Uniwersytet Jagielloński
Wydział Matematyki i Informatyki

Piotr Helm
Nr albumu: 1132708

Analiza algorytmów budowania coresetu dla problemu K-means

Praca licencjacka
na kierunku Informatyka Analityczna

Praca wykonana pod kierunkiem
dr Iwona Cieślik
Instytut Informatyki Analitycznej

Kraków, wrzesień 2020

Streszczenie

W pracy przedstawię stan wiedzy na temat budowania coresetów w kontekście algorytmu K-means. W szczególności poruszę techniki takie jak geometryczna dekompozycja oraz losowe próbkowanie bazując na badaniach [8].

Celem pracy jest przedstawienie wyników teoretycznych [8] jak i implementacja technik budowania coresetów, które mogą mieć zastosowanie praktyczne [3] [6]. Następnie porównam ich działanie testując je na zbiorach punktów z dwuwymiarowej przestrzeni euklidesowej.

1. Wstęp

More is more to jedna z podstawowych doktryn związanych z szeroko rozumiałym Big Data. Więcej danych to więcej informacji, które analizujemy licząc na poznanie ukrytych zależności. W erze Big Data skalowalność rozwiązań jest szczególnie ważna dlatego celem wielu naukowców jest dostarczenie kompromisu pomiędzy szczegółowością informacji a wymaganiami pamięciowymi. Tutaj warto zwrócić uwagę na dużą wartość takich rozwiązań w praktycznych zastosowaniach.

Sketch-and-solve to popularny paradygmat, który zakłada separacje algorytmu agregującego dane od właściwego algorytmu analizującego. Główną ideą jest redukcja danych tak aby ich rozmiar nie był zależny od wejściowych danych lub tylko *trochę* od nich zależał. Następnie aplikowany jest właściwy algorytm, który jest mniej zależy od początkowego rozmiaru danych. W rezultacie wykonuje swoją pracę szybciej, a niekiedy nawet lepiej. Dodatkową zaletą jest fakt, że w większości przypadków nie jest konieczna modyfikacja algorytmu analizującego.

Niestety w kontekście tego paradygmatu największym wyzwaniem jest znalezienie kompromisu pomiędzy stratą jakości danych a ich rozmiarem. To jak określamy charakterystykę ważnych informacji jest ściśle zależne od aplikacji danych. Coresety są strukturą algorytmiczną, która ma na celu indentyfikacje takich cech oraz określenie akceptowalnego kompromisu dla różnych funkcji celu.

Mówiąc ogólniej, mamy na wejściu zbiór danych A, zbiór potencjalnych rozwiązań C oraz funkcję celu f. Chcemy znaleźć istotnie mniejszy zbiór danych S, który dla każdego potenlego rozwiązania $c \in C$ daje $f(S, c)$ *dobrze* aproksymujące $f(A, c)$.

Algorytmy budujące coresety są aplikowalne do wielu problemów klasteryzacji. W tej pracy skupię się na konstrukcjach dla problemu K-means, który należy do klasy problemów *NP-trudnych*. Najpopularniejszym algorytmem heurystycznym dla tego problemu jest algorytm Lloyd'a. Z uwagi na jego niską skalowalność jest on idealnym kandydatem do optymalizacji poprzez odpowiednią konstrukcję coresetu.

2. Notacja i niezbędne definicje

2.1. K-means

Zacznijmy od zdefiniowania problemu dla którego będziemy analizować konstrukcje coresetów.

Definicja 2.1. *Problem K-means.* Niech X to zbiór punktów z $\mathbb{R}^d$. Dla danego X chcemy znaleźć zbiór k punktów Q, który minimalizuje funkcję $\phi_X(Q)$ zdefiniowaną następująco:

$$\phi_X(Q) = \sum_{x \in X} d(x, Q)^2 = \sum_{x \in X} \min_{q \in Q} ||x - q||_2^2$$

Powyższa definicja zakłada, że działamy w przestrzeni euklidesowej. Uogólnioną wersję można zdefiniować analogicznie, zamieniając d na odpowiednią funkcję miary w danej przestrzeni.

Definicja 2.2. *Problem K-means - wersja ważona.* Niech C to zbiór punktów z $\mathbb{R}^d$ oraz niech w będzie funkcją $C \to \mathbb{R}_{\geq 0}$. Dla danego X oraz funkcji w chcemy znaleźć zbiór k punktów Q, który minimalizuje funkcję $\phi_X(Q)$ zdefiniowaną następująco:

$$\phi_X(Q) = \sum_{x \in X} w(x) d(x, Q)^2$$

Ewaluację funkcji ϕ dla optymalnego rozwiązania oznaczamy $\phi^k_{OPT}(X)$. Funkcję ϕ w literaturze nazywamy błędem kwantyzacji.

2.2. Coreset

To jak definujemy coreset ściśle zależy od problemu, który optymalizujemy. Zacznijmy od podstawowej definicji coresetu dla problemu *K-means*.

Definicja 2.3. *Coreset.* Niech X to zbiór punktów z $\mathbb{R}^d$ oraz niech Q to dowolny podzbiór X rozmiaru co najwyżej k. Zbiór C nazywamy (ϵ, k) coresetem jeżeli zachodzi:

$$|\phi_X(Q) - \phi_C(Q)| \leq \epsilon \phi_X(Q)$$

Zauważmy, że taka definicja daje nam bardzo moce gwarancje teoretyczne. Ewaluacji coresetu $\phi_C(Q)$ musi aproksymować $\phi_X(Q)$ z dokładnością $1+\epsilon$ jednocześnie dla dowolnego zbioru k kandydatów na rozwiązanie Q w kontekście całego zbioru X. Jest to na tyle istotne, że w literaturze odróżnia się tą wersję nazywając ją *Strong Coreset*.

Definicja 2.4. *Coreset - weak.* Niech X to zbiór punktów z $\mathbb{R}^d$ oraz niech Q będzie optymalnym rozwiązaniem. Słabym coresetem nazywamy zbiór C dla którego zachodzi:

$$|\phi_X(Q) - \phi_C(Q)| \leq \epsilon\phi_X(Q)$$

3. Lightweight Coreset

Budowa coresetów w kontekście problemu *K-means* ma bardzo długo historię. Za przełomową pracę uznaje się [7], która jako pierwsza pokazuje konstrukcje $(1+\epsilon)$ aproksymacji dla problemu *K-means*. Bazuje ona na budowie zbioru centroidów, czyli geometrycznym wariancie coresetu. Wyróżnia się trzy techniki budowania coresetów.

- Geometryczna dekompozycja problemu.
- Losowe próbowanie zbioru.
- Zawansowane metody algebraiczne.

Pierwsza z technik cechuje się mocnymi gwarancjami teoretycznimi. Niestety większość rozwiązań jest mało praktyczna i kosztowna czasowo. Losowe próbkowanie w praktyce daje bardzo przyzwoite wyniki jednak nie daje nam żadnej gwarancji odnośnie optymalności rozwiązania. Autorzy pracy [3] zaproponowali rozwiązanie o nazwie *lightweight coreset*, które w swoich założeniach ma łączyć:

- Prostą implementacje.
- Gwarancje teoretyczne.
- Szybkie działanie oparte na próbkowaniu zbioru danych.

3.1. Lightweight coreset

Zacznijmy od wproadzenia definicji *lightweight coresetu*.

Definicja 3.1. *Lightweight coreset dla problemu K-means.* Niech $\epsilon > 0$ oraz $k \in \mathbb{N}$. Niech X to zbiór punktów z $\mathbb{R}^d$ wraz ze średnią $\mu(X)$. Zbiór C jest (ϵ, k) lightweight coresetem dla X, jeżeli dla dowolnego zbioru $Q \subset X$ o mocy co najwyżej k zachodzi:

$$|\phi_X(Q) - \phi_C(Q)| \leq \frac{\epsilon}{2}\phi_X(Q) + \frac{\epsilon}{2}\phi_X(\mu(X))$$

Jak możemy zauważyć definicja (3.1) trochę się różni od (2.2). Notacje *lightweight* coresetu możemy interpretować jako relaksację gwarancji teoretycznych zdefiniowanych w (2.2). Wprowadza ona oprócz błędu multiplikatywnego, błąd

addytywny. Składnik $\frac{\epsilon}{2}\phi_X(Q)$ pozwala na odpowiednie skalowanie błędu aproksymacji dla funkcji ϕ. Druga część $\frac{\epsilon}{2}\phi_X(\mu(X))$ odpowiada za skalowalność rozwiązania zgodnie z wariancją dany.

Głowną motywacją stojącą za konstrukcjami coresetów jest to, żeby rozwiązanie obliczone na tym zbiorze było konkurencyjne z rozwiazaniem optymalnym dla całego zbioru danych. Dlatego w konkeście *lightweight* udowodnię następujące twierdzenie.

Twierdzenie 3.1. *Niech $\epsilon \in (0, 1]$. Niech X będzie dowolnym zbiorem dany oraz niech C będzie (ϵ, k) lightweight coresetem dla X. Optymalne rozwiązanie problemu K-means dla X oznaczamy Q_X^*. Optymalne rozwiązanie problemu K-means dla C oznaczamy Q_C^*. Dla takich założeń zachodzi:*

$$\phi_X(Q_C^*) \leq \phi_X(Q_X^*) + 4\epsilon\phi_X(\mu(X))$$

Dowód. Zgodnie z własnością lightweight coresetu otrzymujemy:

$$\phi_C(Q_X^*) \leq (1 + \frac{\epsilon}{2})\phi_X(Q_X^*) + \frac{\epsilon}{2}\phi_X(\mu(X))$$

oraz

$$\phi_C(Q_C^*) \geq (1 - \frac{\epsilon}{2})\phi_X(Q_C^*) - \frac{\epsilon}{2}\phi_X(\mu(X))$$

Wiemy z definicji, że $\phi_C(Q_C^*) \leq \phi_C(Q_X^*)$ oraz $1 - \frac{\epsilon}{2} \geq \frac{1}{2}$. A więc:

$$\phi_X(Q_C^*) \leq \frac{1 + \frac{\epsilon}{2}}{1 - \frac{\epsilon}{2}}\phi_X(Q_X^*) + \frac{\epsilon}{1 - \frac{\epsilon}{2}}\phi_X(\mu(X))$$

$$\leq (1 + 2\epsilon)\phi_X(Q_X^*) + 2\epsilon\phi_X(\mu(X))$$

Zauważając, że:

$$\phi_X(Q_X^*) \leq \phi_X(\mu(X))$$

dowodzimy tezę twierdzenia. □

Twierdzenie 1 dowodzi, że kiedy wartość ϵ maleje koszt optymalnego rozwiązania otrzymanego na zbiorze C zbiega do kosztu rozwiązania otrzymanego na całym zbiorze danych.

3.2. Konstrukcja

Konstrukcja oparta jest na próbkowaniu z uwzględnieniem ważności danego punktu. Niech $q(x)$ będzie dowolnym rozkładem prawdopodobieństwa na zbiorze X oraz niech $Q \subset R^d$ będzie dowolnym potencjalnym zbiorem rozwiązań mocy k. Wtedy funkcję ϕ możemy zapisać jako:

$$\phi_X(Q) = \sum_{x \in X} q(x)\frac{d(x, Q)^2}{q(x)}$$

Wynika z tego, że funkcja ϕ może być aproksymowana poprzez wylosowanie m punktów z X korzystając z $q(x)$ i przypisując im wagi odwrotnie proporcjonalne do $q(x)$. Dla dowolnej liczby próbek m oraz dla dowolnego rozkładu $q(x)$

możemy otrzymać sprawiedliwy (unbiased) estymator dla funkcji ϕ. Niestety, nie jest to wystarczające aby spełnić definicję (3.1). W szczególności musimy zagwarantować, jednostajność wyboru dowolnego zbioru k punktu Q z odpowiednim prawdopodobieństwem $1 - \delta$. Funkcja $q(x)$ może mieć wiele form, autorzy rekomendują postać:

$$q(x) = \frac{1}{2}\frac{1}{|X|} + \frac{1}{2}\frac{d(x,\mu)^2}{\sum_{x' \in X} d(x',\mu)^2}$$

Algorithm 1

procedure LIGHTWEIGHT ▷ Require: Set of data points X, coreset size m
$\mu \leftarrow$ mean of X
for $x \in X$ **do**
$q(x) = \frac{1}{2}\frac{1}{|X|} + \frac{1}{2}\frac{d(x,\mu)^2}{\sum_{x' \in X} d(x',\mu)^2}$
$C \leftarrow$ sample m weighted points from X where each point x has weight $\frac{1}{mq(x)}$ and is sampled with probability $q(x)$
return lightweight coreset C

Pierwszy składnik rozkładu $q(x)$ to rozkład jednostajny, który zapewnia, że każdy punkt jest wylosowany z niezerowym prawdopodobieństwem. Drugi składnik uwzględnia kwadrat odległości punktu od średniej $\mu(X)$ dla całego zbioru. Intuicyjnie, punkty, które są daleko od średniej $\mu(X)$ mogą mieć istotny wpływ na wartość funkcji ϕ. Musimy więc zapewnić, odpowiednią częstotliwość wyboru takich punktów. Jak pokazuje pseudokod, implementacja takiej konstrukcji jest całkiem prosta. Zauważmy, że jest ona też bardzo praktyczna. Algorytm przechodzi przez zbiór danych jedynie dwukrotnie, a jego złożoność to $O(nd)$. Nie mamy zależności od k co jest kluczowe w konkeście praktyczności takiego rozwiązania.

3.3. Analiza

W tej części chciałbym udowodnić, że zaproponowany w poprzdniej części algorytm oblicza lightweight coreset dla odpowiedniego m.

Twierdzenie 3.2. *Niech $\epsilon > 0$, $\delta > 0$ oraz $k \in \mathbb{N}$. Niech X to zbiór punktów z $\mathbb{R}^d$ oraz C zbiorem, który dostajemy z algorytmu dla*

$$m \geq c\frac{dk\log k + \log\frac{1}{\delta}}{\epsilon^2}$$

gdzie c to stała. Wtedy z prawdopodobieństwem co najmniej $1 - \delta$ zbiór C jest (ϵ, k) lightweight coresetem dla X.

Dowód. Na początku wyprowadzę rozkład dla punktów $x \in X$ uwzględniający ich wagę. Następnie pokażę, że wybierając wystarczającą liczbę punktów z tego rozkładu uzyskuje się (ϵ, k) lightweight coreset.

Zacznę od ograniczenia wagi dla każdego punktu $x \in X$. W tym celu definiuje funkcję:

$$f(Q) = \frac{1}{2|X|}\phi_X(Q) + \frac{1}{2|X|}\phi_X(\mu(X))$$

gdzie $\mu(X)$ to średnia zbioru X oraz dowodzę następujący lemat.

Lemat 1. *Niech X to zbiór punktów z $\mathbb{R}^d$ wraz ze średnią $\mu(X)$. Dla każdego $x \in X$ oraz $Q \subset \mathbb{R}$ zachodzi:*

$$\frac{d(x,Q)^2}{f(Q)} \leq \frac{16d(x,\mu(X))^2}{\frac{1}{|X|}\sum_{x' \in X} d(x',\mu(X))^2} + 16$$

Dowód. Z nierówności trójkąta oraz z faktu, że $(|a| + |b|)^2 = 2a^2 + 2b^2$, otrzymujemy

$$d(\mu(X),Q)^2 \leq 2d(x,\mu(x))^2 + 2d(x,Q)$$

Biorąc średnie wszystkich $x \in X$ otrzymujemy:

$$d(\mu(X),Q)^2 \leq \frac{2}{|X|}\sum_{x \in X} d(x,\mu(x))^2 + \frac{2}{|X|}\sum_{x \in X} d(x,Q)$$

$$= \frac{2}{|X|}\phi_X(\mu(X)) + \frac{2}{|X|}\phi_X(Q)$$

To implikuje, że dla każdego $x \in X$ oraz $Q \subset \mathbb{R}$ zachodzi:

$$d(x,Q)^2 \leq 2d(x,\mu(x))^2 + 2d(\mu(X),Q)$$

$$\leq 2d(x,\mu(x))^2 + \frac{4}{|X|}\phi_X(\mu(X)) + \frac{4}{|X|}\phi_X(Q)$$

Dzieląc powyższą nierówność przez wyżej zdefiniowaną funkcję $f(Q)$ dostajemy:

$$\frac{d(x,Q)^2}{f(Q)} \leq \frac{2d(x,\mu(x))^2 + \frac{4}{|X|}\phi_X(\mu(X)) + \frac{4}{|X|}\phi_X(Q)}{\frac{1}{2|X|}\phi_X(Q) + \frac{1}{2|X|}\phi_X(\mu(X))}$$

$$\leq \frac{2d(x,\mu(x))^2 + \frac{4}{|X|}\phi_X(\mu(X))}{\frac{1}{2|X|}\phi_X(\mu(X))} + \frac{\frac{4}{|X|}\phi_X(Q)}{\frac{1}{2|X|}\phi_X(Q)}$$

$$\leq \frac{16d(x,\mu(X))^2}{\frac{1}{|X|}\sum_{x' \in X} d(x',\mu(X))^2} + 16$$

co kończy dowód lematu. □

Powyższy lemat implikuje, że stosunek pomiędzy kosztem kontrybucji jednego punku $x \in X$ a $f(Q)$ jest ograniczona dla każdego $Q \subset X$ przez:

$$s(x) = \frac{16d(x,\mu(X))^2}{\frac{1}{|X|}\sum_{x' \in X} d(x',\mu(X))^2} + 16$$

Zdefinuje $S = \sum_{x' \in X} s(x')$ zauważając, że $S = 32$ dla każdego zbioru X. Dzięki temu mogę zapisać rozkład $q(x)$ jako:

$$q(x) = \frac{1}{2}\frac{1}{|X|} + \frac{1}{2}\frac{d(x,\mu)^2}{\sum_{x' \in X} d(x',\mu)^2} = \frac{s(x)}{S|X|}$$

dla każdego $x \in X$. Rozpatruję funkcję:

$$g_Q(x) = \frac{d(x,Q)^2}{f(Q)s(x)}$$

dla każdego $x \in X$ oraz $Q \subset \mathbb{R}$. Zauważmy, że dla dowolnego zbioru $Q \subset \mathbb{R}^d$ zachodzi:

$$\phi_X(Q) = \sum_{x \in X} d(x,Q)^2 = S|X|f(Q) \sum_{x \in X} \frac{s(x)}{S|X|}\frac{d(x,Q)^2}{f(Q)s(x)}$$

$$= S|X|f(Q) \sum_{x \in X} q(x)g_Q(x)$$

Wprowadźmy notację:

$$\mathbb{E}_q[g_Q(x)] = \sum_{x \in X} q(x)g_Q(x)$$

dzięki której przekształcamy ostatnie równanie:

$$\phi_X(Q) = S|X|f(Q)\mathbb{E}_q[g_Q(x)]$$

Następnym krokiem jest ograniczenie wartości $\mathbb{E}_q[g_Q(x)]$. Autorzy [3] nie dowodzą wprost tego ograniczenia, powołując się na inne prace. Dowód jest bardzo skompilowany i wykracza tematyką istotnie poza ramy tej pracy więc go pomijam. Korzystam z finalnego ograniczenia:

$$|\mathbb{E}_q[g_Q(x)] - \frac{1}{|C|}\sum_{x \in X} g_X(x)| \leq \frac{\epsilon}{32}$$

Powyższe ograniczenie jest prawdziwe z prawdopodobieństwem $1-\delta$ dla dowolnego $Q \subset \mathbb{R}^d$ o rozmiarze nie większym niż k. Mnożąc obie strony nierówności przez $32|X|f(Q)$ otrzymujemy:

$$|32|X|f(Q)\mathbb{E}_q[g_Q(x)] - \frac{32|X|f(Q)}{|C|}\sum_{x \in X} g_X(x)| \leq \epsilon|X|f(Q)$$

Niech (C,u) będzie ważonym zbiorem, gdzie dla każdego $x \in C$ definujemy funkcję $u(X) = \frac{1}{|C|q(x)}$. Wynika z tego, że:

$$\frac{32|X|f(Q)}{|C|}\sum_{x \in X} g_X(x) = \sum \frac{1}{|C|q(x)} d(x,Q)^2$$

$$= \sum u(x)d(x,Q)^2 = \phi_C(Q)$$

A więc otrzymujemy:

$$|32|X|f(Q)\mathbb{E}_q[g_Q(x)] - \phi_C(Q)| \leq \epsilon|X|f(Q)$$

$$|\phi_Q(Q) - \phi_C(Q)| \leq \frac{\epsilon}{2}\phi_X(Q) + \frac{\epsilon}{2}\phi_X(\mu(X))$$

co kończy dowód twierdzenia 3.2.

□

4. Geometryczna Dekompozycja

W tej części pracy przestawię konstrukcję budowy coresetu bazującą na geometrycznej dekompozycji problemu. Punktem wyjścia były badania [8], na których bazuje opisany poniżej algorytm. Zgodnie z nimi budowa coresetu w kontekście problemu K-means to wieloetapowy proces, który jest sekwencją algorytmów z prac: [5] [6] [2] [8]. W rozdziale przyjmuję następujący porządek analizy konstrukcji:

- Na początku opiszę konstrukcję pomocnicze w kolejności: [5], [6], [2].
- W sekcji *Podsumowanie* opiszę właściwy algorytm z pracy [8]

4.1. Algorytm Gonzalez'a

Pierwszy algorytm, który opiszę to *Farthest point algorithm* z pracy [5]. Jest to pierwszy algorytm aproksymacyjny rozwiązujących problem k-centrów z błędem nie większem niż $2OPT$, gdzie OPT to optymalne rozwiązanie. Jego złożoność to $O(nk)$, gdzie n to liczba punktów.

Na potrzeby tego rozdziału wprowadzę kilka definicji

Definicja 4.1. Niech *G = (V, E, W)* będzie ważonym nieskierowanym grafem ze zbiorem wierzchołków V, krawędzi E oraz funkcją $W : E \to \mathbb{R}^+$ przyporządkowującą wagi krawędziom. W tym rozdziale utożsamiam wagi z dystansem pomiędzy dwoma punktami.

Definicja 4.2. Podział zbioru wierzchołków na k zbiorów $B_1, \ldots, B_k$, nazywamy *k-split*.

Definicja 4.3. Zbiory B_i podziału k-split nazywamy *klastrami*.

Dla każdego k-splitu defninujemy funkcję celu $f : B_1, \ldots, B_k \to \mathbb{R}^+$. W tym rozdziale zakładamy, że funkcją celu jest $max(M_1, \ldots, M_k)$, gdzie M_i to największa waga krawędzi pomiędzy dowolnymi dwoma punktami z B_i.

Definicja 4.4. *Problem klastrowania.* Dla danego grafu G, funkcji celu f oraz $k \in \mathbb{N}^+$ znaleźć k-split dla którego funkcja f jest zminimalizowana. Dla przykładu: znaleźć k-split $(B_1^*, \ldots, B_k^*)$ taki, że

$$f(B_1^*, \ldots, B_k^*) = min\{f(B_1, \ldots, B_k) | (B_1, \ldots, B_k) \text{ to k-split dla G } \}$$

Niech S to bedzie zbiór który chcemy sklastrować oraz niech T będzie podzbiorem S. Zakładamy, że $|S| > k$ ponieważ w przeciwnym przypadku problem jest trywialnie rozwiązywalny.

Definicja 4.5. Zbiór T nazywamy $(k+1)$ kliką wysokości h jeżeli moc zbioru T jest równa $k+1$ oraz odległość pomiędzy parą dwóch rożnych punktów jest równa co najmniej h.

Niech $OPT(S)$ oznacza optymalne rozwiązanie problemu k-centrów dla zbioru S. Udowodnię teraz następujący lemat, który jest potrzebny w dowodzie błędu aproksymacji algorytmu.

Lemat 2. *Jeżeli w zbiorze S istnieje $(k+1)$ klika wysokości h, to $OPT(S) \geq h$.*

Dowód. Wartość funkcji celu dla kliki wysokości h to conajmniej h ponieważ kilka ma $k+1$ elementów to 2 z nich wylądują w jednym klastrze. W takim razie waga krawędzi pomiędzy tymi punktami to conajmniej h co implikuje, że $OPT(S) > h$. □

Algorytm składa się z fazy inicjalizującej oraz $k-1$ faz powiększających. W fazie inicjalizującej wszystkie elementy są przypisana do zbioru B_1, który jest pierwszym klasterm. Jeden z elementów tego zbioru oznaczamy jako (t_1) - środek klastra B_1. Wybór tego elementu jest losowy. Podczas j fazy powiększającej, niektóre elementy z istniejącego podziału na klastry $B_1, \ldots, B_j$ trafiają do nowego zbioru B_{j+1}. Dodatkowo jeden z elemntów nowego zbioru będzie oznaczony jako (t_{j+1}) - środek klastra B_{j+1}. Budowę zbioru B_{j+1} rozpoczynamy od wyboru punktu v, który należy do jednego ze zbiorów $B_1, \ldots, B_j$ oraz odległość do centrum jego akutalnego klastra jest największa spośród wszystkich punktów. Taki punkt będzie oznaczony jako (t_{j+1}), czyli jest centerm klastra B_{j+1}. Każdy punkt dla którego dystans do v jest nie większy niż dystans do centra klastu w którym się znajduje zostaje przeniesiony do B_{j+1}.

Algorithm 2

For each point not in T, the algorithm keeps $neighbor(p)$, the nearest point in T, and $dist(p)$, the distance from p to $neighbor(p)$.
procedure FARTHEST POINT ALGORITHM
  $T \leftarrow \emptyset$
  $dist(p) \leftarrow$ inf for all $p \in S$
  while $|T| \leq k$ **do**
    $D \leftarrow max\{dist(p) | p \in S - T\}$
    choose v from $S - T$ such $dist(v) = D$
    add v to T
    update $neighbor(p)$ and $dist(p)$ for all $p \in S - T$
return T

Taki algorytmy buduje jakiś k-split. Teraz pokażę, że dla takiego k-splitu wartość funkcji celu jest ograniczona przez $2OPT(S)$.

Niech $v \in B_j$ oraz $1 \leq j \leq k$ będzie wierzchołkiem którego odległość do (t_j) jest maksymalna. Niech h będzie tą odległością. Ponieważ dla zbioru wag krawędzi zachodzi nierówność trójkąta to:

$$W(E(x,y)) \leq W(E(x,t_i)) + W(E(y,t_i)) \leq 2h$$

gdzie t_i to centrum klastra dla punktów $x, y \in B_i$. A więc możemy ograniczyć wartość naszej funkcji celu przez $\leq 2h$. Ponieważ v nigdy nie zostało wybrane na centrum klasta to wiemy, że $W(t_i, t_j) \geq h$ dla $i \neq j$. Niech $T = \{t_1, \ldots, t_k, v\}$. Z definicji wiemy, że T jest $(k+1)$ kliką wagi h więc z lematu 2 $OPT(S) \geq h$. A więc ograniczenie wartości funkcji celu to $\leq 2h \leq 2OPT(S)$.

4.2. Konstrukcja kraty wykładniczej

W tym podrozdziale omówię część pracy [6]. Tematem pracy są konstrukcje algorytmów dla problemów K-means oraz K-median. W kontekście problemu K-means autorzy zaproponowali algorytm bazujący na budowaniu coresetu, który uzyskuje lepszą złożoność od [7]. Schemat konstrukcji jest następujący:

- Obliczmy szybką ale niedokładną aproksymację dla problemu K-means z dużym k.
- Obliczoną aproksymację przekształcamy w (ϵ, k) coreset używając kraty wykładniczej.

4.2.1. Szybka aproksymacja dla problemu K-means.

Zacznijmy od pierwszej części. Dokładniej udowodnię następujące twierdzenie dla konstrukcji którą opiszę.

Twierdzenie 4.1. *Dla danego zbioru n punktów $X \subset \mathbb{R}^d$ oraz parametru $k \in \mathbb{N}^+$ możemy obliczyć zbiór P o mocy $O(k \log^3 n)$ dla którego*

$$\phi_X(P) \leq 32\phi^k_{opt}(X)$$

Czas działania algorytmu to $O(n)$ dla $k = O(n^{\frac{1}{4}})$ oraz $O(n \log(k \log n))$ w przeciwnym przypadku.

Niech $P \in \mathbb{R}^d$ to dany na wejściu zbiór k punktów. Będziemy chcieli szybko obliczyć aproksymację dla problemu K-means na tym zbiorze, gdzie k będzie rzędu $O(k \log^3 n)$.

Definicja 4.6. *Złe/dobre punkty.* Dla zbioru punktów X, punkt $p \in P$ nazywamy *złym* jeżeli

$$d(p, X) \geq 2d(p, C_{opt})$$

gdzie C_{opt} to zbiór punktór realizujący $\phi^k_{opt}(P)$. Punkt jest *dobry* jeżeli nie jest zły.

Na początku opiszę procedurę która dla danego P, wyznacza zbiór centrów X oraz zbiór $P^{'} \subset P$. Zbiór $P^{'}$ będzie zwierać *dobre* punkty dla zbioru X.

Konstrukcję zbioru X zaczynamy od obliczenia 2-aproksymacji problemu k-centrów dla zbioru P. Niech obliczony zbiór centrów to V. Taką aproksymację dla $k = O(n^{\frac{1}{4}})$ możemy obliczyć w czasie $O(n)$ oraz dla $k = \Omega(n^{\frac{1}{4}})$ w czasie $O(n \log k)$ [4]. Niech L będzie promieniem takiej aproksymacji, czyli największą

odległością pomiędzy centrem a punktem $p \in P - V$. Ponieważ [4] bazuje na [5] to dystans pomiędzy dowolną parą punktów z V to conajmniej L. To implikuje następujące ograniczenia:

$$\left(\frac{L}{2}\right)^2 \leq \phi^k_{opt}(V) \leq \phi^k_{opt}(P) \leq nL^2$$

Następnym krokiem będzie wylosowanie $\rho = \gamma k \log^2 n$ punktów ze zbioru P. Niech wylosowane punkty to Y oraz γ to stała, która wynika z analizy, którą zaraz przeprowadzimy. Finalnie, $X = Y \cup V$ będzie zbiorem centrów klastów. Dla $\rho > n$ przyjmujemy $X = P$.

Konstrukcja zbioru X jest stosunkowo prosta. Dużo cięższym zadaniem jest zbudowanie zbioru $P^{'}$, który jest zbiorem *dobrych* punktów dla X.

4.2.2. Konstrukcja zbioru dobrych punktów dla X.

Rozpatrzmy zbiór C_{opt}, który jest optymalnym zbiorem centrów dla problemu K-means w kontekście P. Dla każdego $c_i \in C_{opt}$ tworzymy kulę b_i o środku w c_i. Każda taka kula będzie zawierać $\eta = \frac{n}{20kk \log n}$ punktów z P. Jeżeli γ jest odpowiednio duże to z wysokim prawdopodobieństwem w każdym b_i jest przynajmniej jeden punkt z X. Dokładniej:

$$X \cap b_i \neq \emptyset \text{ dla } i = 1 \dots k$$

Niech P_{bad} będzie zbiorem złych punktów P w kontekście zbioru X. Załóżmy, że dla każdego b_i istnieje punkt $x_j \in X$, który $x_j \in b_i$. Zauważmy, że dla każdego punktu $p \in P \setminus b_i$ dla którego x_j jest centerm mamy $||px_j|| \leq ||pc_i||$. W szczególności takie punkty są *dobre*, dla c_i będącymi optymalnymi centrami dla tych punktów. Zatem z wysokim prawdopodobieństwem jedyne *złe* punkty będą w kulach b_i dla $i = 1, \dots, k$. To implikuje, że z wysokim prawdopodobieństwem liczba złych punktów w P dla zbioru X to co najwyżej $\beta = k\eta = \frac{n}{20 \log n}$.

W takim razie złych punktów nie jest dużo. Mimo tego bezpośrednie wyznaczenie tych punktów jest skomplikowane. Autorzy pracy [6] budują zbiór $P^{'}$ tak aby koszt złych punktów w $P^{'}$ był jak najmniejszy. Koszt w tym kontekście oznacza to jaki wkład ma punkt w wartość $\phi_{P'}(X)$. Dla każdego punktu w P obliczamy najbliższego sąsiada w X. Niech $r(p) = d(p, X)$ dla każdego punktu $p \in P$. Teraz podzielimy P na zbiory według następującej formuły:

$$P[a, b] = \{p \in P \mid a \leq r(p) \leq b\}$$

A dokładniej

$$P_0 = P\Big[0, \frac{L}{4n}\Big]$$

$$P_\infty = P\Big[2Ln, \infty\Big]$$

$$P_i = P\Big[\frac{2^{i-1}L}{n}, \frac{2^i L}{n}\Big]$$

dla $i = 1, \ldots, M$, gdzie $M = 2\lceil \lg n \rceil + 3$. Taki podział możemy wykonać w czasie linowym. Niech P_α będzię ostatnim zbiorem, który zawiera więcej niż $2\beta = \frac{n}{10 \log n}$. Szukany zbiór zdefiniujemy następująco:

$$P^{'} = V \cup \bigcup_{i \leq \alpha} P_i$$

gdzie $|P^{'}| \geq \frac{n}{2}$ oraz $\phi_{P'}(X) = O(\phi_{P'}(C_{opt}))$. Teraz udowodnimy, że faktycznie tak zdefiniowane $P^{'}$ spełnia powyższe założenia.

Dowód. Moc zbioru $P^{'}$ jest na pewno równa conajmniej $\Big(n - |P_\infty| - M\frac{n}{10 \log n}\Big)$. Zauważmy, że $P_\infty \subseteq P_{bad}$ oraz $|P_{bad}| \leq \beta$. A więc:

$$\begin{aligned} |P^{'}| &\geq n - \frac{n}{20 \log n} - M\frac{n}{10 \log n} \\ &= n - \Big(\frac{n}{10 \log n}\Big)\Big(M + \frac{1}{2}\Big) \\ &= n - \Big(\frac{n}{10 \log n}\Big)\Big(2\lceil \lg n \rceil + 3 + \frac{1}{2}\Big) \\ &\geq \frac{n}{2} \end{aligned}$$

Jeżeli $\alpha > 0$ to $|P_\alpha| \geq 2\beta = \frac{n}{10 \log n}$. Teraz chcemy oszacować jaką kontrybucję w $P^{'}$ mają złe punkty. Z uwagi na to jak budujemy $P^{'}$ w najgorszym przypadku wszystkie złe punkty będą w P_α. Wtedy takie punkty będą miały największy wpływ na funkcję ϕ. Niech $Q^{'} = P_\alpha \backslash P_{bad}$. Dla dowolnego punktu z $p \in P^{'} \cap P_{bad}$ oraz $q \in Q^{'}$, mamy $d(p, X) \leq 2d(q, X)$. Dodatkowo $|Q^{'}| > |P_{bad}|$, a więc:

$$\phi_{P' \cap P_{bad}}(X) \leq 4\phi_{Q'}(X) \leq 16\phi_{Q'}(C_{opt}) \leq 16\phi_{P'}(C_{opt})$$

Teraz możemy wyprowadzić następujące ograniczenie:

$$\begin{aligned} \phi_{P'}(X) &= \phi_{P' \cap P_{bad}}(X) + \phi_{P' \backslash P_{bad}}(X) \\ &\leq 16\phi_{P'}(C_{opt}) + 4\phi_{P'}(C_{opt}) = 20\phi_{P'}(C_{opt}) \end{aligned}$$

Jeżeli $\alpha = 0$ to dla dowolnego punktu $p \in P^{'}$ mamy:

$$(d(p, X))^2 \leq n\Big(\frac{L}{4n}\Big)^2 \leq \frac{L^2}{4n}$$

Zatem:

$$\phi_{P'}(X) \leq \frac{L^2}{4} \leq \phi_V(C_{opt}) \leq \phi_{P'}(C_{opt})$$

gdzie $V \subseteq P^{'}$. □

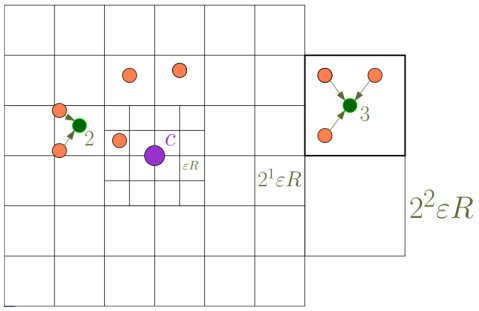
Powyższa analiza dowodzi poprawności konstrukcji dla zbiorów X i $P^{'}$. Podsumowując otrzymujemy zbiór X o mocy $O(k \log^2 n)$ oraz zbiór $P^{'}$ dla którego mamy $\phi_{P'}(X) \leq 32\phi_{P'}(C_{opt})$. Czas działania tego algorytmu to $O(n)$ dla $k = O(n^{\frac{1}{4}})$ oraz $O(n \log(k \log n))$ w przeciwnym przypadku. Aby otrzymać taką złożoność kluczowe jest szybkie obliczanie najbliższych sąsiadów. Autorzy proponują [1].

Wróćmy teraz do twierdzenia 4.1, które zostało zdefiniowane na początku podrozdziału 4.2. Powyżej zdefiniowaną procedurę powtarzamy dla zbioru $P_1 = P \setminus P^{'}$. Analogicznie otrzymamy zbiór $P_1^{'}$ oraz X_1. Kolejny raz aplikujemy procedurę na zbiorze $P_2 = P \setminus (P^{'} \cup P_1^{'})$. Ponieważ za każdym razem zbiór P_i zmieniejsza się o połowę to taki proces zakończy się po $O(\log n)$ powtórzeniach. Finalnie otrzymamy zbiór $X \cup X_1 \dots X_i$ o mocy $O(k \log^3 n)$ dla którego $\phi_P(X) \leq 32\phi_P(C_{opt})$. Złożoność pozostanie taka sama, czyli $O(n)$ dla $k = O(n^{\frac{1}{4}})$ oraz $O(n \log(k \log n))$ w przeciwnym przypadku.

4.2.3. Krata wykładnicza.

Przejdzmy teraz do kluczowej części tego podrozdziału, czyli konstrukcji kraty wykładniczej. Niech $P \subset \mathbb{R}^d$, $|P| = n$ oraz niech $A = \{x_1, \dots, x_m\}$ będzie zbiorem punktów dla którego zachodzi $\phi_P(A) \leq c\phi_{opt}^k(P)$, gdzie c to stała. Nasze A otrzymamy z konstrukcji opisanej w 4.2.1 dla $k = O(kpolylog)$.

Niech $R = \sqrt{\frac{\phi_P(A)}{cn}}$ będzie dolnym ograniczeniem dla $R_{opt}^{\phi}(P,k) = \sqrt{\frac{\phi_{opt}(P,k)}{n}}$ Niech P_i będzie podzbiorem punktów z P dla których punkt $x_i \in A$ jest dla niech najbliższym sąsiadem. Dla dowolnego $p \in P_i$, mamy $||px_i|| \leq \sqrt{xn}R$, ponieważ $||px_i||^2 \leq \phi_P(A)$ dla $i = 1, \dots, m$.



Rysunek 4.1: Intuicyjnie kratę wykładniczą możemy przedstawić obrazowo, gdzie $c = x_i$.

Kolejnym krokiem będzie budowa kraty wykładniczej wokół każdego punktu x_i oraz nałożenie jej na zbiór P. Niech $Q_{i,j}$ będzie kwatratem o boku długości $R2^j$ o środku w punkcie x_i dla $j = 0, \dots, M$, gdzie $M = \lceil 2 \lg(cn) \rceil$, który jest równoległy do osi układu współrzednych dla danej przestrzeni. Następnie, niech $V_{i,0} = Q_{i,0}$ oraz niech $V_{i,j} = Q_{i,j} \setminus Q_{i,j}$ dla $j = 0, \dots, M$. Kolejnym krokiem będzie przekształcenie $V_{i,j}$ w kratę, której komórki będą długości $r_j = \frac{\epsilon R 2^j}{10cd}$ oraz niech G_i oznacza wynikową kratę wykładniczą dla $V_{i,0}, \dots, V_{i,M}$. Mając G_i ob-

liczamy dla każdego punktu z P_i, komórkę do której należy. Dla każdej niepustej komórki z kraty wybieramy losowy punkt z P_i, który będzie jej reprezentantem. Do takiego punktu przypisujemy wagę, która bedzie równa liczbie punktów z komórki dla której jest reprezentantem. Po przejściu całej kraty otrzymamy zbiór S_i takich punktów. Definujemy $S = \bigcup_i S_i$, który jest (ϵ, k) coresetem. Zauważmy, że $|S| = O\Big(\frac{|A| \log n}{(c\epsilon)^d}\Big)$, ponieważ każda krata ma $\log n$ poziomów, a każdy poziom stałą liczbę komórek.

Dowód. TBA □

4.3. Heurystyka single swap

W tym podrozdziale opiszemy heurystykę single swap [2], która jest przykładem techinki *local search*. Algorytm przedstawi konstrukcję $(25 + \epsilon)$ aproksymacji dla problemu K-means.

4.4. Podsumowanie

TBA

5. Analiza Implementacji

TBA

6. Bibliografia

[1] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM 45*, 6 (Nov. 1998), 891–923.

[2] Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V. Local search heuristics for k-median and facility location problems. *SIAM J. Comput. 33* (2004), 544–562.

[3] Bachem, O., Lucic, M., and Krause, A. Scalable k-means clustering via lightweight coresets.

[4] Feder, T., and Greene, D. Optimal algorithms for approximate clustering. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1988), STOC '88, Association for Computing Machinery, p. 434–444.

[5] Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci. 38* (1985), 293–306.

[6] Har-Peled, S., and Mazumdar, S. On coresets for k-means and k-median clustering. 291–300.

[7] Matousek, J. On approximate geometric k-clustering.

[8] Munteanu, A., and Schwiegelshohn, C. Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms. *Künstliche Intell. 32*, 1 (2018), 37–53.

6. Spis treści