Project 2019-06 | Multivariate Analysis

Pattern Recognition 79 (2018)

402-413

I-k-means−+: An iterative clustering algorithm based on an enhanced version of the k-means

Hassan Ismkhan

(Computer engineering, University of Bonab, Bonab, East Azerbaijian, Iran)

학 번	이 름
182STG26	이은진

1. Introduction

데이터 Clustering은 분류 문제 뿐 아니라, 모델링 기법으로 사용될 수 있다. 또한 머신러닝, 딥러닝 기술의 발전으로 영상이나 이미지 프로세싱, 텍스트 분석, 소셜 네트워크 분석, 문서 분류 등과 같이 다양한 범위에 적용되어 사용되고 있다. Clustering 기법의 종류로는 DBSCAN, CURE Chameleon, k-means 등 다양하게 있다. 이 논문에서는 k-means clustering을 보완한 논문이다.

k-means는 각 데이터가 속한 그룹의 평균과의 유클리디안 거리를 최소화하는 알고리즘이다.

1. k개의 center를 랜덤하게 선정한다. $C = \{c_1, c_2, \dots, c_k\}$.
2. 각각의 점이 가장 가까운 center로 membership을 정한다. (그룹을 나눠준다)
3. 새롭게 만들어진 그룹의 평균을 center로 변경한다.
4. 알고리즘 2-3을 C가 변하지 않을 때까지 반복한다.

위 논문에서는 k-means가 SSEDM을 최소화하는 알고리즘이라고 정의 내린다.

- For an input I, and solution S, where $S = S_1 \cup S_2 \cup \dots \cup S_k$, and $S_i \cap S_j = 0 (i \neq j)$, the SSEDM value of S is calculated as:

$$SSEDM(S) = SSEDM(I, S, K) = \sum_{i=1}^{k} SSEDM(S_i)$$

- In this paper, $SSEDM(S_i)$ is called partial SSEDM of cluster S_i, and can be computed using following equation:

$$SSEDM(S_i) = \sum_{\forall P \in S_i} dis\left(P, mean(S_i)\right)^2$$

해당 solution S에서 cluster가 $S = S_1 \cup S_2 \cup \dots \cup S_k$, and $S_i \cap S_j = 0 (i \neq j)$로 존재한다면 $SSEDM(S_i)$의 전체 합인 $SSEDM(S)$을 최소화하는 알고리즘이라고 설명한다. k-means 알고리즘은 간단하고 직관적이라는 장점이 있지만, 초기값의 영향을 많이 받아 초기값을 잘못 찾으면 성능이 매우 나빠진다. 따라서 이 논문은 k-means의 단점을 보완하는 방법을 제안하고 있다.

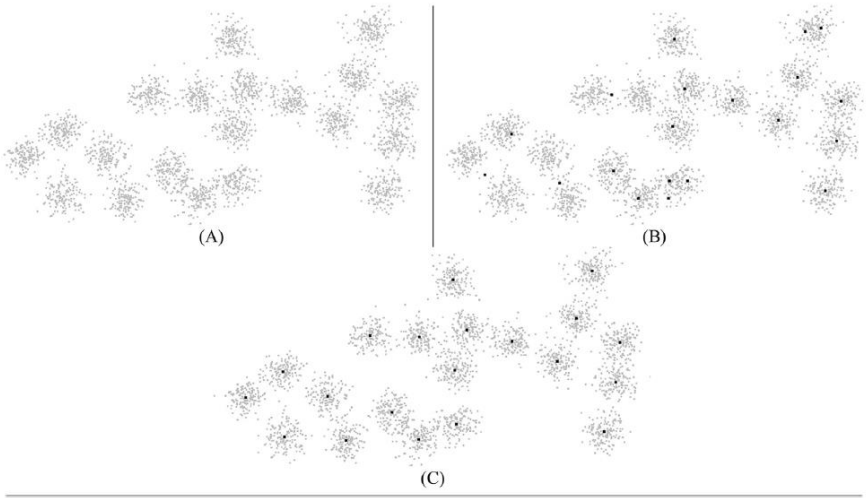


Fig. 1. (A) Original dataset with 20 clusters. (B) A solution obtained by k-means++, with SSEDM = 23829449829. (C) A solution with SSEDM = 12146257522. Perhaps it is the possible optimal solution.

위 그림 (A)는 cluster 20개인 원래 dataset 그림이고, (B)는 초기값을 잘못 찾았을 때, (C)는 잘 찾았을 때의 그림이다. k-means(++)의 성능 차이가 초기값에 따라 확연히 차이남을 알 수 있다.

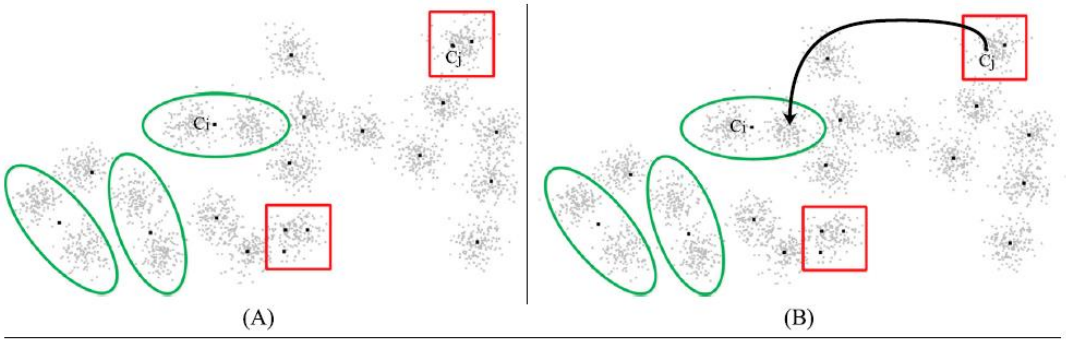


Fig. 2. (A) Each oval includes a cluster which can be split, and each square includes at least one cluster which can be removed. (B) Changing the location of C_j into cluster C_i, and applying re-clustering can improve the solution.

저자는 이렇게 초기값의 영향을 많이 받는 k-means의 알고리즘을 보완하기 위해 minus-plus 과정을 도입한다. minus-plus 과정을 간략히 말하자면, 위의 그림에서 cluster j는 원래는 하나의 cluster임에도 불구하고 2개의 cluster로 잘못 분류되어있다. 반면에 cluster i는 2개의 cluster인데, 하나의 cluster로 분류되어 나눠질 필요가 있어 보인다. 이때, cluster j와 cluster i를 각각 매칭시켜, cluster j에 있는 한 center를 cluster j로 옮긴다면, k-means의 성능을 높일 수 있을 것이다.

2. Recent works for k-means

2.1 Initialization methods for the k-means

k-means의 초기값을 보완하기 위해 연구한 논문들이다.

- Clustering to minimize the maximum intercluster distance
- A new initialization technique for generalized Lloyd iteration
- A method for initializing the k-means clustering algorithm using kd-trees
- K-means++: the advantages of careful seeding
- A deterministic method for initializing k-means clustering(PCA-part)
- The global k-means clustering algorithm(GKM)
- The MinMax k-means clustering algorithm

2.2 Methods for speeding up the k-means

k-means의 속도를 보완하기 위해 연구한 논문들이다.

- Randomized dimensionality reduction for k-means clustering
- Using the triangle inequality to accelerate k-means
- Making k-means even faster
- GAD : general activity detection for fast clustering on large data
- Improvement of the k-means clustering filtering algorithm

3. The I-k-means -+

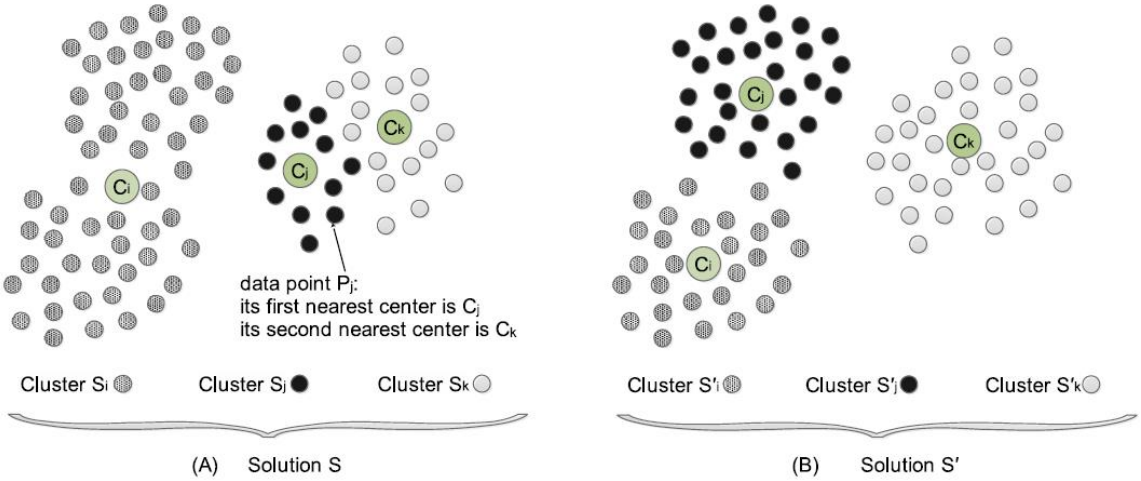


Fig. 4. A cluster S_i is divided into two clusters and a cluster S_j is removed, and a new solution S' is produced. It is obvious that when a cluster is removed (S_j), its data points are redistributed among other suitable clusters (S'_k). These are performed, simply, by changing C_j into a random data point in S_i and then applying re-clustering process.

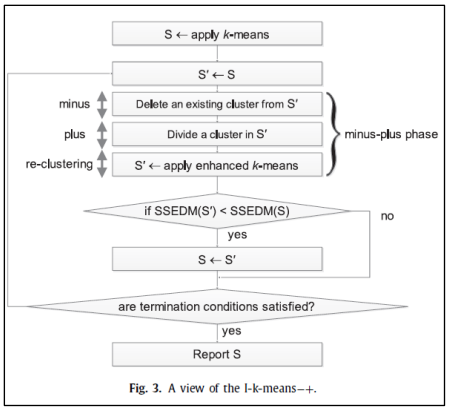


Fig. 3. A view of the I-k-means−+.

앞에서 말했듯이 cluster i 와 cluster j 를 한 pair 로 매칭시켜, minus-plus 과정을 진행한다. 하지만 이때 두가지 challenges 가 있다. 첫째로 cluster i-cluster j 처럼 적합한 pair 를 매칭하는 과정이 필요하다. 모든 pairs 를 고려하면 좋지만, 너무 시간이 오래 걸려 비효율적이다. 둘째로, re-cluster 할 수 밖에 없을 때 re-cluster 과정을 어떻게 효율적으로 할 지이다.

3.1 Detecting suitable pairs

Running time 절약을 위해, minus-plus 단계 전에 우선 cluster 간 적합한 pair를 찾는 과정이 필요하다(first challenge). 그래서 cluster S_j를 제거했을 때 증가하는 SSEDM이나, cluster S_i를 두 개로 나눴을 때 줄어드는 SSEDM을 비교해 적합한 pair를 찾는다. 이때 저자는 Cost와 Gain의 값을 정의 내린다.

- **Definition 1.** For a solution S, the cost of removing a cluster S_j, denoted by $\mathrm{Cost}(S_j)$, is what is added to the SSEDM(S), when S_j is removed from S.

$$Cost(S_j) = SSEDM(S) - SSEDM(S^*)$$
$$= SSEDM(S_j) - \sum_{\forall P \in S_j} dis(P, CC_p)^2$$

where CC_p is the second nearest center of P.

- **Definition 2.** For a solution S, the gain of dividing a cluster S_i, denoted by $\mathrm{Gain}(S_i)$, is what value is decreased from the SSEDM(S) by dividing S_i from S to two clusters.

$$Gain(S_i) = SSEDM(S) - SSEDM(S^*)$$
$$= SSEDM(S_i) - (SSEDM(S'_i) + SSEDM(S'_j))$$

cluster S_i 를 두 개로 나눴을 때 발생하는 $Gain(S_i)$ 을 구할 때, $\mathrm{SSEDM}(S_i)$ 는 바로 구할 수 있지만, $\mathrm{SSEDM}(S'_i)$, $\mathrm{SSEDM}(S'_j)$는 구할 수 없어, 추정치를 사용해야 한다.

Gain 추정치를 구하기 위해, X라는 개념을 새롭게 정의한다. cluster S_i의 center를 C_i이고, 해당 cluster에 속한 개수를 n이라고 한다면

$$X = \frac{1}{n}\sum_{\forall P \in S_i} dis(P, C_i)$$

X는 cluster에 속한 점들이 center와의 평균 거리라고 얘기할 수 있다(average distance). cluster S_i를 쪼개서 새롭게 생긴 cluster를 각각 S_i', S_j'이라고 한다면 해당 cluster의 개수는 n/2, center와의 평균 거리는 X/2라고 가정한다. 그러면 다음과 같은 식을 구할 수 있다.

$$SSEDM(S_i') \approx SSEDM(S_j') \approx \frac{n}{2} \times \left(\frac{X}{2}\right)^2$$

또한 X를 이용해, $SSEDM(S_i) \approx n \times X^2$도 구할 수 있다. 위의 식들을 조합하면, 다음과 같은 추정치를 얻을 수 있다.

$$Gain(S_i) \approx n \times X^2 - 2 \times \frac{n}{2} \times \left(\frac{X}{2}\right)^2$$
$$= \frac{3}{4} n \times X^2 \approx \frac{3}{4} SSEDM(S_i)$$

이때, coefficient $\alpha = \frac{3}{4}$로 놓아

$$Gain(S_i) \approx \alpha SSEDM(S_i)$$

라는 간단한 추정치를 얻을 수 있다. Results 파트에서 coefficient를 조절해 해당 알고리즘 성능을 비교하고자 한다.

다음으로 저자는 Heuristic으로 알고리즘을 보다 효율적으로 작동시킬 수 있을 것이라 말한다. 또한 adjacent와 strong adjacent 개념을 정의한다.

- **Heuristic 1.** A pair of clusters, S_i and S_j can be considered for the minus-plus phase, S_i for dividing, and S_j for removing, if $Gain(S_i) > Cost(S_j)$.
- **Definition 3.** A cluster S_j with center C_j is an adjacent of a cluster S_i with center C_i
  ⇔ there is a data point P in S_i (its first nearest center is C_i) s.t the second nearest center of P is C_j.
  (When a cluster S_j is adjacent to a cluster S_i, it dose not imply S_i is adjacent to S_j.)
- **Definition 4.** A cluster S_j is a strong adjacent to a cluster S_i
  ⇔ S_j is an adjacent of S_i, and S_i is an adjacent of S_j.
- **Heuristic 2.** If a cluster is produced by dividing another cluster in the current minus-plus phase, it and its strong adjacent clusters cannot be removed in the next iterations.
- **Heuristic 3.** If a cluster is removed in the current minus-plus phase, its strong adjacent clusters cannot be divided in the next iterations.

3.2 The topical k-means

re-cluster을 할 수 밖에 없을 때 저자는 t-k-means 알고리즘을 사용해 re-clustering을 제안한다(second challenge). re-clustering은 시간이 많이 소요될 수 있으므로, 한번만 적용한다. 이때 t-k-means 알고리즘으로 affected points와 unaffected points를 찾아 re-clustering의 시간을 향상시킬 수 있다.

- **Definition 5.** In a clustering solution S, a data point P is affected point of a cluster center C_i
  ⇔ the first or the second nearest center of P is C_i.

Step#1.

1. Let *AC* (active centers) be an initially empty set of centers.
2. $AC \leftarrow AC \cup \{C_i\} \cup \{C_j\}$.
3. *AC-Adjacent* ←adjacent center of C_j before changing.
4. *AP* ← affected points of C_j before changing.

Step#2.

1. If *AC* is an empty set go to the end.
2. *AC-Adjacent* ← adjacent centers of centers in *AC*.
3. *Potential-AC* ← {}.
4. AP ← affected points of centers in *AC*.

Step#3.
For each affected point, *P*, of centers in *AC*

1. Update the first and the second nearest centers of *P*, considering $AC \cup AC\text{-}Adjacent$.
2. If the first center of *P* is changed from C_x to C_y, then $Potential\text{-}AC \leftarrow Potential\text{-}AC \cup \{C_x\} \cup \{C_y\}$

Step#4. Update centers.
Step#5. $AC \leftarrow Potential\text{-}AC$
Step#6. Go to Step#2.
Step#7. The end of the algorithm.

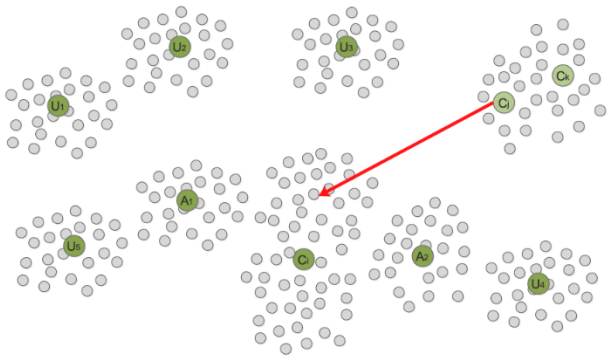


Fig. 5. When only one center C_j of a clustering solution is updated, then in order to accelerate, re-clustering can be applied topically.

AC(active centers)와 Potential-AC, AP를 구해 unaffected points는 무시하고, affected points를 찾아내 re-clustering을 효율적으로 할 수 있는 알고리즘이다.

3.3 The I-k-means -+ via detailed instructions

3.1과 3.2로 두가지 challenges를 해결한다. 다음은 두가지 challenges를 보완하는 알고리즘을 사용하여 I-k-means-+의 전체적인 알고리즘 설명이다.

① k-means 알고리즘에서는 초기값을 랜덤하게 정하지만, 저자는 useful nearest centers을 사용해 초기값을 정한다. 첫번째 center는 첫번째 축에서 가장 작은 값을 가진 점으로 정하고, 다음 center는 UNC_P가 P의 useful nearest centers일 때, $\frac{\text{average}(\text{dis}(P,c))_{\forall c \in UNC_P}}{\max(dis(P,c))_{\forall c \in UNC_P}} \times \sum_{\forall c \in UNC_P} \ln(dis(P,c))$가 가장 큰 값으로 정한다. (해당 논문은 An initialization method for the k-means using the concept of useful nearest centers (2017)에서 자세히 볼 수 있다)

② $\#success \leftarrow 0$

③ indivisible cluster 중에서 Gain 값이 가장 큰 cluster S_i를 찾는다. 이때 S_i가 없다면 알고리즘을 중단한다.

④ k/2개의 cluster가 S_i보다 Gain값이 크다면 알고리즘을 중단한다.

⑤ 다음 조건을 만족하는 cluster 중에서 Cost 값이 가장 작은 cluster S_j를 찾는다.

1) $S_j \neq S_i$
2) $\text{Cost}(S_j) < \text{Gain}(S_i)$
3) Pair S_i and S_j should not be marked as an *unmatchable pair*.
4) S_i is not an adjacent of S_j, and S_j is not an adjacent of S_i.
5) The cluster S_j should not be marked as an *irremovable* cluster.

⑥ k/2개의 cluster가 S_j보다 cost가 작고, 위 조건 5가지를 만족한다면 알고리즘을 중단한다.

k/2개의 cluster가 S_j보다 cost가 작고, S_i가 다른 cluster와 pair될 수 없다면 S_i를 indivisible cluster로 저장 후 step3으로 돌아간다(나눌 다른 cluster를 찾기 위해).

⑦ 현재 solution S를 S′로 지정 후, S_j의 center를 S_i에 속한 점으로 랜덤하게 이동한다. 그 다음 해당 solution S′에 t-k-means 알고리즘을 적용한다.

⑧ SSEDM(I, k, S′) > SSEDM(I, k, S)라면, update된 solution S′는 보완되었다고 할 수 없으므로 해당 pair S_i와 S_j는 unmatchable pair로 저장한다.

SSEDM(I, k, S′) < SSEDM(I, k, S)라면,

1) S_i와 S_j는 irremovable cluster로 저장한다.

2) update되기 전 solution에서 S_j의 strong adjacent cluster를 indivisible cluster로 저장한다.

3) S ← S'

4) 현재 update된 solution에서 S_i와 S_j의 strong adjacent cluster를 irremovable cluster로 저장한다.

5) $\#success \leftarrow \#success + 1$

⑨ $\#success > k/2$일 때, 알고리즘을 중단한다.

4. Results of performed experiments

저자는 3가지 dataset에 IKM-+ 알고리즘을 적용하였다.

Table 1
Datasets used in experiments.

Type	Instance name		Dimension	#data points	#clusters
#1	A-series	A1	2	3000	20
		A2	2	5250	35
		A3	2	7500	50
	S-series	S1	2	5000	15
		S2	2	5000	15
		S3	2	5000	15
		S4	2	5000	15
		Birch1	2	100,000	100
#2	Iris		4	150	3
	Human-Activity-Recognition (HAR)		561	10,299	6
	ISOLET		617	7797	26
	Letter-Recognition (LR)		16	20,000	9
	Musk		168	6598	2
	Statlog (Shuttle)		9	58,000	7
#3	KDDCUP04Bio		74	145,751	2000

2차원의 synthetic 데이터와, UCI machine learning repository와 protein dataset(KDDCUP04Bio)에 대해 적용하였다. 각각의 data 수도 다르고 cluster개수도 다양하다. 특히 마지막 dataset은 data points도 가장 많고 cluster 개수도 제일 많다. 해당 데이터를 I-k-means-+에 적용 후 k-means 알고리즘과 k-means++ 알고리즘과 비교하였다. accuracy 비교 기준은 SSDEM과 해당 알고리즘의 runtime 그리고 부분 SSEDMs의 최댓값을 비교하였다.

4.1 Results on synthetic datasets

다음은 첫번째 데이터셋을 k-means, k-means++, IKM-+에 각각 적용했을 때 결과다.

Table 2
Results of 50 runs on synthetic datasets.

	Maximum of partial SSEDMs			SSEDM			Runtime (s)			t_{IKM-+}/t_{KM}
	KM	KM++	IKM−+	KM	KM++	IKM−+	KM	KM++	IKM−+	
A1	5.39E + 09	4.08E + 09	**7.51E + 08**	2.08E + 10	1.73E + 10	**1.22E + 10**	**1.70E − 02**	2.10E − 02	2.50E − 02	1.45
A2	5.78E + 09	4.31E + 09	**7.12E + 08**	3.47E + 10	2.99E + 10	**2.03E + 10**	5.50E − 02	9.30E − 02	**5.00E − 02**	0.91
A3	7.58E + 09	4.98E + 09	**7.13E + 08**	5.23E + 10	4.29E + 10	**2.90E + 10**	1.17E − 01	2.10E − 01	**1.12E − 01**	0.96
S1	6.59E + 12	6.20E + 12	**8.54E + 11**	1.85E + 13	1.67E + 13	**8.92E + 12**	**2.00E − 02**	2.30E − 02	2.20E − 02	1.11
S2	5.78E + 12	5.25E + 12	**1.33E + 12**	2.01E + 13	1.82E + 13	**1.33E + 13**	2.10E − 02	2.70E − 02	**2.00E − 02**	0.93
S3	3.41E + 12	3.04E + 12	**1.56E + 12**	1.94E + 13	1.90E + 13	**1.69E + 13**	**2.70E − 02**	2.60E − 02	5.30E − 02	1.96
S4	2.56E + 12	2.19E + 12	**1.79E + 12**	1.70E + 13	1.67E + 13	**1.57E + 13**	**4.00E − 02**	4.20E − 02	4.00E − 02	1.00
Birch1	3.15E + 12	2.96E + 12	**1.05E + 12**	1.13E + 14	1.06E + 14	**9.28E + 13**	**1.81E + 01**	2.47E + 01	2.73E + 01	1.51

SSEDM을 기준으로 비교했을 때, IKM-+의 accuracy가 훨씬 좋게 나온 것을 확인할 수 있다. 또한 IKM-+의 Runtime도 k-means에 비해 크게 증가하지 않거나, 몇몇 경우에서는 오히려 더 빠른 것을 확인할 수 있다.

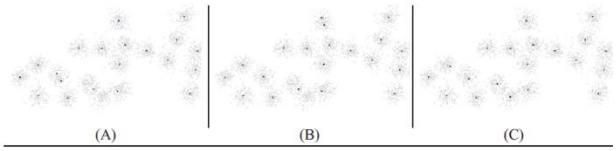


Fig. 6. For A1, the *best* solution of (A) KM, (B) KM++, and (C) the *worst* solution of IKM−+.

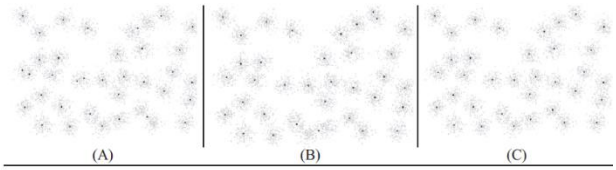


Fig. 7. For A2, the *best* solution of (A) KM, (B) KM++, and (C) the *worst* solution of IKM−+.

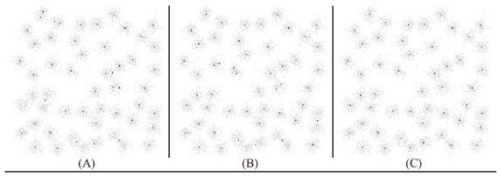


Fig. 8. For A3, the *best* solution of (A) KM, (B) KM++, and (C) the *worst* solution of IKM−+.

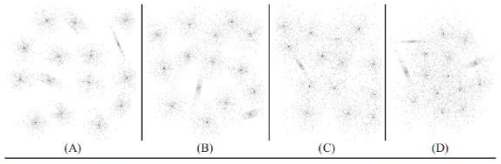


Fig. 9. For S1, S2, S3, and S4 the *best solution of KM, and KM++* and the *worst* solution of IKM−+ are similar to (A), (B), (C), and (D), respectively.

위 그림은 데이터셋 A1, A2, A3, S1, S2, S3에 적용하여 나타난 최종 결과 그림이다. k-means와 k-means++은 best solution을 IKM-+는 worst solution임에도 불구하고, IKM-+가 훨씬 성능이 좋은 것을 알 수 있다.

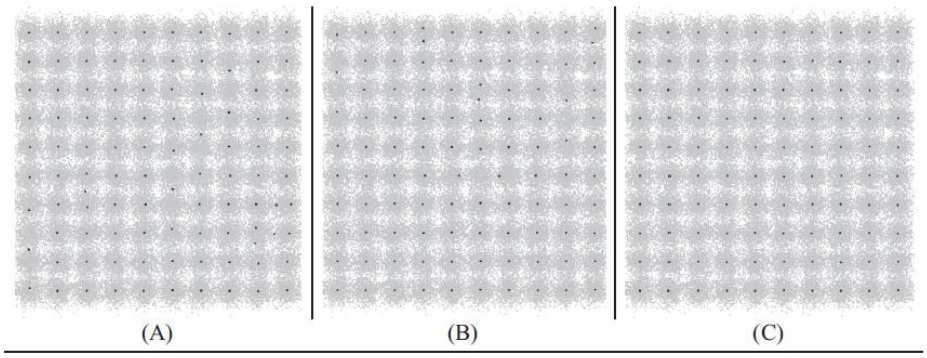


Fig. 10. For Birch1, the *best* solution of (A) KM, (B) KM++, and (C) the *worst* solution of IKM−+.

또한, 첫번째 데이터셋에서 가장 데이터수도 많고, cluster 수도 많은 Birch의 경우이다. cluster의 수가 증가할수록 k-means와 k-means++의 성능은 떨어지는 반면, IKM-+는 worst solution임에도 불구하고, 성능이 훨씬 좋았다.

4.2 Results on real-world datasets

다음은 UCI machine learning repository에 각각 k-means, k-means++, IKM-+을 적용한 결과다.

Table 3
Results on real-world datasets.

	Maximum of partial SSEDMs			SSEDM			Runtime (s)			t_{IKM-+}/t_{KM}
	KM	KM++	IKM−+	KM	KM++	IKM−+	KM	KM++	IKM−+	
Iris	6.53E+01	4.55E+01	**3.98E+01**	9.95E+01	8.41E+01	**7.89E+01**	0	0	0	1.00
HAR	5.02E+04	5.10E+04	**3.83E+04**	1.85E+05	1.85E+05	**1.82E+05**	1.696	1.635	4.641	2.74
ISOLET	4.23E+04	4.17E+04	**2.78E+04**	4.46E+05	4.47E+05	**4.41E+05**	5.066	6.712	6.207	1.23
LR	4.17E+04	4.29E+04	**3.93E+04**	6.20E+05	6.21E+05	**6.16E+05**	1.238	1.497	3.686	2.98
Musk	4.61E+09	4.47E+09	**4.35E+09**	6.09E+09	6.01E+09	**5.92E+09**	0.026	0.023	0.022	0.84
Statlog	**2.41E+08**	2.78E+08	3.32E+08	8.13E+08	7.99E+08	**4.35E+08**	0.426	0.434	0.342	0.80

이때에도 SSEDM을 기준으로 비교했을 때, IKM-+의 accuracy가 더 좋게 나온 것을 확인할 수 있다. 또한 IKM-+의 Runtime이 k-means에 비해 증가하는 경우도 있지만, 몇몇 경우에서는 오히려 더 빠른 것을 확인할 수 있다.

4.3 Effects of α and number of improvements on IKM-+

앞에서 Gain의 추정치를 구할 때 SSEDM에 α를 곱해 구했는데, 이때 α값을 달리하여 알고리즘 성능을 비교하였다. 또한 3.3 알고리즘 설명 부분에서 $\#success > k/2$ 일 때 알고리즘을 중단했는데, $\#success > k$ 로 기준을 바꿔 실행한 결과도 같이 비교하였다.

- v1: IKM−+ (original setting with $\alpha = 0.75$)
- v2: IKM−+ with $\alpha = 0.5$
- v3: IKM−+ with $\alpha = 1.0$
- v4: IKM−+ with "If #success > k, then go to end", instead "If #success > $k/2$, then go to end".
- v5: Random initialization for the first use of the k-means, in IKM−+.
- v6: KM++ initialization for the first use of the k-means, in IKM−+.

case 6까지 설정을 달리해 비교한 결과다. α을 바꾼다고 해서 SSEDM은 큰 차이가 나지 않았다. 또한 $\#success > k$일 때 runtime도 큰 차이가 나지 않음을 확인할 수 있었다.

Table 4
Effects of different settings of IKM−+ on SSEDM.

	SSEDM					
	v1	v2	v3	v4	v5	v6
Iris	7.89E+01	7.89E+01	7.89E+01	7.89E+01	9.69E+01	8.80E+01
HAR	1.82E+05	1.82E+05	1.82E+05	1.82E+05	1.82E+05	1.82E+05
ISOLET	4.41E+05	4.41E+05	4.40E+05	4.41E+05	4.40E+05	4.40E+05
LR	6.16E+05	6.16E+05	6.15E+05	6.16E+05	6.15E+05	6.16E+05
Musk	5.92E+09	5.92E+09	5.92E+09	5.92E+09	6.08E+09	6.10E+09
Statlog	4.35E+08	5.44E+08	4.35E+08	4.35E+08	5.70E+08	5.59E+08

Table 5
Effects of different settings of IKM−+ on runtime.

	Runtime (s)					
	v1	v2	v3	v4	v5	v6
Iris	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
HAR	5.55E+00	5.47E+00	5.55E+00	5.54E+00	3.42E+00	3.31E+00
ISOLET	7.30E+00	7.19E+00	7.53E+00	7.26E+00	1.00E+01	1.14E+01
LR	5.07E+00	4.20E+00	5.30E+00	4.76E+00	5.20E+00	5.55E+00
Musk	2.00E−02	2.00E−02	2.00E−02	2.00E−02	2.00E−02	2.00E−02
Statlog	3.90E−01	2.60E−01	4.10E−01	4.00E−01	5.50E−01	5.40E−01

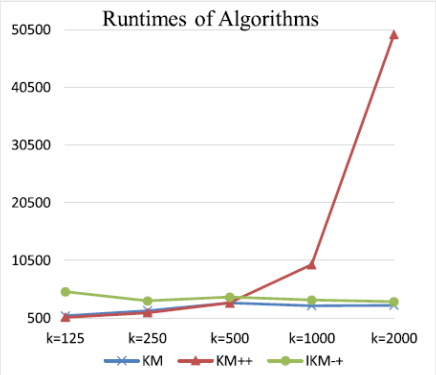
4.4 Results on the third type dataset

다음은 데이터 수도 가장 많고, cluster 개수도 가장 많은 protein dataset(KDDCUP04Bio) 데이터셋을 적용한 결과다.

Table 6
Results on KDDCUP04Bio.

	Maximum of partial SSEDMs			SSEDM			Runtime (s)			t_{IKM-+}/t_{KM}
	KM	KM++	IKM−+	KM	KM++	IKM−+	KM	KM++	IKM−+	
$k=125$	8.34E+10	1.70E+10	**1.58E+09**	2.30E+11	1.62E+11	**1.43E+11**	**851.35**	618.62	5041.59	5.92
$k=250$	8.09E+10	7.56E+09	**7.05E+08**	2.06E+11	1.31E+11	**1.21E+11**	**1771.26**	1389.27	3466.35	1.96
$k=500$	6.08E+10	1.69E+09	**3.47E+08**	1.71E+11	1.07E+11	**1.04E+11**	**3111.37**	3112.03	4096.47	1.32
$k=1000$	2.81E+10	7.60E+08	**2.11E+08**	1.26E+11	9.10E+10	**8.94E+10**	**2646.5**	9732.63	3624.23	1.37
$k=2000$	7.07E+09	3.82E+08	**1.72E+08**	9.21E+10	**7.71E+10**	7.73E+10	**2703.1**	49,635.6	3309.17	1.22

cluster의 개수가 증가할수록, k-means의 accuracy는 떨어지는 것을 확인할 수 있다. 반면에 IKM-+는 크게 달라지지 않는다. 가장 cluster의 개수가 많은 k=2000일 때, k-means++의 SSEDM이 가장 작지만, Runtime을 비교했을 때 k-means++는 굉장히 느려지는 것을 알 수 있다.



실제 각 알고리즘 별 cluster의 수를 증가시켰을 때 Runtime을 그래프로 나타낸 것이다. k-means가 가장 빠르고 그 다음으로 IKM-+였다. 반면 k=2000일 때 accuracy가 가장 좋았던 k-means++는 runtime이 폭발적으로 증가하였다. 따라서 accuracy와 runtime을 같이 고려했을 때, 가장 효율적인 알고리즘은 IKM-+이었다.

4.5 More experiments to consider effects of increasing number of clusters

Table 7
Effects of increasing number of clusters up to 100 on HAR.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	1.63E+05	1.63E+05	1.62E+05	1.01	1.01	3.56E+00	4.08E+00	1.60E+01
$k=25$	1.45E+05	1.45E+05	1.44E+05	1.01	1.01	7.74E+00	9.35E+00	2.96E+01
$k=50$	1.30E+05	1.30E+05	1.29E+05	1.01	1	1.53E+01	2.08E+01	2.76E+01
$k=100$	1.16E+05	1.16E+05	1.15E+05	1.01	1.01	2.19E+01	5.01E+01	2.98E+01

Table 8
Effects of increasing number of clusters up to 100 on ISOLET.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	5.16E+05	5.16E+05	5.14E+05	1	1	1.99E+00	2.20E+00	5.47E+00
$k=25$	4.50E+05	4.49E+05	4.43E+05	1.02	1.01	4.36E+00	5.38E+00	5.77E+00
$k=50$	4.06E+05	4.05E+05	4.01E+05	1.01	1.01	8.99E+00	1.41E+01	1.29E+01
$k=100$	3.70E+05	3.69E+05	3.67E+05	1.01	1.01	1.53E+01	3.65E+01	1.28E+01

Table 9
Effects of increasing number of clusters up to 100 on LR.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	8.13E+05	8.10E+05	8.08E+05	1.01	1	5.27E−01	5.98E−01	1.16E+00
$k=25$	6.27E+05	6.28E+05	6.25E+05	1	1.01	9.48E−01	1.16E+00	3.33E+00
$k=50$	4.85E+05	4.84E+05	4.78E+05	1.01	1.01	1.92E+00	2.42E+00	6.30E+00
$k=100$	3.67E+05	3.65E+05	3.59E+05	1.02	1.02	3.07E+00	5.57E+00	6.92E+00

Table 10
Effects of increasing number of clusters up to 100 on Musk.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	3.23E+09	3.23E+09	3.15E+09	1.03	1.03	2.65E−01	3.13E−01	5.17E−01
$k=25$	2.62E+09	2.63E+09	2.53E+09	1.04	1.04	5.75E−01	8.69E−01	1.28E+00
$k=50$	2.13E+09	2.12E+09	2.05E+09	1.04	1.03	1.24E+00	2.37E+00	2.09E+00
$k=100$	1.69E+09	1.69E+09	1.61E+09	1.05	1.05	2.34E+00	7.23E+00	1.83E+00

Table 11
Effects of increasing number of clusters up to 100 on Statlog.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=12$	5.60E+08	5.70E+08	2.20E+08	2.54	2.59	7.96E−01	8.82E−01	1.16E+00
$k=25$	3.77E+08	3.30E+08	9.10E+07	4.15	3.62	2.08E+00	1.93E+00	2.73E+00
$k=50$	3.08E+08	1.63E+08	3.00E+07	10.26	5.44	5.05E+00	4.99E+00	8.74E+00
$k=100$	9.98E+07	5.15E+07	1.10E+07	9.11	4.7	1.16E+01	1.36E+01	1.56E+01

Table 12
Effects of increasing number of clusters up to 1000 on Statlog.

	SSEDM			$SSEDM_{IKM-+}$/		Runtime (s)		
	KM	KM++	IKM−+	$SSEDM_{KM}$	$SSEDM_{KM++}$	KM	KM++	IKM−+
$k=125$	9.78E+07	4.14E+07	8.03E+06	12.19	5.16	1.38E+01	1.95E+01	1.14E+01
$k=250$	9.39E+07	1.77E+07	3.78E+06	24.85	4.7	2.68E+01	5.64E+01	1.65E+01
$k=500$	9.25E+07	6.64E+06	2.68E+06	34.48	2.47	4.51E+01	1.83E+02	2.48E+01
$k=1000$	9.05E+07	2.44E+06	1.05E+06	86.35	2.33	6.70E+01	6.80E+02	3.19E+01

위 Table7-12는 앞의 dataset중에서 k를 달리하여 알고리즘을 적용해 실행한 결과다. 이때에는 accuracy가 전체적으로 비슷한 반면에 오히려 runtime이 더 작아 다른 알고리즘에 비해 빨리 실행되었음을 알 수 있다. I-k-means-+가 다른 알고리즘에 비해 시간이 많이 걸릴 것이라 예상했을 수도 있지만, 오히려 몇몇 경우에는 더 빨리 실행되어 가장 효율적인 알고리즘이라고 말할 수 있을 듯 하다.

5. Conclusion

정리하면, I-k-means-+는 없앨 cluster(minus)와 쪼갤 cluster(plus)를 pair로 매칭해 k-means 성능을 향상시키는 알고리즘이다. 보기에 시간이 매우 소요될 것 같지만, 위에 나온 몇몇 heuristics를 이용해 IKM-+의 runtime을 줄일 수 있었다. 또한 데이터 수가 크고 cluster가 증가한 경우에 다른 알고리즘에 비교해 accuracy 뿐 아니라 runtime도 좋았음을 확인하였다. 데이터 수가 크고 cluster가 많은 경우에 IKM-+의 장점은 더욱 도드라질 것이다. 마지막으로, FCM과 같은 유사한 알고리즘에 적용한다면 k-means 알고리즘을 더 향상시킬 수 있을 것이라 기대한다.