Bootcamp Unsupervised learning

Unsupervised learning

Przemysław Spurek

k-means

k-means

Jak zobaczymy za chwilę, algorytm k-means jest niezwykle łatwy w implementacji, ale także bardzo wydajnym obliczeniowo algorytmem grupowania, co może wyjaśniać jego popularność.

k-means

W rzeczywistych zastosowaniach klastrowania (analizy skupień), nie wykorzystuje żadnej informacji na temat danych (w przeciwnym razie klastrowanie należałoby do kategorii nauczania nadzorowanego). Naszym celem jest grupowanie danych w oparciu o podobieństwo poszczególnych punktów.
`https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z01_introduction.ipynb`

k-means

Jednym z przykładów jest algorytm k-means, który można podsumować w czterech następujących krokach:

- Losowo wybierz k centroidów z punktów jako początkowe centroidy grup.
- Przypisz każdą próbkę do najbliższego centroidu $\mu^{(j)}$ dla $j \in 1, \ldots, k$.
- Wylicz nowe środki w nowo powstałych grupach.
- Powtórz kroki 2 i 3, dopóki:
  - przynależność do grup wszytkich punktów się nie zmienia,
  - lub zmiana funkcji kosztu jest mniejsza niż tolerancja zdefiniowana przez użytkownika,
  - lub zostanie osiągnięta maksymalna liczba iteracji.

k-means

Aby powyższy algorytm zadziałał musimy odpowiedzieć na następujące pytanie:

jak mierzymy podobieństwo między obiektami?

Możemy zdefiniować podobieństwo jako odległość, a powszechnie używaną odległością dla grupowania próbek o ciągłych cechach jest kwadrat odległość euklidesowej między dwoma punktami x i y w m–wymiarowej przestrzeni:

$$d^2(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{m} (x_j - x_j)^2 = \|\mathbf{x} - \mathbf{y}\|_2^2$$

k-means

Zauważ, że w poprzednim równaniu, indeks j odnosi się do j–tego wymiaru (kolumna cechy) przykładowych punktów x i y. W dalszej części używać będziemy indeksu górnego i oraz j, aby odnieść się odpowiednio do indeksu elementu danych i indeksu klastrów.

k-means

- k-means iteracyjnie minimalizuje **sum of squared errors (SSE)** wewnątrz klastrów:

$$SEE = \sum_{i=1}^{n} \sum_{j=1}^{k} w^{(i,j)} \|\mathbf{x}^{(i)} - \mu^{(j)}\|$$

gdzie $\mu^{(j)}$ jest reprezentantem (centroidem) klastra j, a $w^{(i,j)}$ jest indeksem przydziału do klastra:

  - $w^{(i,j)} = 1$ gdy element x_i **należy** do j-tego klastra,
  - $w^{(i,j)} = 0$ gdy element x_i **nie należy** do j-tego klastra.

`https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z02_sum_of_squared_error.ipynb`

k-means ++

- Do tej pory omówiliśmy klasyczny algorytm k-means, który wykorzystuje losowy wybór początkowych centroidów, co może skutkować złym klastrowaniem lub powolną zbieżnością (jeśli początkowe centroidy są źle dobrane).
- Jednym ze sposobów rozwiązania tego problemu jest wielokrotne uruchamianie algorytmu k-means i wybór najlepszego modelu pod względem SSE.
- Inną strategią jest umieszczenie początkowych centroidów z dala od siebie za pomocą algorytmu k-means++, co prowadzi do lepszych i bardziej spójnych wyników niż klasyczne k-means.

k-means++

Inicjalizację w **k-means++** można podsumować w następujący sposób:

- Zainicjalizuj pusty zbiór M aby móc w nim zapisać k wybranych centroidów.
- Losowo wybierz pierwszy centroid $\mu^{(j)}$ jako dowolny element z danych i umieść go w M
- Dla każdej próbki $x^{(j)}$, która nie znajduje się w M, znajdź minimalną odległość

$$d^2(\mathbf{x}^{(i)}, M)$$

do dowolnego z centroidów w M.

- Aby losowo wybrać następny centroid $\mu^{(p)}$, użyj ważonego rozkładu prawdopodobieństwa:

$$\frac{d^2(\mathbf{x}^{(p)}, M)}{\sum d^2(\mathbf{x}^{(i)}, M)}$$

- Powtarzaj kroki 2 i 3, aż wybrane zostaną wszystkie centroidy.
- Wykonaj klasyczny algorytm k-means z zadaną inicjalizacją

k-means

Aby użyć k-means++ w scikit-learn, wystarczy ustawić parametr init na k-means++ (ustawienie domyślne) zamiast losowego.

`https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z03_initialization.ipynb`

k-means

UWAGA

Algorytm k-means znajduje sferyczne klastry o podobnym promieniu.

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z04_data_whitening.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z05_imagesegmentation.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z06_mouse_effect.ipynb

k-means

- Ocena wydajności algorytmu grupowania nie jest tak prosta, jak w przypadku uczenia nadzorowanego.
- W szczególności każda metryka oceny nie powinna uwzględniać wartości bezwzględnych etykiet klastra
  - ale raczej podobieństwo do prawdziwych etykiet (w prawdziwych przypadkach ich nie znamy),
  - lub spełnianie pewnych założeń jak podobieństwo elementów w klastrach.

Adjusted Rand index

Biorąc pod uwagę wiedzę o prawdziwym przydziale do klas `labels_true` i wynik algorytmu klastrowania `labels_pred` , **Adjusted Rand index** mierzy podobieństwo tych dwóch przydziałów ignorując permutacje.
Jeśli C to prawdziwy przydział do klastrów, a K to wynik naszego algorytmu to:

- niech a będzie liczbą par elementów znajdujących się w tym samym zbiorze w C i tym samym zbiorze w K;
- niech b będzie liczbą par elementów znajdujących się w różnych zestawach w C i w różnych zestawach w K;

Adjusted Rand index

Rand index jest zdefiniowany jako:

$$RI = \frac{a+b}{C_2^{n_{samples}}}$$

gdzie $C_2^{n_{samples}}$ jest całkowitą liczbą możliwych par w zbiorze danych.

Adjusted Rand index

- Jednak wynik RI nie gwarantuje, że losowe przypisania etykiet będą miały wartość zbliżoną do zera (szczególnie, jeśli liczba klastrów jest jest tego samego rzędu wielkości co liczba próbek).
- Aby przeciwdziałać temu efektowi, możemy znormalizować wzór przez wartość oczekiwaną RI $E[RI]$ dla losowych przydziałów do grup i zdefiniować **Adjusted Rand index**:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

`https://en.wikipedia.org/wiki/Rand_index#Adjusted_Rand_index`

Adjusted Rand index

- Losowe przypisania etykiet daje wynik **Adjusted Rand index** bliski 0.0 dla dowolnej ilości grup i próbek.
- **Adjusted Rand index** jest ograniczony do $[-1, 1]$:
  - wartości ujemne są złe (niezależne oznaczenia),
  - podobne grupy mają dodatni **Adjusted Rand index**,
  - 1.0 oznacza doskonałe dopasowanie.
- Nie ma żadnych założeń na temat struktury klastra.
- **Adjusted Rand index** wymaga znajomości prawdziwego przydziału do grup.

Mutual Information based scores

Biorąc pod uwagę wiedzę o prawdziwym przydziale do klas `labels_true` i wynik algorytmu klastrowania `labels_pred` , **Mutual Information based scores** mierzy podobieństwo tych dwóch przydziałów ignorując permutacje. Dostępne są dwie różne wersje:

- Normalized Mutual Information (NMI),
- Adjusted Mutual Information (AMI).

Mutual Information based scores

Załóżmy, że mamy dwa przypisania etykiet U i V. Ich entropię definiujemy jako:

$$H(U) = -\sum_{i=1}^{|U|} P(i) \log(P(i)),$$

gdzie $P(i) = |U_i|/N$ jest prawdopodobieństwem losowego przydziału punktów z danych U do klasy U_i.
Podobnie dla V:

$$H(V) = -\sum_{j=1}^{|V|} P'(j) \log(P'(j)),$$

gdzie $P'(j) = |V_j|/N$.

Mutual Information based scores

Wzajemną informację (**Mutual Information** MI) między U i V oblicza się jako:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

gdzie $P(i,j) = |U_i \cap V_j|/N$ jest prawdopodobieństwem, że losowy punkt wpada jednocześnie do klas U_i oraz V_j.
Można ją również wyrazić jako:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N}\log\left(\frac{N|U_i \cap V_j|}{|U_i||V_j|}\right)$$

Mutual Information based scores

Normalized Mutual Information (NMI) jest zdefiniowana jako

$$\mathrm{NMI}(U, V) = \frac{\mathrm{MI}(U, V)}{\sqrt{H(U)H(V)}}$$

Wartość **Mutual Information** oraz **Normalized Mutual Information** będzie się zwiększać wraz ze wzrostem liczby różnych etykiet klastrów niezależnie od "wzajemnej informacji"pomiędzy przypisaniami do etykiet.

Mutual Information based scores

Korzystając z wartości oczekiwanej możemy wyliczyć **Adjusted Mutual Information** (AMI):

$$AMI = \frac{MI - E[MI]}{\max(H(U), H(V)) - E[MI]}$$

Mutual Information based scores

- Losowe przypisania etykiet daje wynik **Adjusted Mutual Information** bliski 0.0 dla dowolnej ilości grup i próbek.
- **Adjusted Mutual Information** jest ograniczony do $[-1, 1]$:
  - wartości ujemne są złe (niezależne oznaczenia),
  - podobne grupy mają dodatni **Adjusted Mutual Information**,
  - 1.0 oznacza doskonałe dopasowanie.
- Nie ma żadnych założeń na temat struktury klastra:
- **Adjusted Mutual Information** wymaga znajomości prawdziwego przydziału do grup.

Inne miary

http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation.ipynb

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z07_clustering_evaluation_iris.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z08_clustering_evaluation_MNIST.ipynb

Optymalna ilość klastrów

- Jednym z głównych wyzwań grupowanie danych jest fakt, że nie znamy prawdziwych etykiet.
- W naszym zbiorze danych nie ma prawdziwych etykiet, które pozwalają nam zastosować techniki analogiczne do tych używanych w nauczaniu nadzorowanym.
- W celu określenia jakości grupowania danych, musimy użyć charakterystyk wewnętrznych - takich jak SSE, które omówiliśmy wcześniej.

Optymalna ilość klastrów

- W oparciu o SSE i narzędzia graficzne możemy oszacować optymalną ilość klastrów metodą: **elbow method**.
- Wiemy już, że jeśli k wzrasta, to SSE będzie systematycznie maleć. (Dzieje się tak dlatego, że próbki będą bliżej centroidów.)

Ideą **elbow method** jest określenie wartości k jako miejsca, w którym SSE zaczyna wzrastać najszybciej.

- `https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z09_elbow_method_iris.ipynb`
- `https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z10_elbow_method_MNIST.ipynb`

Silhouette plots

- Inną wewnętrzną miarą oceny jakości grupowania jest **silhouette analysis**, którą można również zastosować do algorytmów grupowania innych niż k-means.
- **Silhouette analysis** może być wykorzystywana jako narzędzie graficzne do określenia, które elementy w grupach są blisko siebie.
- Aby obliczyć **silhouette coefficient** pojedynczej próbki w naszym zbiorze danych, możemy zastosować następujące trzy kroki:
  - Oblicz spójność klastra $a^{(i)}$ jako średnią odległość między punktem x_i, a wszystkimi innymi punktami w tym samym klastrze.
  - Oblicz poziom odseparowania klastra $b^{(i)}$ od najbliższego klastra jako średnią odległość między punktem x_i, a wszystkimi próbkami w najbliższej grupie.
  - Oblicz **silhouette** $s^{(j)}$ jako różnicę między spójnością, a separowalnością:

$$s^{(j)} = \frac{b^{(j)} - a^{(j)}}{\max(b^{(j)}, a^{(j)})}$$

Silhouette plots

- **Silhouette coefficient** zawiera się w zakresie od -1 do 1.
- Na podstawie powyższego wzoru widać, że **silhouette coefficient** wynosi 0 jeśli $b^{(j)} = a^{(j)}$.
- Co więcej, zbliżamy się do idealnego współczynnika **silhouette coefficient** równego 1, jeżeli $(b^{(i)} >> a^{(i)})$, ponieważ $b^{(i)}$ określa jak niepodobna jest próbka do innych klastrów, $a^{(i)}$ mówi nam jak podobna jest do innych próbek we własnym klastrze.

Silhouette plots

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z11_silhouettecoefficient.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z12_silhouettecoefficient_iris.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z14_silhouettecoefficient_MNIST.ipynb

k-means

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z15_dimension_reduction_MNIST.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z16_customer_segmentation.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z17_ensamble.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D01_Z18_kmeans_data_representation.ipynb

Hierarchical clustering

Hierarchical clustering

- Teraz przyjrzymy się alternatywnemu podejściu do klastrowania: **klastrowanie hierarchiczne**.
- Jedną z zalet hierarchicznego grupowania danych jest to, że pozwala nam na narysowanie dendrogramów (wizualizacji binarnej hierarchi), co może pomóc w interpretacji wyników.
- Kolejną przydatną zaletą hierarchicznego podejścia jest to, że nie musimy określać z góry liczby klastrów.

Hierarchical clustering

Dwa główne podejścia do hierarchicznego tworzenia klastrów to:

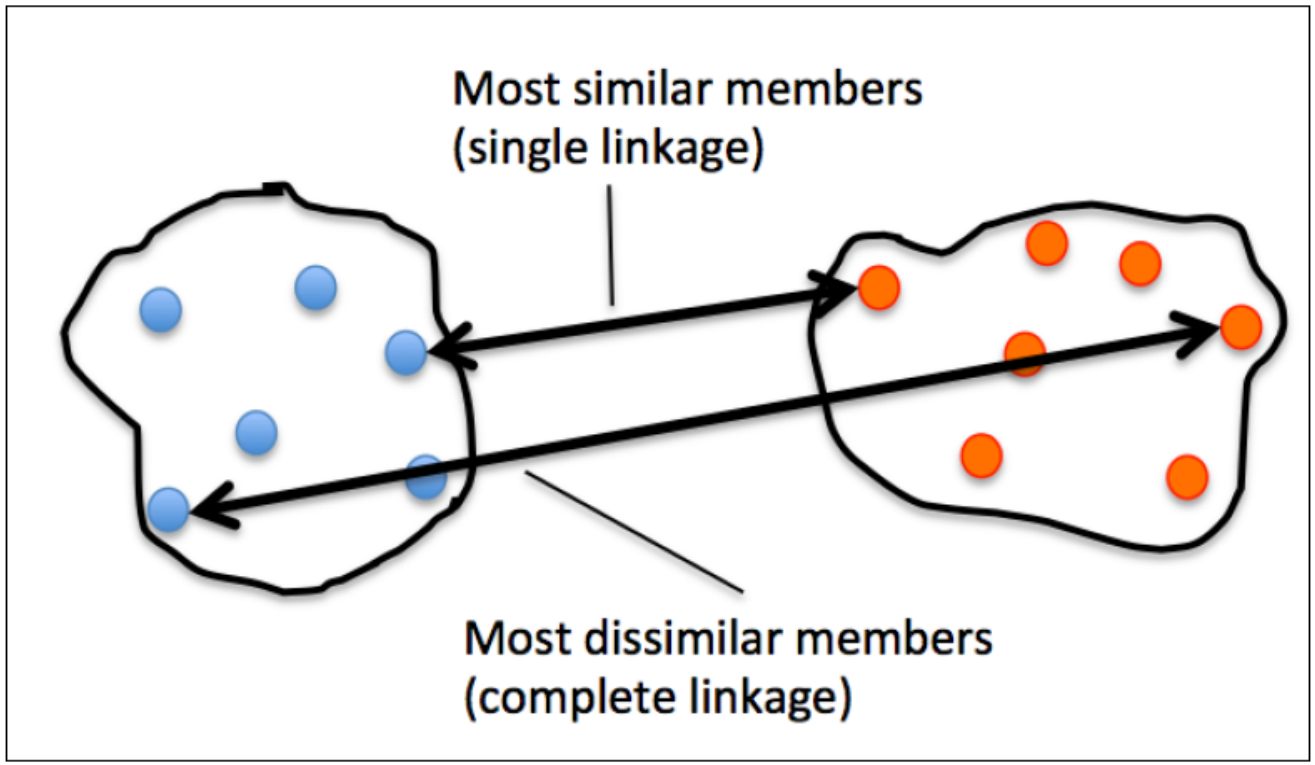
- podejście **divisive** - rozpoczynamy od jednego klastra obejmującego wszystkie nasze próbki, a następnie iteracyjnie dzielimy klaster na mniejsze klastry, aż każdy klaster zawiera tylko jeden punkt.
- podejście **agglomerative** - zaczynamy od każdej próbki jako pojedynczego klastra i łączymy najbliższe pary klastrów, aż pozostanie tylko jeden klaster.

Hierarchical clustering

Dwa standardowe algorytmy grupowania hierarchicznego to:

- **single linkage** - obliczamy odległości między najbardziej podobnymi elementami dla każdej pary klastrów i łączymy dwa klastry, dla których odległość między najbardziej podobnymi elementami jest najmniejsza.
- **complete linkage** - zamiast porównywać najbardziej podobne elementy w każdym zbiorze, porównujemy najbardziej odmienne elementy.

Hierarchical clustering



Hierarchical clustering

Algorytm jest bardzo prosty. Opiszmy jedną z metod np.**complete linkage**. Jest to procedura iteratywna, którą można przedstawić w kilku krokach:

- 1. Oblicz macierz odległości pomiędzy wszystkimi punktami.
- 2. Ustaw każdy punkt jako pojedynczy klaster.
- 3. Połącz dwie najbliższe grupy w oparciu o odległość najbardziej odległych punktów w klastrze.
- 4. Zaktualizuj macierz podobieństwa.
- 5. Powtarzaj kroki od 2 do 4, aż pozostanie jeden pojedynczy klaster.

Hierarchical clustering

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D02_Z01_hierarchical_clustering_introduction.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D02_Z02_hierarchical_clustering_linkage.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D02_Z03_hierarchical_clustering_elbow_method.ipynb

Hierarchical clustering

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D02_Z04_hierarchical_text_data.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D02_Z04_hierarchical_text_data.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D02_Z05_customer_segmentation.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D02_Z06_eurowizja_data.ipynb

DBSCAN

DBSCAN

Density-based Spatial Clustering of Applications with Noise (DBSCAN)

Pojęcie gęstości w DBSCAN definiuje się jako liczbę punktów w określonym otoczeniu.

DBSCAN

W DBSCAN dla każdego punktu przypisywana jest etykieta:

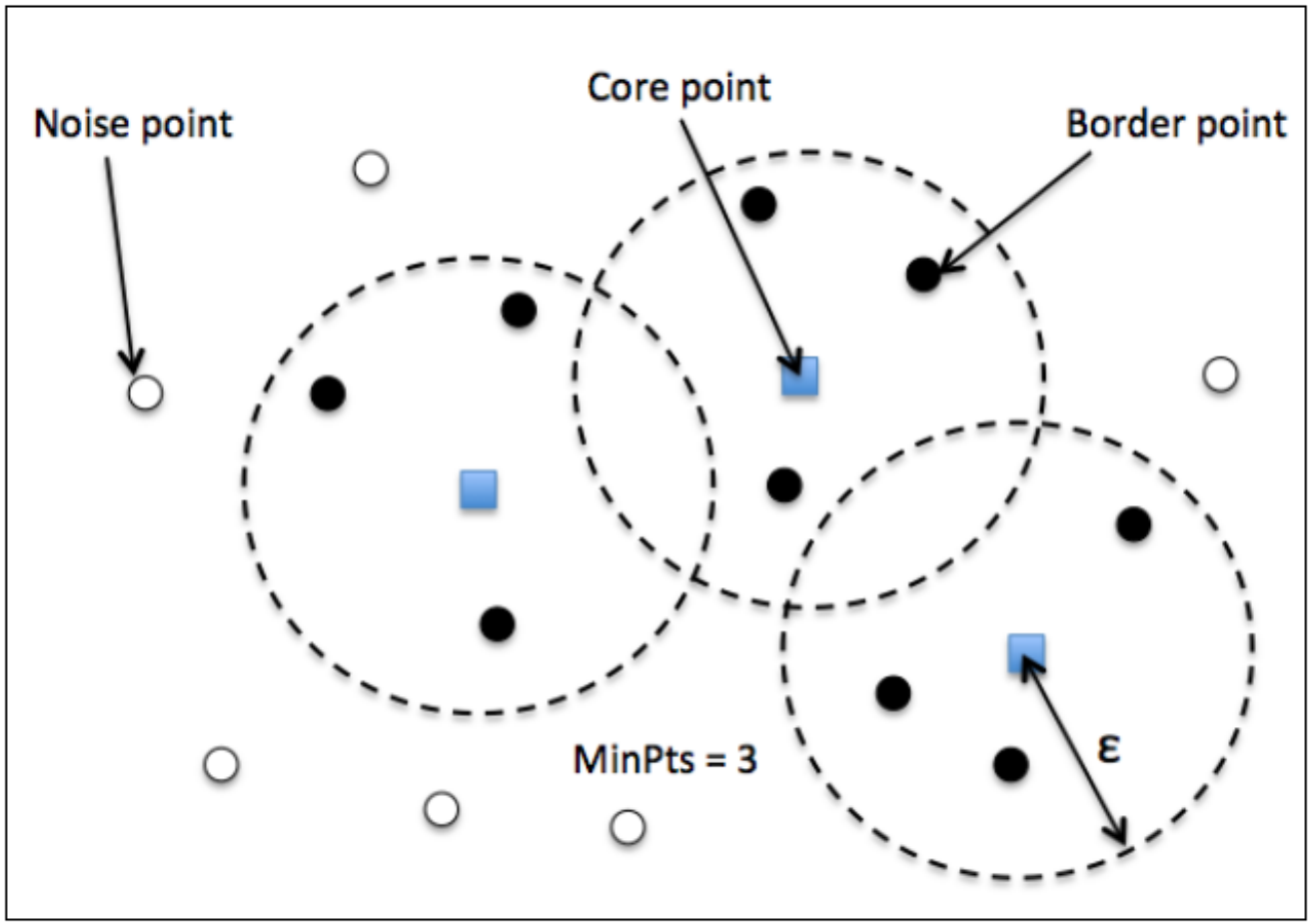
- punkt jest uważany za punkt centralny (**core point**), jeśli przynajmniej `MinPts` sąsiednich punktów mieści się w otoczeniu o promieniu ε;
- punkt jest uważany za punkt graniczny (**border point**), jeśli w jego otoczeniu o promieniu ε jest mniej sąsiadów niż w `MinPts`, ale w jego otoczeniu znajduje się **core point**;
- wszystkie inne punkty są uznawane za szum (**noise points**).

DBSCAN

Po oznaczeniu wszytkich punktów jako **core point**, **border point** lub **noise points** algorytm DBSCAN opiera się na:

- Utwórz osobny klaster dla każdego punktu centralnego lub połączonej grupy punktów centralnych (punkty centralne są połączone, jeśli nie są dalej niż ε).
- Przypisz każdy punkt graniczny do klastra odpowiadającego najbliższemu punktowi centralnemu.

DBSCAN



DBSCAN

- Jedną z głównych zalet algorytmu DBSCAN jest to, że nie zakłada on, że klastry mają sferyczny kształ jak w k-means.
- Co więcej, DBSCAN różni się od k-means i hierarchicznego klastrowania tym, że niekoniecznie przypisuje każdy punkt do klastra, ale jest zdolny do usuwania punktów należących do szumu.

DBSCAN

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D03_Z01_DBSCAN_introduction.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D03_Z02_DBSCAN_params.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D03_Z03_DBSCAN_map.ipynb.ipynb

BIRCH

balanced iterative and clustering using hierarchies

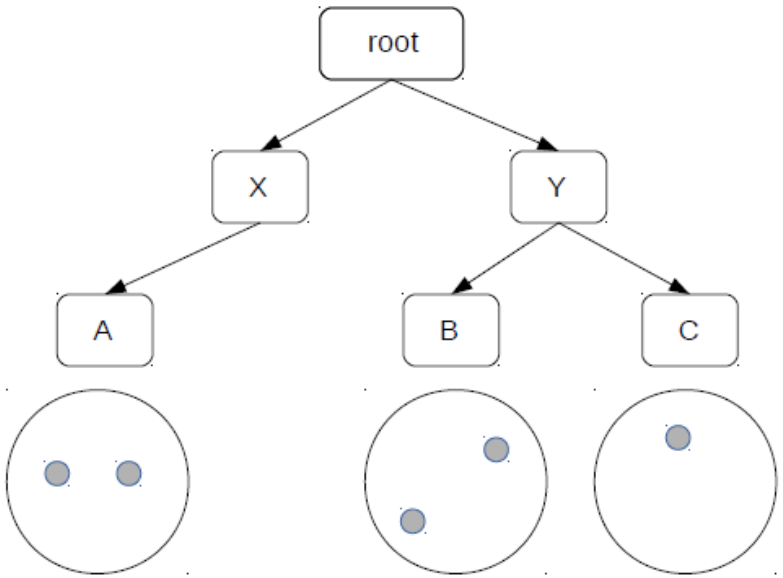
BIRCH

- Algorytm ma na celu wstępne pogrupowanie danych na dużo małych i bardzo jednorodnych skupień.
- Następnie otrzymane klastry poddaje się grupowaniu (dowolnym algorytmem), gdzie nowymi obserwacjami są środki skupień.
- Algorytm jest bardzo wydajny – wymaga tylko jednego przejścia po danych.

BIRCH

BIRCH

Algorytm polega na zbudowaniu drzewa wyznaczajcego klastry.



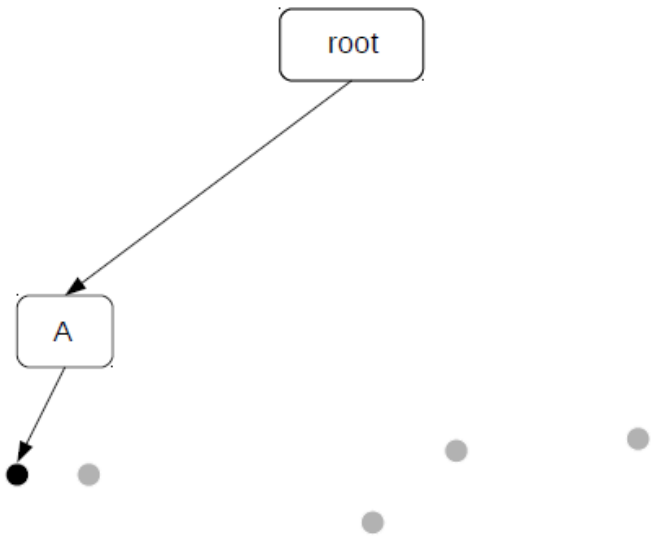
BIRCH

Przyjmijmy B = 2.

Algorytm przechodzi jednokrotnie po zbiorze danych. Zaczynamy od pierwszej obserwacji.

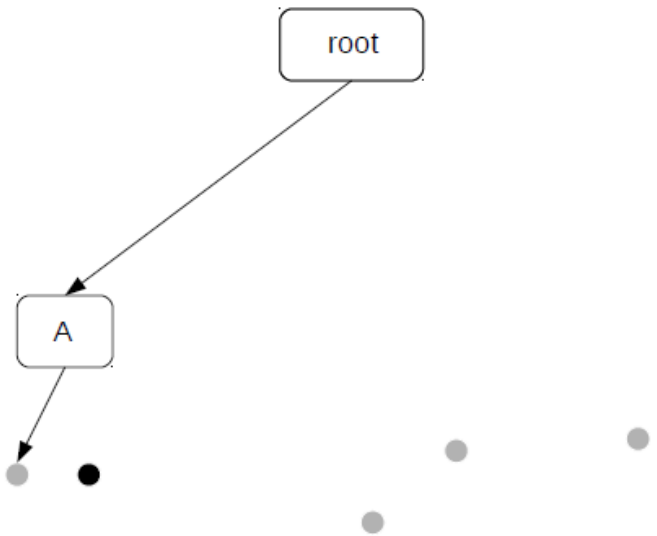
BIRCH

Inicjujemy korzeń drzewa oraz pierwszy klaster - A.



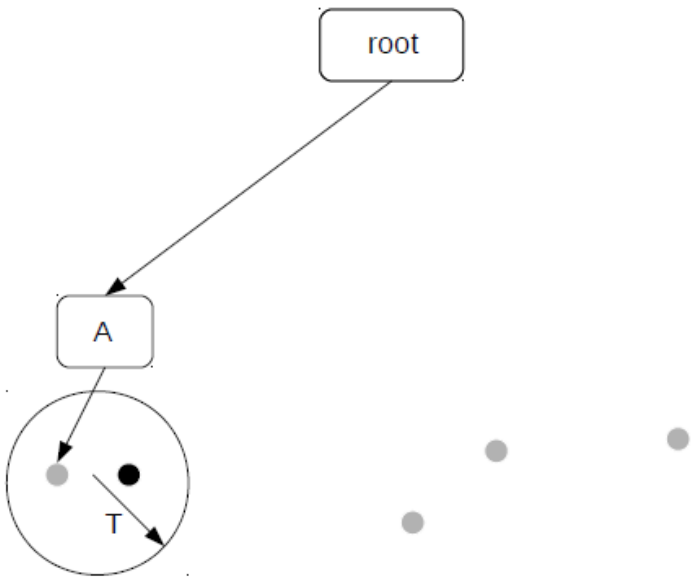
BIRCH

Rozpatrujemy drugi punkt. Idąc po drzewie od korzenia, szukamy najbliższego klastra.

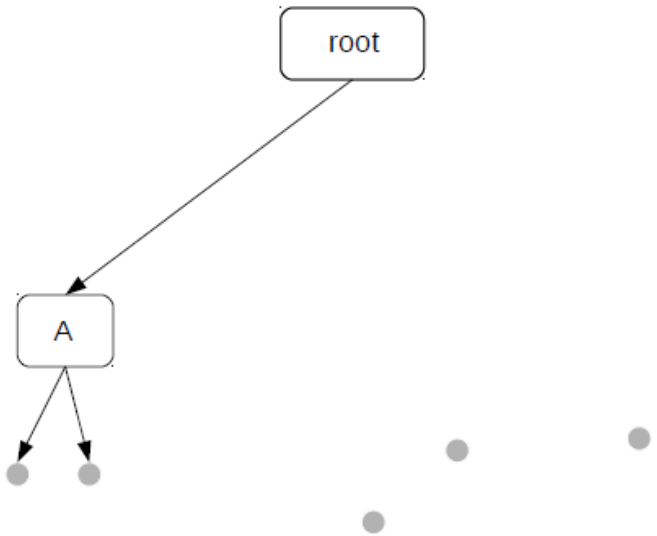


BIRCH

Jest to A (bo jest jedyny). Sprawdzamy czy promień klastra powstałego z połączenia A i nowej obserwacji nie przekroczy T.

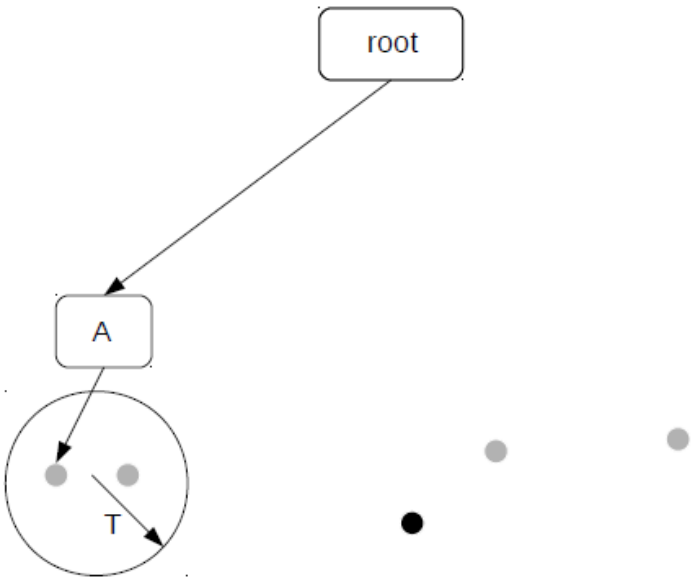


Nie przekroczy. Zatem dołączamy obserwację do A.

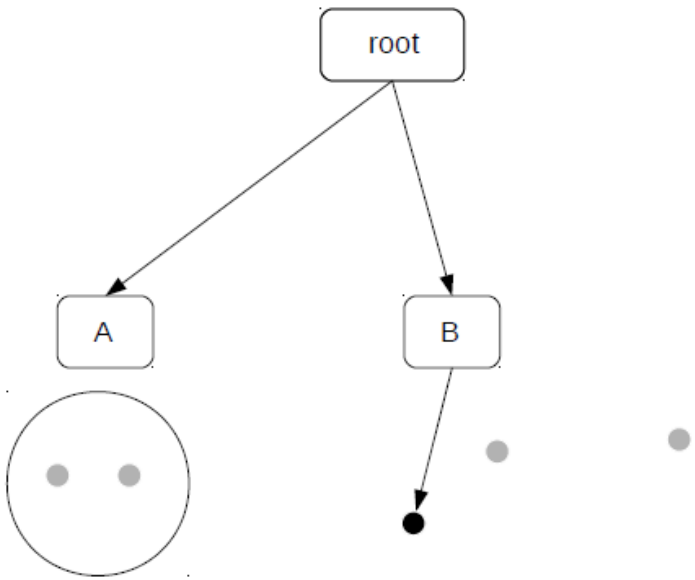


BIRCH

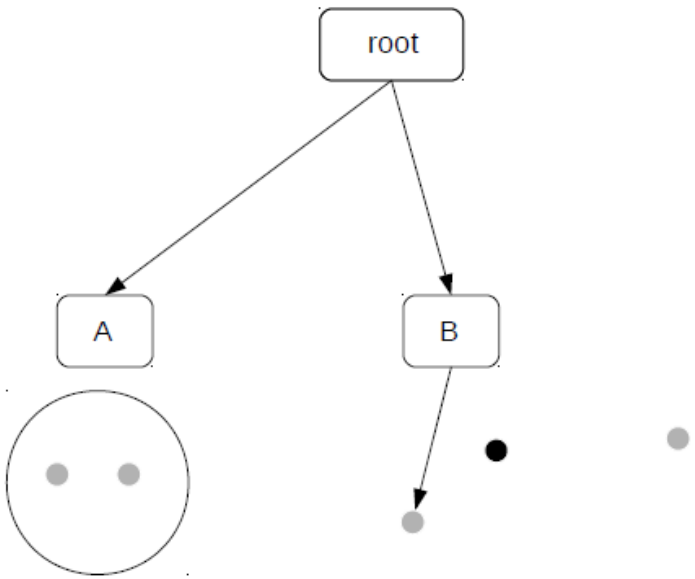
Rozpatrujemy kolejny punkt. Szukamy najbliższego klastra i sprawdzamy czy jest wystarczająco blisko (czy promień po dołączeniu nie przekroczy T).



Nie jest – zatem tworzymy nowy klaster.

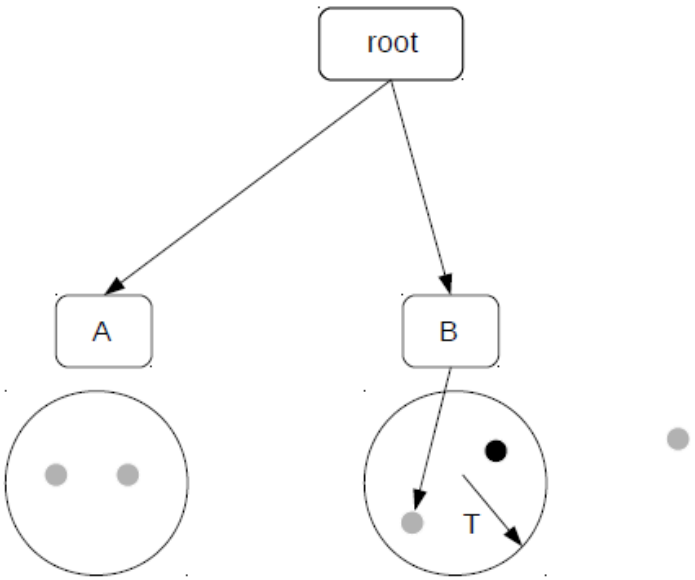


Rozpatrujemy kolejny punkt. Szukamy najbliższego klastra – jest to B.



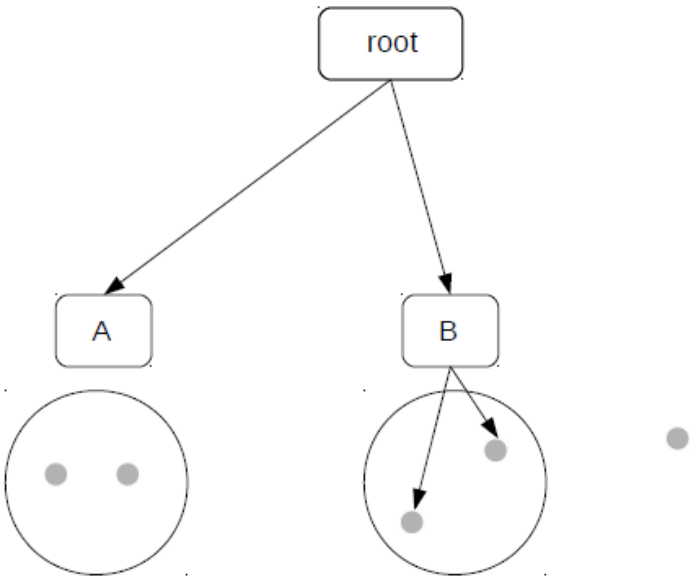
BIRCH

Sprawdzamy czy jest wystarczająco blisko.

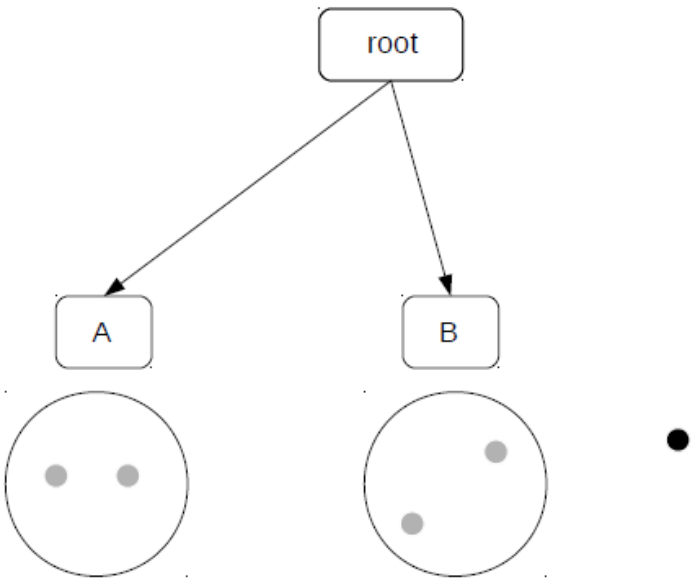


BIRCH

Tak – zatem dołączmy obserwację do B.

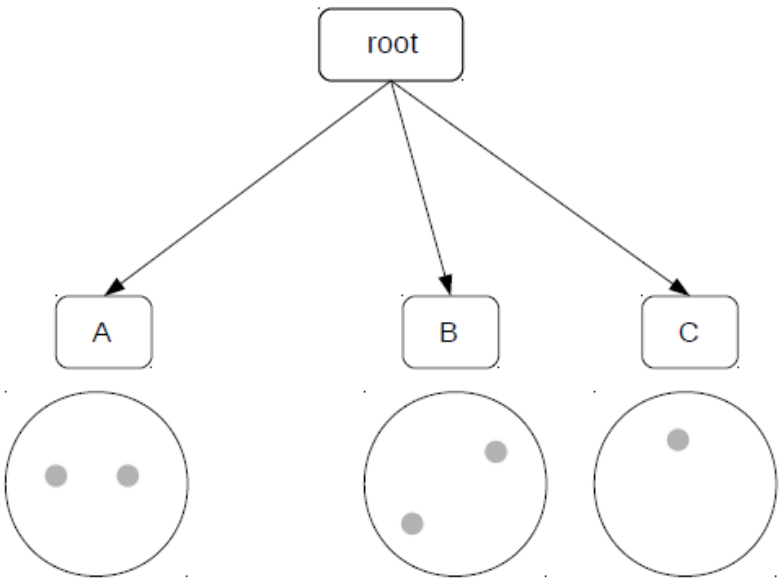


Rozpatrujemy kolejny punkt.



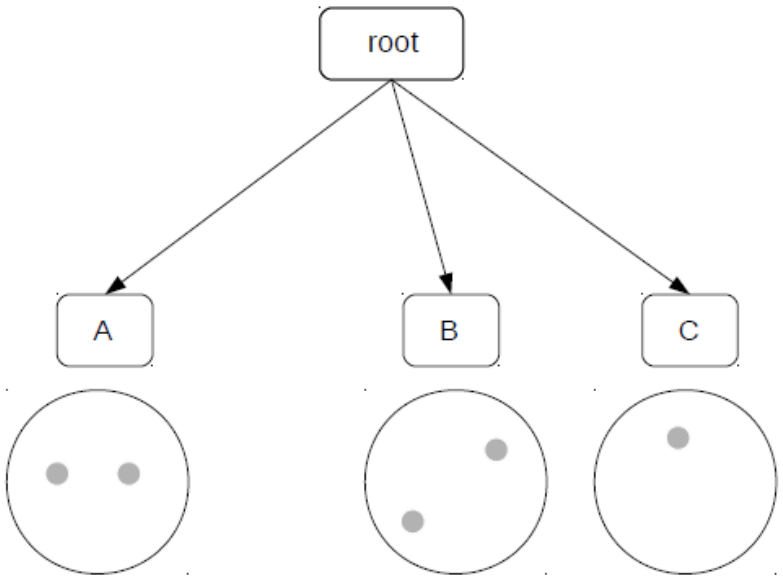
BIRCH

Najbliższy klaster (B) jest za daleko – tworzymy nowy klaster.



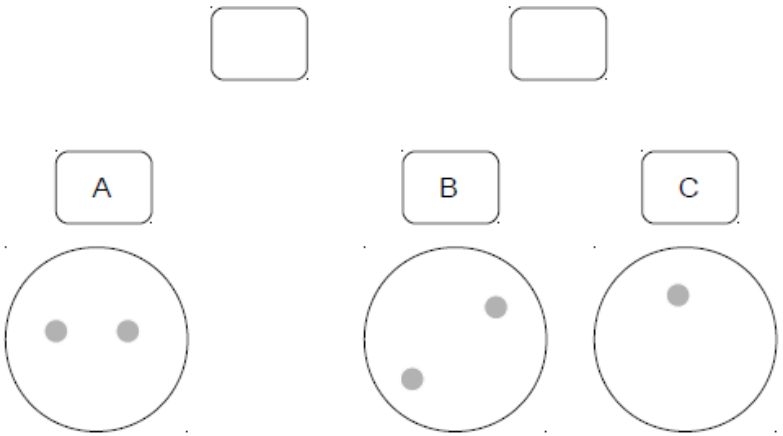
BIRCH

Ale… Z wierzchołka “root” wychodzą trzy gałęzie – przekroczyliśmy B = 2. Trzeba przebudować drzewo.



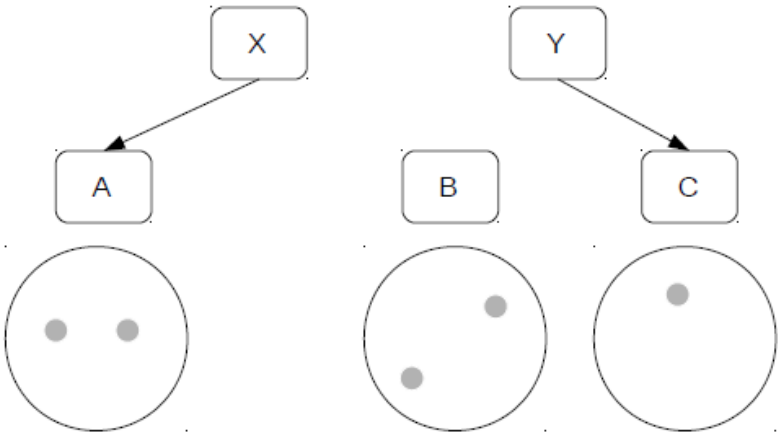
BIRCH

Rozbijamy wierzchołek, z którego wychodzi za dużo gałęzi na dwa i odbudowujemy drzewo od dołu.



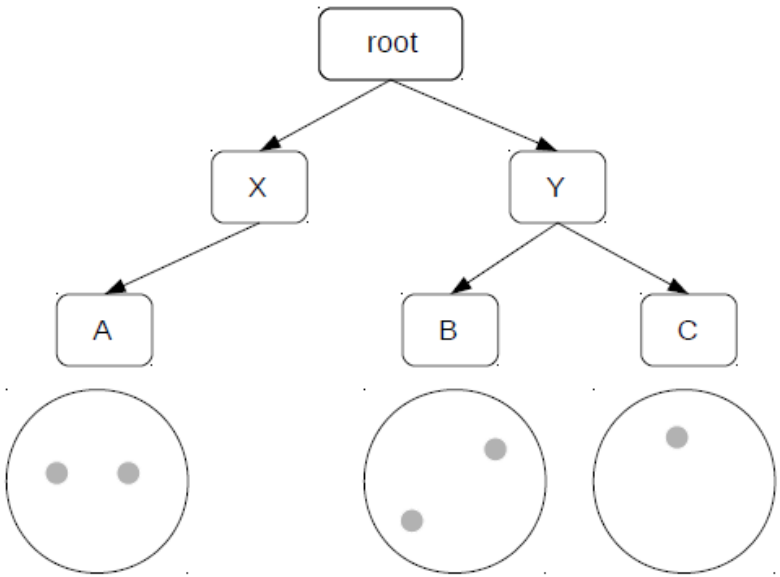
BIRCH

Bierzemy dwa najbardziej oddalone od siebie klastry (tutaj A i C) I przyporządkowujemy je do osobnych węzłów (subklastrów).



BIRCH

I odbudowujemy drzewo do korzenia (aktualizujemy strukturę nad miejscem rozpadu).



BIRCH

Każdy subklaster (węzeł w drzewie) jest opisany trzema wielkościami:

- N – liczba punktów w subklastrze,
- LS – suma punktów - wektor sum po współrzędnych,
- SS – suma kwadratów punktów – wektor sum kwadratów po współrzędnych.

Na podstawie tych wartości wyliczamy wielkości potrzebne do przydzielenia nowego punktu do klastra i aktualizacji struktury drzewa.
To decyduje o niskiej złożoności algorytmu.

Mini Batch K-means

- Algorytm pobiera małe partie zbioru danych (losowo wybrane) dla każdej iteracji.
- Następnie przypisuje klaster do każdego punktu danych w grupie, w zależności od poprzednich lokalizacji centroidów klastra.
- Następnie aktualizuje położenie centroidów klastra na podstawie nowych punktów z partii.
- Aktualizacja wykonywana jest za pomocą metody gradientowej, która jest znacznie szybsza niż zwykła aktualizacja Batch K-Means.

Mini Batch K-means

Formalnie algorytm wygląda następująco:

```
Given: k, mini-batch size b, iterations t, data set X
Initialize each c ∈ C with an x picked randomly from X
v ← 0
for i = 1 to t do
        M ← b examples picked randomly from X
        for x ∈ M do
                d[x] ← f (C, x)              // Cache the center nearest to x
        end for
        for x ∈ M do
                c ← d[x]                     // Get cached center for this x
                v[c] ← v[c] + 1              // Update per-center counts
                η ← 1 / v[c]                 // Get per-center learning rate
                c ← (1 – η)c + ηx            // Take gradient step
        end for
end for
```

http://www.eecs.tufts.edu/~dsculley/papers/fastkmeans.pdf

BIRCH

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D04_Z01_BIRICH.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D04_Z02_mini_batch_k_means.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D04_Z03_mini_batch_k_means_vs_k_means.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D04_Z04_mini_batch_BIRICH.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D04_Z05_mini_batch_BIRICH.ipynb

Zadania

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D05_Z01_faces_dataset.ipynb

Zadania

- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D05_Z02.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D05_Z03.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D05_Z04.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D05_Z05.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D05_Z06.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D05_Z07.ipynb
- https://github.com/przem85/bootcamp/blob/master/unsupervised_learning/D05_Z08.ipynb