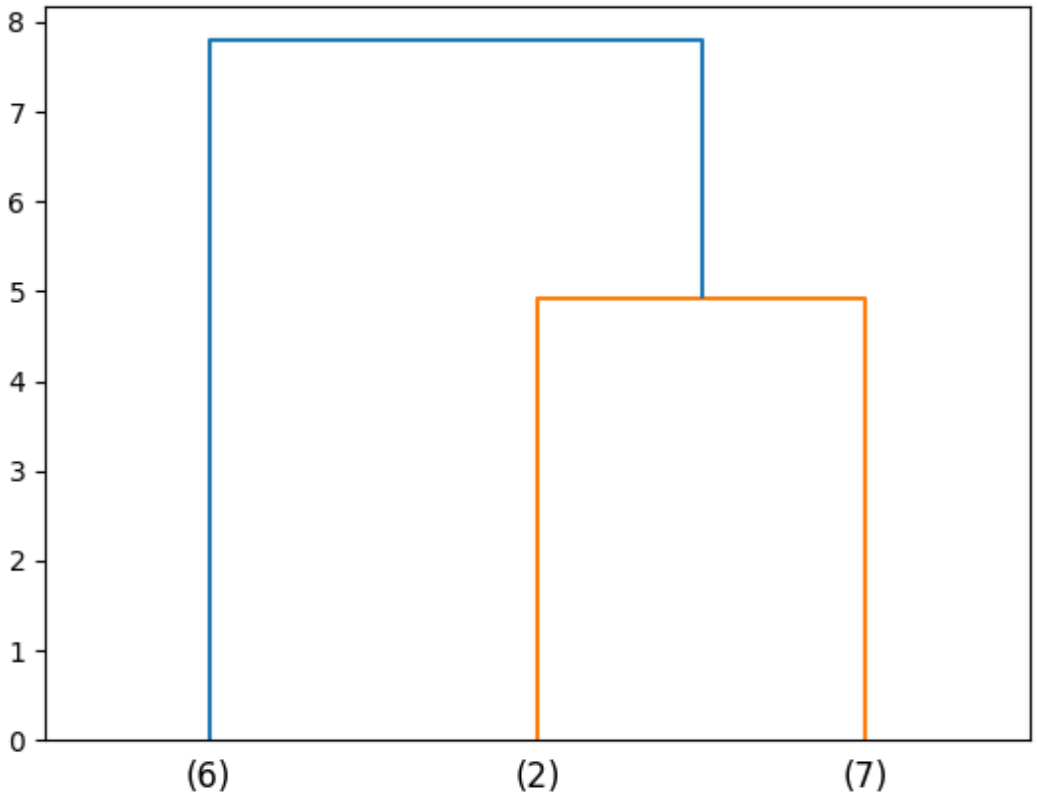
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage, cut_tree
from sklearn.preprocessing import scale

data = pd.read_csv("animals.csv").dropna()

newdata = pd.DataFrame(scale(data), index=data.index,
columns=data.columns)
linkage_data = linkage(newdata, method='ward', metric='euclidean')
dendrogram(linkage_data,labels = newdata.index)
cutree = cut_tree(linkage_data, n_clusters=3)
print(pd.DataFrame(cutree, index=data.index,))
plt.show()
```

Grupowanie hierarchiczne w Pythonie: truncate_mode

truncate_mode pozwala na rysowanie mnie liście na obrazku.

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage, cut_tree
from sklearn.preprocessing import scale

data = pd.read_csv("animals.csv").dropna()

newdata = pd.DataFrame(scale(data), index=data.index,
columns=data.columns)
linkage_data = linkage(newdata, method='ward', metric='euclidean')
dendrogram(linkage_data, truncate_mode = 'lastp', p=3)
cutree = cut_tree(linkage_data, n_clusters=3)
print(pd.DataFrame(cutree, index=data.index,))

plt.show()
```

Grupowanie hierarchiczne w Pythonie: truncate_mode




```
ant  0     chi  1       ele  1       man  1
bee  0    cow  1      fly  0        rab  1
cat  1     duc  2       her  0       wha  1
cpl  0     eag  2       liz  0
```

Klasteryzacja metodą k-średnich: wprowadzenie

Grupowanie k-średnich to metoda kwantyzacji wektorów, która ma na celu podzielenie n obserwacji na k klastrów, w których każda obserwacja należy do klastra o najbliższej średniej (centra lub centroida klastrów), służąc jako prototyp klastra. Klasteryzacja metodą k-średnich minimalizuje wariancje wewnątrz skupień (kwadrat odległości euklidesowych), ale nie regularne odległości euklidesowe, co byłoby trudniejsze.
Mając zbiór obserwacji $(X_1, X_2, \ldots, X_n)$, gdzie każda obserwacja jest m-wymiarowym wektorem rzeczywistym, grupowanie k-średnich ma na celu podzielenie n obserwacji na $(k \leq n)$ zbiorów $\mathrm{S} = \{S_1, S_2, \ldots, S_k\}$, aby zminimalizować sumę kwadratów wewnątrz skupienia (WCSS) (tj. wariancję).

Klasteryzacja metodą k-średnich: wprowadzenie

Formalnie celem jest znalezienie:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{X \in S_i} \|X - \mu_i\|^2 = \arg\min_{S} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

gdzie μ_i jest średnią (zwaną także centroidą) punktów w S_i, tj.

$$\mu_i = \frac{1}{|S_i|} \sum_{X \in S_i} X,$$

gdzie $|S_i|$ jest rozmiarem S_i jest typową L^2-normą. Jest to równoważne minimalizacji odchyleń kwadratów parami punktów w tym samym klastrze:

$$\arg\min_{S} \sum_{i=1}^{k} \frac{1}{|S_i|} \sum_{X, Y \in S_i} \|X - Y\|^2.$$

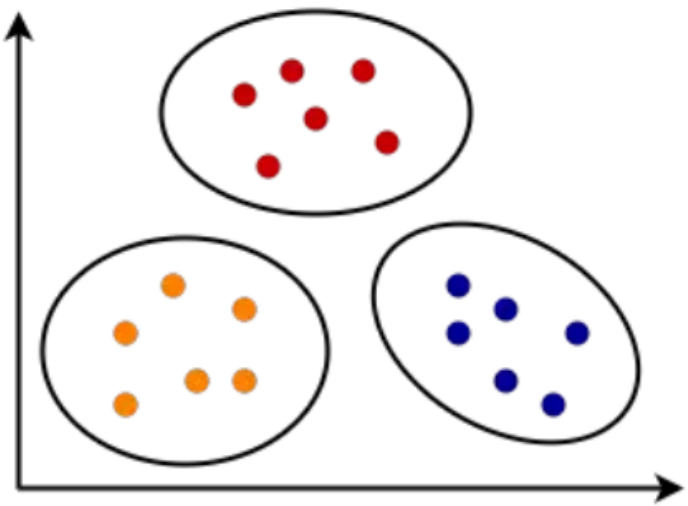
Równoważność wynika z tożsamości:

$$\frac{1}{2} \sum_{X, Y \in S_i} \|X - Y\|^2 = |S_i| \sum_{X \in S_i} \|X - \mu_i\|^2$$

Standardowy algorytm klasteryzacji metodą *k*-średnich

Najpopularniejszy algorytm wykorzystuje technikę iteracyjnego udoskonalania. Ze względu na swoją wszechobecność często nazywany jest „algorytmem k-średnich”; jest on również nazywany algorytmem Lloyda, szczególnie w społeczności informatycznej. Czasami nazywa się to również „naiwnymi k-średnimi”, ponieważ istnieją znacznie szybsze alternatywy.

Before K-Means



After K-Means

Źródło: `https://www.gatevidyalay.com/k-means-clustering-algorithm-example/`

Standardowy algorytm klasteryzacji metodą k-średnich

1. **Inicializacja**. Powszechnie stosowanymi metodami inicjalizacji są Metoda losowego podziału (Random Partition) oraz Metoda Forgy'ego. Metoda losowego podziału najpierw losowo przypisuje klaster do każdej obserwacji, a następnie przechodzi do etapu aktualizacji, obliczając w ten sposób początkową średnią stanowiącą centroid (środek ciężkości) losowo przypisanych punktów klastra. Metoda Forgy'ego losowo wybiera k obserwacji ze zbioru danych i wykorzystuje je jako średnie początkowe
2. **Iteracje**. Mając początkowy zestaw k-średnich $\mu_1, \ldots, \mu_k$, algorytm wykonuje naprzemiennie dwa kroki:
   2.1 **Krok przypisywania**: przypisz każdą obserwację do klastra z najbliższym centroidem (tzn z tej o najmniejszej kwadratowej odległości euklidesowej):

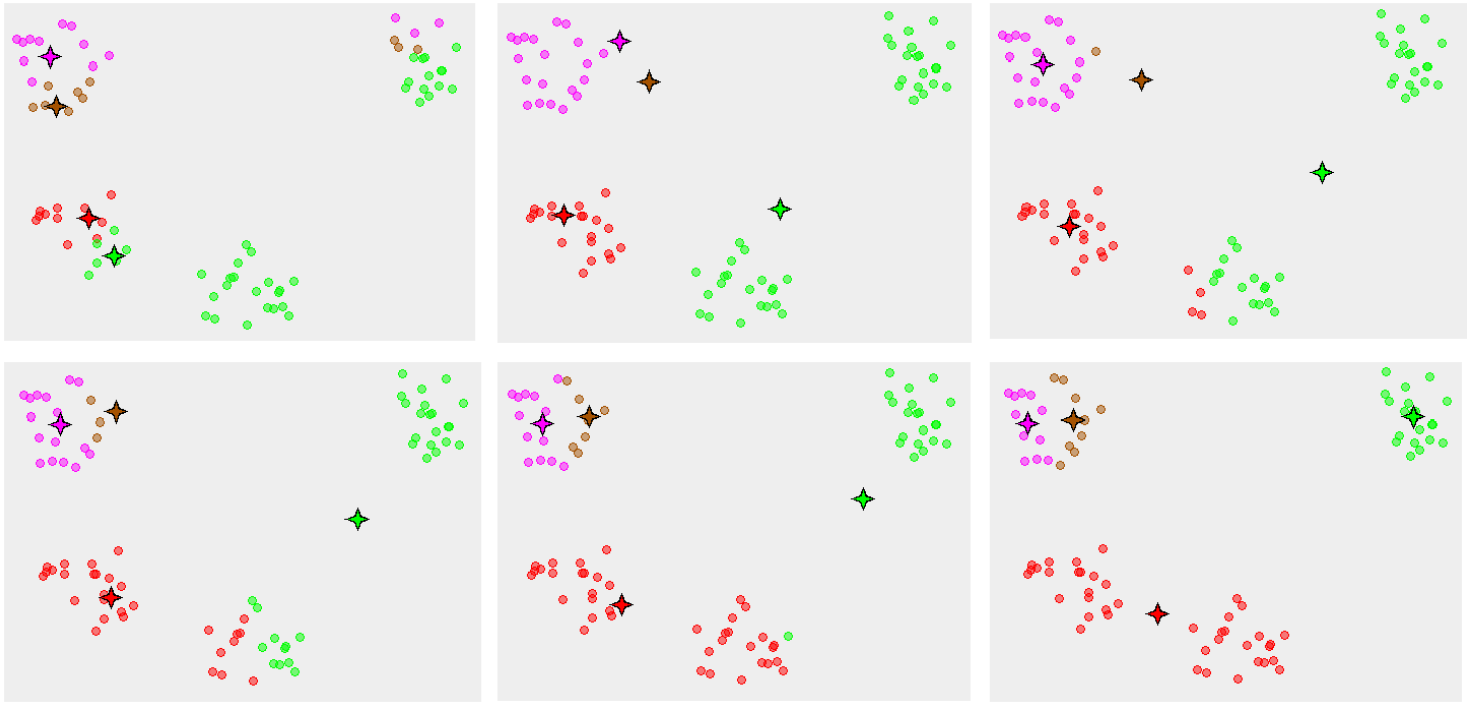
$$S_i^{(t)} = \{X_p : \|X_p - \mu_i^{(t)}\|^2 \leq \|X_p - \mu_j^{(t)}\|^2 \ \forall j, 1 \leq j \leq k\},$$

   gdzie każdy X_p jest przypisany dokładnie do jednego $S^{(t)}$, nawet jeśli mógłby być przypisany do dwóch lub więcej z nich.
   2.2 **Krok aktualizacji**: ponowne obliczenie centroidow obserwacji przypisanych do każdego klastra:

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

   Algorytm osiągnął zbieżność, jeżeli przypisania już się nie zmieniają. Algorytm nie gwarantuje znalezienia optymalnego przypisania.

Obrazki



Typowy przykład zbieżności k-średnich do minimum lokalnego. W tym przykładzie wynik grupowania k-średnich (po prawej stronie) jest sprzeczny z oczywistą strukturą skupień zbioru danych. Małe kółka to punkty danych, cztery gwiazdy promieniste to centroidy (średnie). Początkowa konfiguracja znajduje się na lewym rysunku. Algorytm osiąga zbieżność po pięciu iteracjach przedstawionych na rysunkach, od lewej do prawej, z góry w dol, źródło: `https://en.wikipedia.org/wiki/K-means_clustering`

Przykład 1

Niech mamy cztery punkty: $A = (2, 3), B = (6, 1), C = (1, 2), D = (3, 0)$. Losowo dzielimy ich na dwa klastry (metoda Random Partition): $\{A, B\}$ oraz $\{C, D\}$.

- Centroidy: $\mu_{AB} = (4, 2), \mu_{CD} = (2, 1)$.
- Odległość (kwadrat Euklidesowej!) punktów od centroidow:

	A	B	C	D
μ_{AB}	5	5	9	5
μ_{CD}	4	16	2	2

- Ponieważ $\|A - \mu_{AB}\|^2 > \|A - \mu_{CD}\|^2$, przenosimy A do drugiego klastra. Mamy teraz klastry $\{B\}$ oraz $\{A, C, D\}$ z centroidami: $\mu_B = (6, 1)$ oraz $\mu_{ACD} = \frac{1}{3}(2 + 1 + 3, 3 + 2 + 0) = (2, 1.67)$
- Odległość (kwadrat Euklidesowej!) punktów od centroidow:

	A	B	C	D
μ_B	20	0	26	10
μ_{ACD}	1.78	16.44	1.11	3.78

- Nic nie potrzebuje przenoszenia, koniec. Mamy klastry $\{B\}$ oraz $\{A, C, D\}$.

Przykład 2

Niech mamy cztery punkty: $A = (1, 1), B = (2, 1), C = (4, 3), D = (5, 4)$.
Losowo bierzemy dwa klastra (metoda Forgy'ego): $\mu_1 = A$ oraz $\mu_2 = B$.

- Centroidy: $\mu_1 = (1, 1), \mu_2 = (2, 1)$.
- Odległość (kwadrat Euklidesowej!) punktów od centroidow:

	A	B	C	D
μ_1	0	1	13	25
μ_2	1	0	8	18

- Dodajemy C i D do drugiego klastra. Mamy teraz klastry $\{A\}$ oraz $\{B, C, D\}$ z centroidami: $\mu_1 = (1, 1)$ oraz $\mu_2 = (3.67, 2.67)$
- Odległość (kwadrat Euklidesowej!) punktów od centroidow:

	A	B	C	D
μ_1	0	1	13	25
μ_2	9.91	5.58	0.21	3.54

- Przenosimy B do pierwszego klastra. Mamy teraz klastry $\{A, B\}$ oraz $\{C, D\}$ z centroidami: $\mu_1 = (1.5, 1)$ oraz $\mu_2 = (4.5, 3.5)$
- Odległość (kwadrat Euklidesowej!) punktów od centroidow:

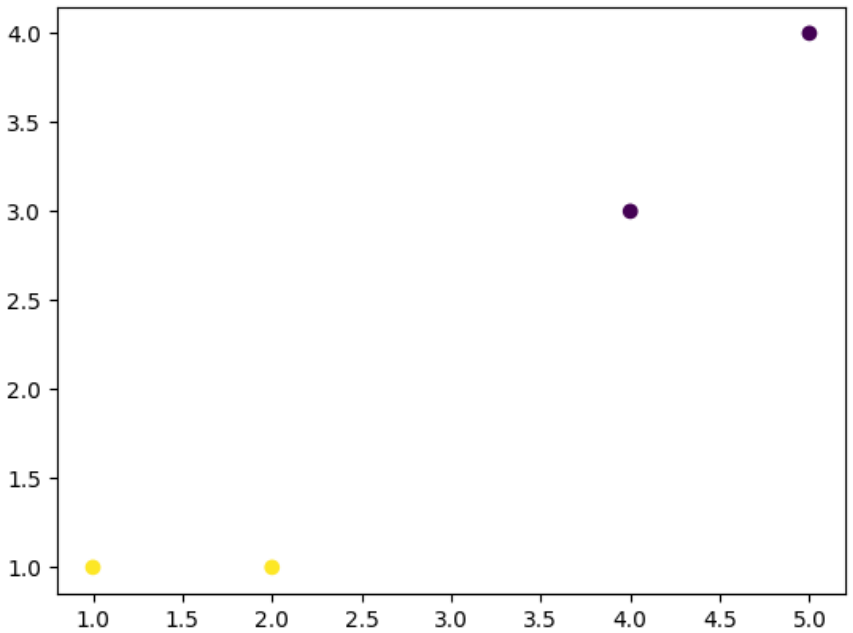
	A	B	C	D
μ_1	0.25	0.25	10.24	21.25
μ_2	18.5	12.58	0.5	0.5

- Nic nie potrzebuje przenoszenia, koniec. Mamy dwa klastra $\{A, B\}$ oraz $\{C, D\}$

Przykład 2 w Python

```
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering

x = [1,2,4,5]
y = [1,1,3,4]

data = list(zip(x, y))

hierarchical_cluster = AgglomerativeClustering(n_clusters=2, metric='eu
labels = hierarchical_cluster.fit_predict(data)

plt.scatter(x, y, c=labels)
plt.show()
```

Przykład 2 w Python: obrazek



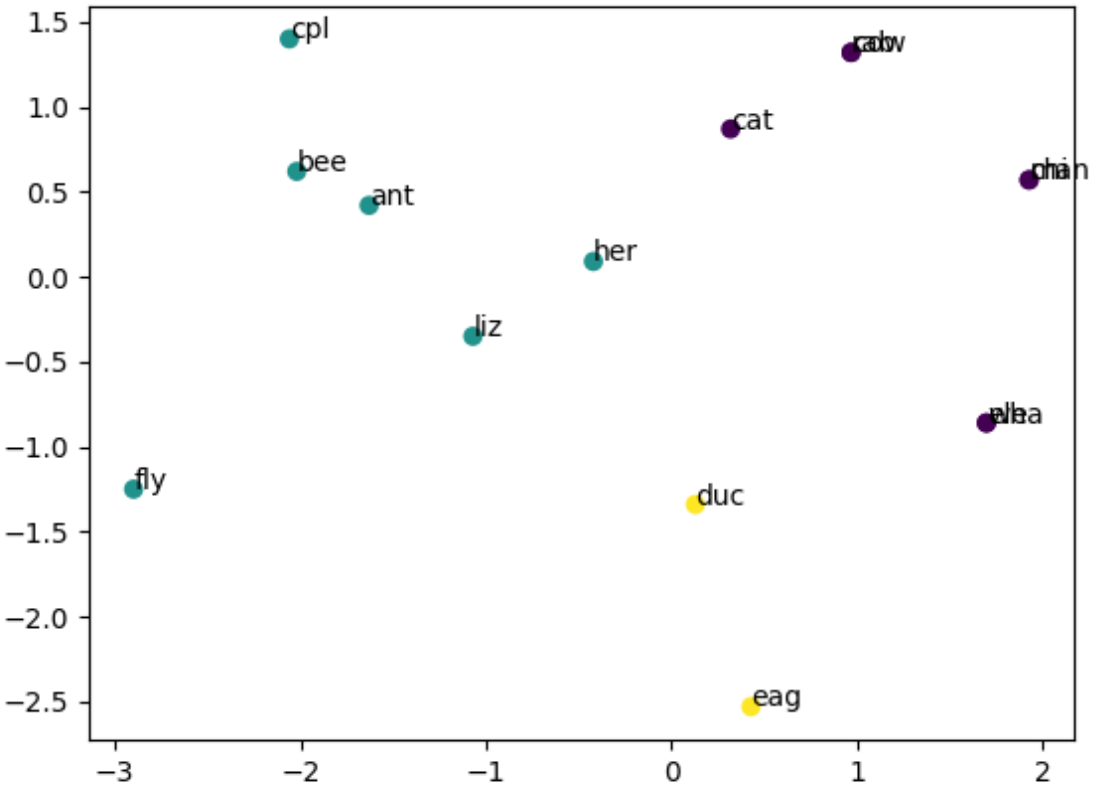
Przykład w Python: animals

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering

data = pd.read_csv("animals.csv").dropna()

newdata = pd.DataFrame(scale(data), index=data.index,
columns=data.columns)



hierarchical_cluster = AgglomerativeClustering(n_clusters=3,
                metric='euclidean', linkage='ward')
labels = hierarchical_cluster.fit_predict(newdata)
```

Przykład w Python animals: obrazek na dwóch składowych

```
pca = PCA(n_components=2)
pca.fit(newdata)
print(pca.components_)#e.v
print(sum(pca.explained_variance_ratio_))#%of information
data2 = pca.transform(newdata).round(2)
print(data2)


x, y = data2[:,0], data2[:,1]
plt.scatter(x,y, c=labels)

for i, txt in enumerate(data.index):
    plt.annotate(txt, (x[i], y[i]))
plt.show()
```

Przykład w Python animals: obrazek



Przykład w Python 2 składowych

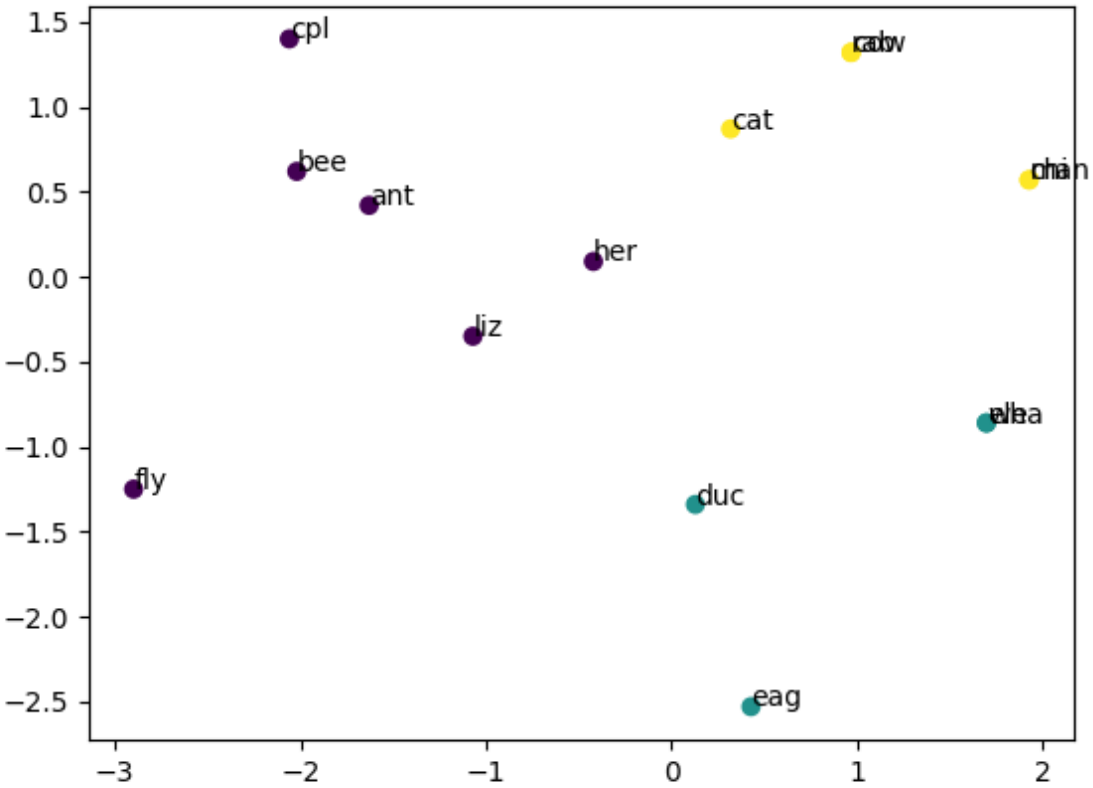
Uwaga! Można tez zrobić odwrotnie: na początku dwie składowych, potem – klasteryzacja!

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering

data = pd.read_csv("animals.csv").dropna()

newdata = pd.DataFrame(scale(data), index=data.index,
columns=data.columns)



print(newdata)
pca = PCA(n_components=2)
pca.fit(newdata)
print(pca.components_)#e.v
print(sum(pca.explained_variance_ratio_))#%of information
data2 = pca.transform(newdata).round(2)
```

Przykład w Python 2 składowych: klasteryzacja

```
hierarchical_cluster = AgglomerativeClustering(n_clusters=3,
                                               metric='euclidean', linkage='ward')
labels = hierarchical_cluster.fit_predict(data2)
x, y = data2[:,0], data2[:,1]
plt.scatter(x,y, c=labels)

for i, txt in enumerate(data.index):
    plt.annotate(txt, (x[i], y[i]))
plt.show()
```

Przykład w Python 2 składowych: obrazek



Metody rozmytej analizy skupień: wprowadzenie

Klasterowanie rozmyte (nazywane również klasterowaniem miękkim lub miękkimi k-średnimi) to forma klasterowania, w której każdy punkt danych może należeć do więcej niż jednego klastra.
W klastrowanie nierozmytem (znane również jako klastrowanie twarde) dane są dzielone na odrębne klastry, przy czym każdy punkt danych może należeć tylko do dokładnie jednego klastra. W klastrach rozmytych punkty danych mogą potencjalnie należeć do wielu klastrów. Na przykład jabłko może być czerwone lub zielone (grupowanie twarde), ale jabłko może być również czerwone ORAZ zielone (grupowanie rozmyte). Tutaj jabłko może być do pewnego stopnia czerwone i do pewnego stopnia zielone. Zamiast jabłka należącego do zielonego [zielony = 1] i nie czerwonego [czerwony = 0], jabłko może należeć do zielonego [zielony = 0,5] i czerwonego [czerwony = 0,5]. Wartości te są normalizowane w zakresie od 0 do 1; jednakże nie reprezentują one prawdopodobieństw, więc te dwie wartości nie muszą sumować się do 1.
Do każdego z punktów danych (tagów) przypisane są stopnie członkostwa. Te stopnie członkostwa wskazują stopień, w jakim punkty danych należą do każdego klastra. Zatem punkty na krawędzi klastra, o niższych stopniach przynależności, mogą znajdować się w klastrze w mniejszym stopniu niż punkty w centrum klastra.

Metoda FCM

Jednym z najczęściej stosowanych algorytmów grupowania rozmytego jest algorytm grupowania rozmytego C-średniego (angl. Fuzzy C-means clustering, FCM). Klastrowanie rozmytych C-średnich (FCM) zostało opracowane przez J.C. Dunna w 1973 r.

Algorytm rozmyty c-średnich jest bardzo podobny do algorytmu k-średnich:

1. Wybierz liczbę klastrów I i liczba $m \in (1, \infty)$, charakteryzującą na ile rozmyty będą klastry ($m = 1$ – nie rozmyty klastry).
2. Wybierz losowo centry klastrów $c_1, \ldots, c_I$.
3. Powtarzaj, aż algorytm osiągnie zbieżność (tzn. zmiana współczynników pomiędzy dwiema iteracjami nie będzie większa niż ε – zadany próg czułości):
   - 3.1 Oblicz współczynnik ze X_i należy do klastra z centroidem c_j:

$$w_{ij} = \frac{1}{\sum_{k=1}^{I} \left(\frac{\|X_i - c_j\|}{\|X_i - c_k\|} \right)^{\frac{2}{m-1}}}.$$

   - 3.2 Oblicz środek ciężkości dla każdego klastra.

$$c_k = \frac{\sum_x w_k(X)^m X}{\sum_x w_k(X)^m}.$$

Metoda FCM

FCM ma na celu zminimalizowanie funkcji:

$$J(W, C) = \sum_{i=1}^{n} \sum_{j=1}^{l} w_{ij}^{m} \|X_i - c_j\|^2$$

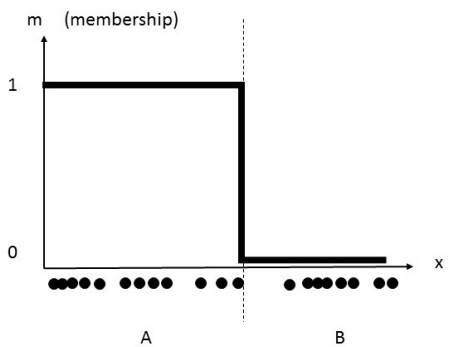
FCM metoda jest bardzo ważnym narzędziem do przetwarzania obrazu w grupowaniu obiektów na obrazie. Funkcje przynależności mają na celu opisanie kolorów zgodnie z ludzką intuicją identyfikacji kolorów.

Metoda FCM

Aby lepiej zrozumieć FCM metodą, poniżej na rysunku przedstawiono klasyczny przykład danych jednowymiarowych na osi x, :

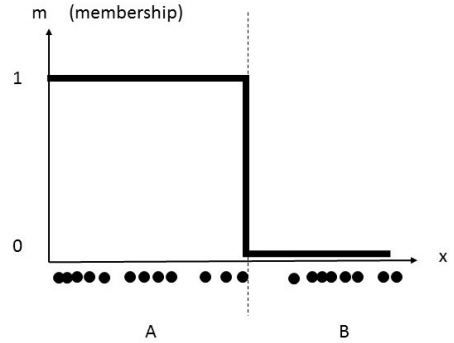
Rysunek: Punkty na osi x, źródło: Wikipedia

Ten zbiór danych można tradycyjnie podzielić na dwa klastry. Wybierając próg na osi x, dane są rozdzielane na dwa klastry. Powstałe klastry są oznaczone jako A i B, jak widać na rysunku poniżej. Każdy punkt należący do zbioru danych miałby zatem współczynnik przynależności 1 lub 0. Ten współczynnik przynależności każdego odpowiedniego punktu danych jest reprezentowany przez włączenie osi y.

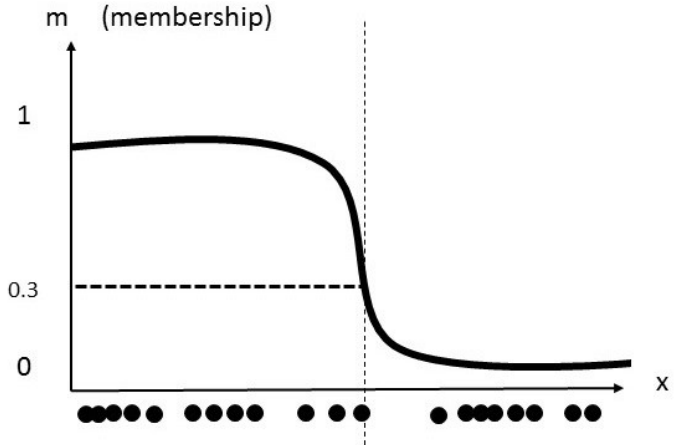


Rysunek: Punkty na osi x oraz próg klasteryzacji, źródło: Wikipedia

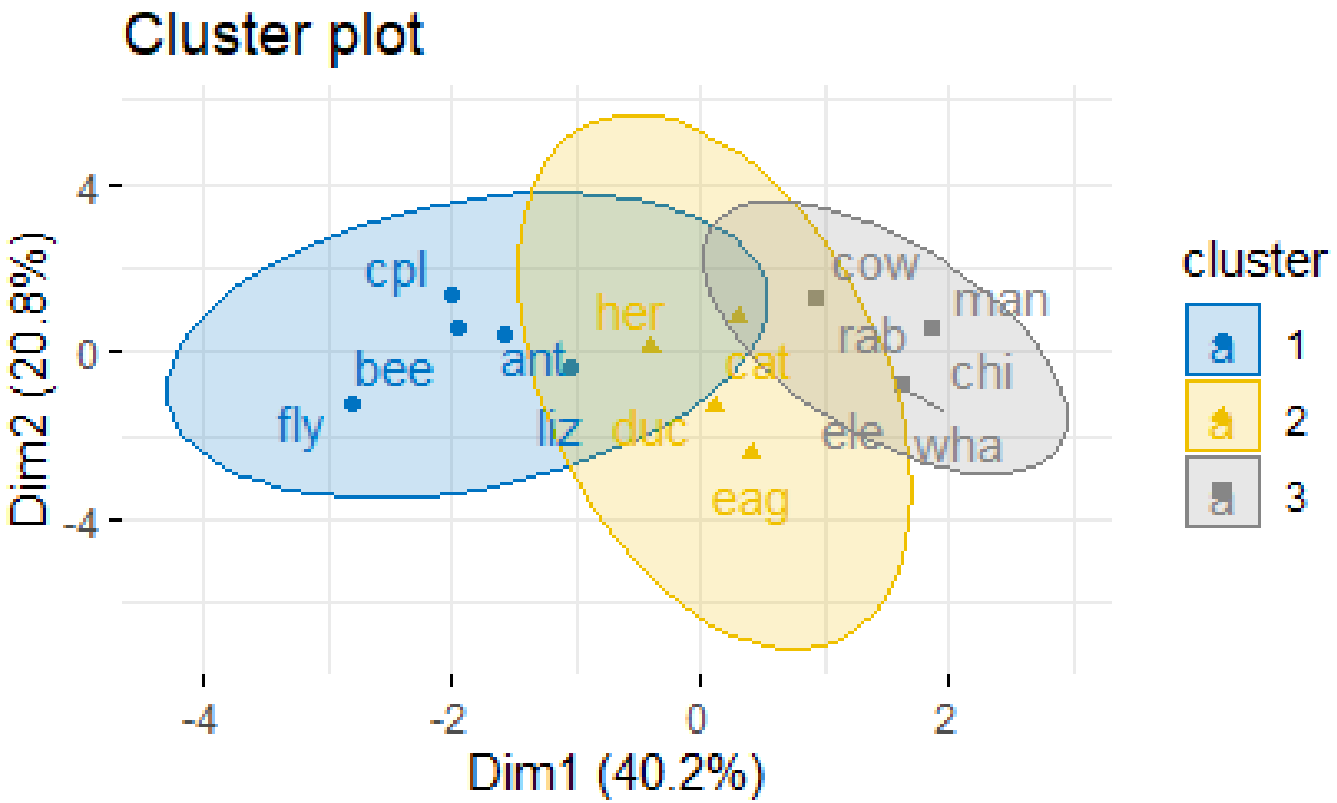
Metoda FCM



W klastrach rozmytych każdy punkt danych może należeć do wielu klastrów. Po rozluźnieniu definicji współczynników przynależności od ściśle 1 lub 0, wartości te mogą mieścić się w zakresie od dowolnej wartości od 1 do 0. Rysunek ponizej przedstawia zestaw danych z poprzedniego grupowania, ale teraz zastosowano rozmyte grupowanie C-średnich. W pierwszej kolejności można wygenerować nową wartość progową definiującą dwa klastry. Następnie generowane są nowe współczynniki przynależności dla każdego punktu danych na podstawie centroid klastrów, a także odległości od każdego centroidu



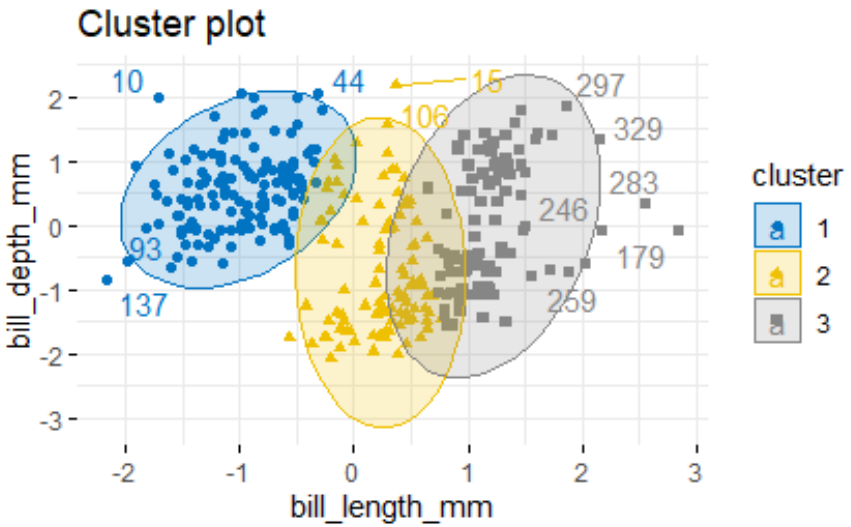
Obrazki



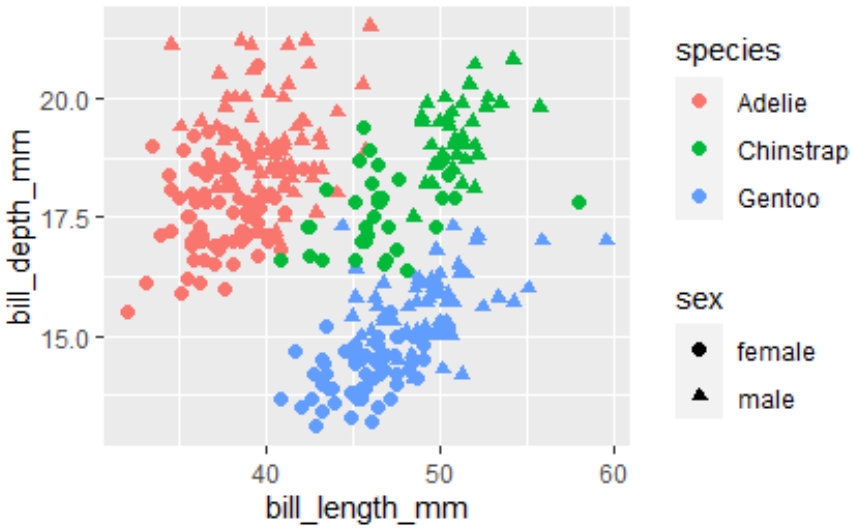
Żywioły, podzielony na 3 rozmytych klastra w R-ze- W Pythonie wizualizacja nie jest łatwa.

Obrazki

Pingwiny podzielone na 3 rozmytych klastra według szerokości i długości dzióby.



Pingwiny według szerokości i długości dzióby: pleć i gatunek.



Przykłady

- Punkty $(1,2),(2,2),(4,7),(3,4)$ podziel na dwa klastry na karteczce i w Pythonie.
- Punkty $(2,3),(6,1),(1,2),(3,0),(4,2),(2,7),(4,9)$ podziel na dwa klastry w Pythonie.
- Zrób dendragram na zestawie danych „mtcars".

  `https://gist.github.com/seankross/a412dfbd88b3db70b74b`

  Podziel samochody na 5 grup. Narysuj odpowiedni dendrogram.
- Zrób klasteryzacją „iris" (tylko dane liczbowe!) metodą k-średnich na 3 i 6 klastrówi narysuj obrazek dla dwóch składowych głównych.
- Zrób klasteryzacją „pingwinów" (tylko dane liczbowe!) metodą k-średnich na 3 klastrowi narysuj obrazek dla dwóch składowych głównych i porównaj z gatunkami.