Laboratorium 7 – grupowanie nienadzorowane

ver. 1.03

Streszczenie

Grupowanie nienadzorowane inaczej analiza skupień lub klasteryzacja (ang. *data clustering*) jest metodą łączenia obiektów w grupy (klastry). Uczenie odbywa się w sposób nienadzorowany ponieważ w trakcie działania algorytmu nie znamy rzeczywistej przynależności obiektu do danej klasy (w przeciwieństwie do zadania klasyfikacji). Łączenie w grupy odbywa się jedynie na podstawie podobieństwa obiektów np. ich odległości w przestrzeni. W wyniku działania algorytmu powstają grupy danych. Obiekty należące do jednego klastra są w miarę jednorodne i podobne do siebie w większym stopniu niż obiekty należące do innego klastra. Definicja klastra nie jest jednoznaczna i może się różnić od siebie w zależności od działania konkretnego algorytmu. Algorytmy grupowania danych dzielimy na kilka podstawowych kategorii, w tym m.in.:

- Grupowanie hierarchiczne (w tym metody aglomeracyjne), gdzie głównym celem jest aby odległość między obserwacjami należącymi do tego samego klastra była jak najmniejsza, a odległości między klastrami były jak największe.
- Algorytm centroidów w tym metoda k-średnich (ang. *k-means*), gdzie klaster reprezentowany jest przez centralny wektor danych.
- Algorytmy klasteryzacji oparte na rozkładach prawdopodobieństwa, w których obiekty można zdefiniować jak elementy należące z dużym prawdopodobieństwem do tego samego rozkładu. Przykładem może być użycie Mieszanki Rozkładów Gaussowskich (ang. *Gaussian Mixture Models*) uczonych algorytmem EM (ang. *Expectation-Maximization*).

1 Cel

Celem laboratorium jest zapoznanie się z algorytmami grupowania nienadzorowanego w szczególności z alg.: k-średnich, metodami aglomeracyjnymi oraz Mieszanką Rozkładów Gaussowskich uczonych alg. EM. Użycie ww. algorytmów w procesie klasteryzacji danych oraz kwantyzacji wektorowej.

Informacja. Odpowiedzi na pytania teoretyczne zawarte poniżej można umieścić w lakonicznej formie w postaci komentarzy do napisanego kodu.

2 Klasteryzacja

1. Wczytaj dane `iris` oraz rozdziel je na cześć wejściową (X) i decyzje (Y).

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

2. Wykonaj klasteryzację danych X przy użyciu aglomeracyjnych metod grupowania hierarchicznego `sklearn.cluster.AgglomerativeClustering`. Przetestuj jak działa metoda przy użyciu różnych kryteriów łączenia punktów w klastry (parametr `linkage` do konstruktora klasy `AgglomerativeClustering`):

   - metoda najbliższego sąsiedztwa
   - metoda średnich połączeń
   - metoda najdalszych połączeń
   - metoda Warda

   Pytanie. W jaki sposób działają metody aglomeracyjne?

3. Dopasuj rezultat klasteryzacji do rzeczywistych klas decyzyjnych.
   Informacja. W wyniku klasteryzacji dane grupowane są w sposób nienadzorowany. Nawet w przypadku idealnego pogrupowania danych, indeksy klas decyzyjnych mogą różnić się od rzeczywistych wartości. Dla przykładu macierz Y_{real} zawiera rzeczywiste etykiety, natomiast macierz Y_{pred} zawiera etykiety uzyskane w wyniku działania dowolnego algorytmu klasteryzacji:

$$Y_{real} = [0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2],$$
$$Y_{pred} = [1, 1, 1, 2, 2, 2, 2, 2, 0, 0, 0, 0].$$

   Należy pamiętać, że w przypadku rzeczywistych zbiorów danych, nie wszystkie próbki, zostaną przydzielone do właściwych klastrów i obie macierze mogą mieć następującą postać:

$$Y_{real} = [0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2],$$
$$Y_{pred} = [1, 1, 1, 2, 1, 2, 0, 2, 0, 2, 0, 0].$$

   W obu przypadkach należy znaleźć taką permutację, która mapuje Y_{real} na Y_{pred}, tak aby Y_{pred} jak najbardziej odpowiadało Y_{real}. Można to zrobić w poniższy sposób:

```
def find_perm(clusters, Y_real, Y_pred):
        perm=[]
        for i in range(clusters):
                idx = Y_pred == i
                new_label=scipy.stats.mode(Y_real[idx])[0][0]
                perm.append(new_label)
        return [perm[label] for label in Y_pred]
```

Parametry powyższej metody są następujące:

- clusters - liczba klastrów,
- Y_real - rzeczywiste etykiety punktów,
- Y_pred - etykiety punktów nadane przez alg. klasteryzacji (np.: atrybut `labels_` w klasie `AgglomerativeClustering`).

Pytanie. W jaki sposób działa powyższa funkcja.

4. Oblicz indeks Jaccarda dla danych pogrupowanych względem klasy decyzyjnej (`sklearn.metrics.jaccard_score`).
   Informacja. Odpowiedni dobór parametrów pozwoli na policzenie indeksu dla wielu klas przy jednym wywołaniu funkcji.
   Pytanie. Co to jest indeks Jaccarda i jaka jest jego interpretacja?

5. Zwizualizuj dane w przestrzeni 2D. Wizualizacji należy dokonać przy pomocy metody PCA rzutując dane na dwie pierwsze składowe główne.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)
```

   Należy stworzyć trzy wykresy:

   - dane z podziałem na rzeczywiste klasy,
   - dane z podziałem na klasy decyzyjne obliczone w wyniku działa algorytmu,
   - wykres obrazujący, które dane zostały poprawnie sklasyfikowane, a które nie.

   Klasy otoczyć powłoką wypukłą `scipy.spatial.ConvexHull`. Przykładowa wizualizacja została zaprezentowana na Rysunku 1.

6. W analogiczny sposób wykonaj wizualizację w przestrzeni 3D. Przykładowa wizualizacja została zaprezentowana na Rysunku 2.

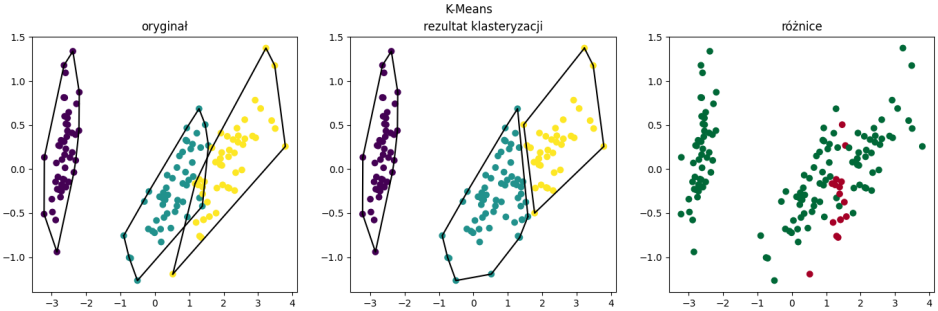
7. Narysuj dendrogram (`scipy.cluster.hierarchy.dendrogram`) obrazujący związki pomiędzy wybranymi elementami.

8. Powtórz działanie dla następujących metod:

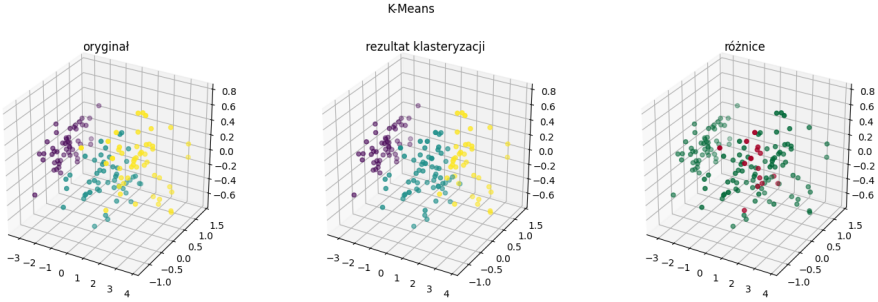
   - metody k-means (`sklearn.cluster.KMeans`),
   - metody k-means napisanej na potrzeby laboratorium 6,
   - metody GMM (`sklearn.mixture.GaussianMixture`).

9. Powtórz działanie dla zbioru `zoo` dołączonego do laboratorium 5.

Rysunek 1: Przykładowa wizualizacja 2D.



Rysunek 2: Przykładowa wizualizacja 3D.



3 Kwantyzacja

1. Wczytaj dowlony obraz np.: w rozmiarze $640 \times 480 \times 3$.
   Każdy piksel możemy traktować jako punkt w przestrzeni 3D o współrzędnych R, G, B.

2. Przekształć (zwektoryzuj) obraz do rozmiaru $640 \cdot 480 \times 3$ tj. 307200×3.
   Należy zamieniamy rozmiar tablicy z punktu 1 tak aby otrzymać tablicę, w której każdy wiersz to pojedynczy piksel. Macierz będzie miała 3 kolumny: R, G i B. Otrzymamy następującą macierze:

$$img = \begin{bmatrix} x_{1,R} & x_{1,G} & x_{1,B} \\ x_{2,R} & x_{2,G} & x_{2,B} \\ x_{3,R} & x_{3,G} & x_{3,B} \\ & \cdots & \\ x_{n,R} & x_{n,G} & x_{n,B} \end{bmatrix},$$

   gdzie n to liczba pikseli w klastrze (tu 307200).

3. Wybranymi trzema metodami grupowania danych podziel tak utworzone próbki na kolejno 2, 3, 5, 10, 30 i 100 klastrów.
   Dla przykładu dla algorytmu k-means przy $k = 5$ otrzymamy 5 centrów:

$$C = \begin{bmatrix} c_{1,R} & c_{1,G} & c_{1,B} \\ c_{2,R} & c_{2,G} & c_{2,B} \\ & \cdots & \\ c_{k,R} & c_{k,G} & c_{k,B} \end{bmatrix},$$

   otrzymamy również macierz $labels$, mówiącej o przynależności każdego piksela z macierzy img do konkretnego centrum z macierzy C np.:

$$labels = [1, 1, 5, 2, 4, \dots].$$

   Wskazówka. Proponuję zacząć od metody k-means lub GMM ze względu na łatwość wyciągnięcia współrzędnych punktu centralnego. Dla metod aglomeracyjnych punkt centralny należy policzyć ręcznie.

4. Wykonaj kwantowanie wektorowe zastępując wszystkie próbki należące do klastra punktem centralnym.
   W wyniku działania każdy piksel obrazu powinien zostać zastąpiony odpowiednim centrum na podstawie macierzy C i P tj.:

$$img\_quant = \begin{bmatrix} c_{1,R} & c_{1,G} & c_{1,B} \\ c_{1,R} & c_{1,G} & c_{1,B} \\ c_{5,R} & c_{5,G} & c_{5,B} \\ c_{2,R} & c_{2,G} & c_{2,B} \\ c_{4,R} & c_{4,G} & c_{4,B} \\ & \cdots & \end{bmatrix}$$

5. Na danych skwantowanych wykonaj przekształcenia odwrotnego.
   W wyniku działania obraz *img_quant* powinien zostać przekształcony do przestrzeni oryginalnej tj.: $640 \times 480 \times 3$.

6. Zwizualizuj oba obrazy (przed i po kwantyzacji, tj. *img* i *img_quant*) dla różnych metod grupowania oraz dla różnej liczby skupień.

7. W każdym przypadku wyznacz błąd kwantyzacji (przy pomocy błędu średniokwadratowego), a następnie zwizualizuj rozkład błędu dla konkretnych pikseli.
   Wizualizacja powinna posiadać postać obrazu monochromatycznego o wymiarach 640×480, gdzie wartość konkretnego piksela jest wartością błędu.

8. Wykonaj wektoryzację do rozmiaru $640 \cdot (480/2^n) \times 3 \cdot 2^n$, dla różnych wartości n.
   W tym przypadku algorytm grupowania danych będzie działał w wyżej wymiarowej przestrzeni. Dla $n = 3$ będzie to działanie w przestrzeni 24 wymiarowej $(640 \cdot 60 \times 24)$. Macierz *img* będzie miała wtedy postać:

$$img\_quant8 = \begin{bmatrix} x_{1,R} & x_{1,G} & x_{1,B} & \cdots & x_{8,R} & x_{8,G} & x_{8,B} \\ x_{9,R} & x_{9,G} & x_{9,B} & \cdots & x_{16,R} & x_{16,G} & x_{16,B} \\ x_{17,R} & x_{17,G} & x_{17,B} & \cdots & x_{24,R} & x_{24,G} & x_{24,B} \\ & & & \cdots & & & \end{bmatrix}.$$

   Pytanie. Jaka będzie wizualna różnica w porównaniu do obrazu utworzonego z obrazu *img_quant*? Jak na wygląd obrazu będzie wpływało zwiększanie parametru n?

9. Przetestuj jak algorytm będzie działał w przypadku permutacji pikseli.
   W celu wykonania zadania należy wykonać perwmutację pikseli tj. `new_img = np.reshape(img, (640*480,3))[perm,:]`, gdzie `perm` jest macierzą permutacji. *Informacja.* Należy napisać metodę dokonującą permutacji odwrotnej w celu odtworzenia właściwej kolejności pikseli w obrazie.