Zgrupowanie reprezentacji

Celem zadania jest bliższe zapoznanie się z algorytmem klasteryzacji *k-means*, sposobami na ustalenie optymalnego *k*, oraz metodami oceny jakości samego grupowania.

Dane do pracy

Do wykonania ćwiczenia będą nam potrzebne dwa zbiory danych. Drugi z nich jest już (niemal gotowy) - to baza informacji o czołowych piłkarzach utworzona na bazie ich statystyk wykorzystywanych w popularnej grze FIFA 22 (użyjemy jej w dalszej części zadania). Pierwszy jednak musimy przygotować sami.

Będzie to dwuwymiarowy zbiór służący do testowania naszej metody i wizualnej oceny uzyskiwanych rezultatów. Powinien składać się z około 10 dość oczywistych klastrów, które nie mają idealnie jednorodnego kształtu i rozmiaru i z których niektóre nie są w pełni odseparowane od pozostałych. Dane możemy wygenerować używając kodu:

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=150,
                  n_features=2,
                  centers=10,
                  cluster_std=0.5,
                  shuffle=True,
                  random_state=0)
```

Zależy nam na tym, by istniała znana nam "właściwa" liczba klastrów *k* oraz by zbiór stanowił przynajmniej umiarkowane wyzwanie dla techniki *k-means*.

Narzędzia diagnostyczne

Potrzebujemy również (względnie) obiektywnego sposobu oceny jak solidna jest uzyskana klasteryzacja, lepszego niż tzw. technika „na oko” (liczby można porównywać, z subiektywnymi wrażeniami jest nieco trudniej). Na szczęście jest to standardowa potrzeba i odpowiednie narzędzia są powszechnie dostępne, przykładowo biblioteka *scikit-learn* oferuje szerokie spektrum sposobów na ewaluację otrzymanego grupowania (patrz: https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation).

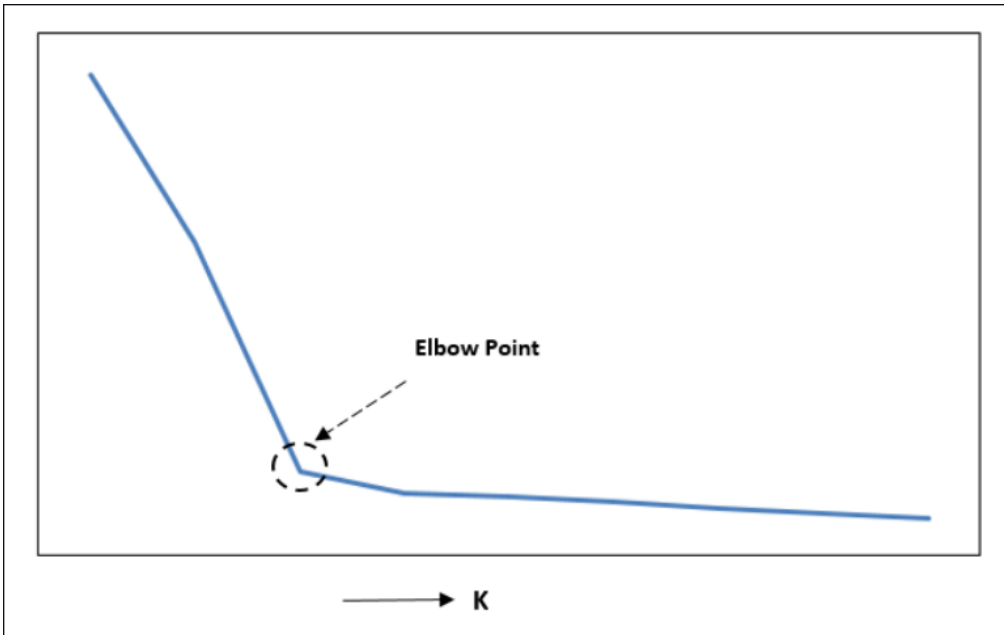
Na potrzeby zadania wybierzmy dwie dowolne miary jakości klasteryzacji. Nadają się dowolne dwie. Warto jednak przeczytać o nich choć trochę (wiedzieć czym się charakteryzują, na co są wyczulone, co ignorują, etc.).

Ustalanie właściwego poziomu rozdrobnienia

Pora zabrać się za wspominane wcześniej dobieranie parametru *k* (czyli liczby klastrów do odnalezienia). Podejdziemy do tego na dwa sposoby.

1. Prostszy z nich to technika znana powszechnie jako tzw. zasada łokcia (*elbow rule*).

- Przyjmijmy rozsądny zakres rozważanych *k* (np. od 1 do 30). Dla każdego *k* z tego zakresu:
  - kilkakrotnie wykonajmy klasteryzację metodą *k-means* (z inicjalizacją *k-means++*);
  - dla każdego uzyskanego grupowania obliczmy wybrane w poprzedniej sekcji miary jakości, wyciągnijmy z nich średnie, zanotujmy odchylenia standardowe.
- W efekcie powinniśmy uzyskać dwa wykresy przedstawiające zależność danej metryki od parametru *k*. Dla zadania klasteryzacji typowy jest trend w którym początkowo wartość metryki bardzo szybko ulega poprawie - ale od pewnego *k* stabilizuje się i zmienia swą wartość znacznie wolniej. Tworzy to charakterystyczny kształt ugiętej ręki. Miejsce, gdzie ręka miałaby łokieć zazwyczaj wyznacza potencjalnie-optymalną wartość parametru:

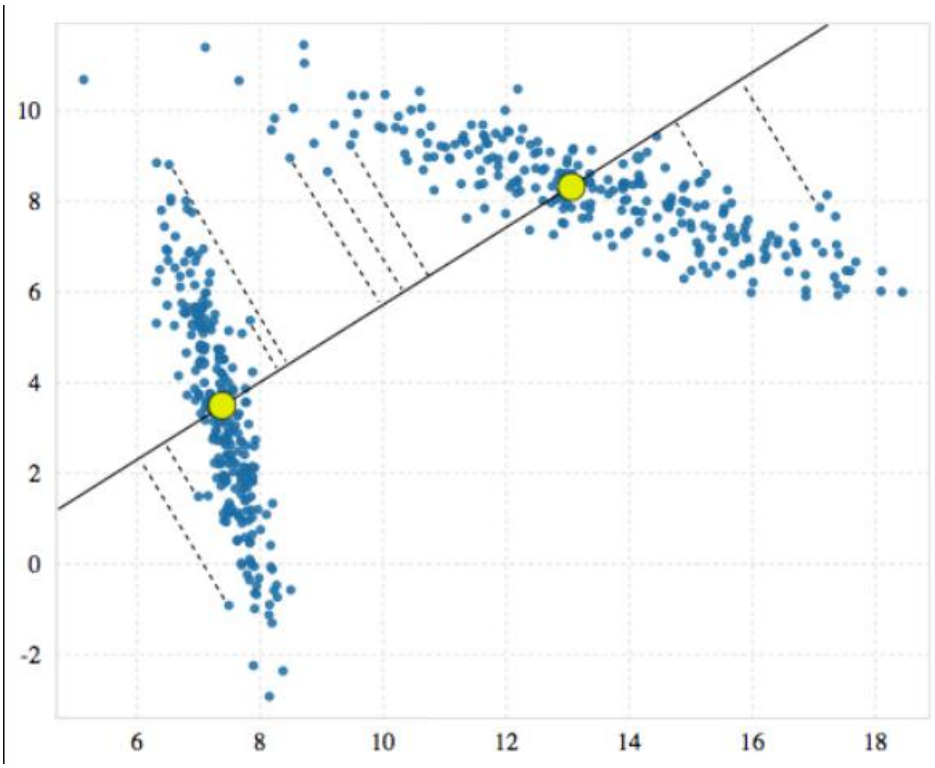


- Sprawdźmy wartości *k* wyznaczone w ten sposób dla każdej z miar jakości.
  - Czy są takie same?
  - Czy są zgodne z oczekiwanym *k*?
  - Jak dla tych *k* wygląda podział punktów na klastry? Czy jest zgodny z tym oczekiwanym? Czy „ma sens”?

2. Druga technika jest nieco trudniejsza i zakłada adaptatywne hierarchiczne dobieranie liczby klastrów. Wykorzystujemy tu następujący algorytm:

- Początkowo wszystkie punkty należą do jednego klastra.
- Dla tego klastra wykonujemy klasteryzację *k-means* z *k=2*.
- Uzyskamy w ten sposób dwa nowe centra klastrów. Wyznaczają one pewną prostą przecinającą całą przestrzeń obserwacji.

- Rzutujemy wszystkie obserwacje z naszego klastra na tą prostą - teraz będą je reprezentowały pojedyncze liczby.



- Sprawdzamy, czy efekt takiego rzutowania pasuje do rozkładu normalnego (jego histogram tworzy jedną górkę o charakterystycznym "gaussowatym" kształcie). Jak to zrobić? Tutorial z fragmentami kodu do skopiowania dostępny tutaj (sekcja *Statistical Normality Tests*): https://machinelearningmastery.com/a-gentle-introduction-to-normality-tests-in-python/.
  Niezależnie od wybranego testu, będzie trzeba ustalić poziom czułości (dobrać odpowiednią wartość progową dla *p-value*).
- Wynikiem testu jest jeden z dwóch scenariuszy.
  - Tak! Punkty po zrzutowaniu na prostą tworzą rozkład normalny. To znaczy, że nasz oryginalny klaster jest zupełnie w porządku i nie należy go rozdrabniać na dwa mniejsze. Zostawiamy go takim, jakim był.
  - Nie! Punkty po zrzutowaniu na prostą nie pasują do rozkładu normalnego. To znaczy, że początkowy klaster wymaga rozbicia na mniejsze podgrupki.
    - Dzielimy klaster na dwa podklastry znalezione przed chwilą przez *k-means*.
    - Dla każdego z podklastrów rekurencyjnie uruchamiamy tę samą procedurę (podział na dwa mniejsze klastry, wyznaczenie prostej, rzutowanie, test na normalność, decyzja o dalszych podziałach).
- Algorytm kończymy, gdy nie są już konieczne dalsze podziały (każdy ze znalezionych klastrów zdał test na normalność).
  - Alternatywnie - nie rozdrabniamy dalej klastrów, które mają mniej elementów niż ustalona minimalna wartość (jeżeli grupa zawiera np. 5 elementów, to raczej nie warto jej dalej dzielić).
- Pobawmy się chwilę algorytmem, eksperymentując z różnymi progami czułości.
  - Jakie podziały znajdujemy?
  - Czy są lepsze od tych "łokciowych" (pod względem metryk, jak i "na oko")?

- Czy metoda znalazła oczekiwaną liczbę klastrów?

Zderzenie z rzeczywistością

Przetestujmy oba podejścia ("łokciowe" i "hierarchiczno-rekurencyjne") na bardziej rzeczywistych danych. W załączniku znajduje się plik CSV zawierający informacje o charakterystykach wirtualnych piłkarzy (opartych o dane znajdujące się na tej witrynie https://sofifa.com/player/158023/lionel-messi/). Naszym celem jest poklastrowanie zawodników na grupy o podobnej charakterystyce. W tym celu wykonujemy następujące kroki.

- Sprzątamy zbiór danych. Usuwamy z niego te kolumny, które nie pasują do problemu (albo zawierają już informacje o arbitralnym pogrupowaniu). Skupiamy się na tych cechach, które mają wartości liczbowe. Część z nich wymaga ponownego przeliczenia (bo jest np. w formacie 90+2).
- Oczyszczony zbiór dzielimy na grupy z użyciem obu przygotowanych wcześniej metod.
  - Na ile klastrów został podzielony?
  - Jakich zawodników zawierają te klastry (ustalamy to na podstawie położenia centrów i/lub nazwisk leżących w centrum klastra zawodników)?
  - Czy podział wydaje się mieć sens?

Materiały

1. https://github.com/rasbt/machine-learning-book/blob/main/ch10/ch10.ipynb
2. https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.11-K-Means.ipynb