Algorithms for Data Science: Lecture on Clustering

Barna Saha

1 Clustering

Given a set of points with a notion of distance between points, group the points into some number of clusters so that members of a cluster are "close" to each other, while members of different clusters are far. The problem of clustering is ubiquitous. We may want to cluster documents by topic they represent, we may want to cluster the moviegoers by the types of movies they like, or cluster genes by their pattern etc. The clustering may seem easy in low-dimensional Euclidean space, but becomes notoriously hard in higher dimensions and considering non-Euclidean distances.

Often, given a set of points in a space, we would like to cluster them in a way to optimize some goodness criteria that tells how closely knit the clusters are. There had been many such proposals of goodness criteria. Here we elaborate on three such popular clustering paradigms: k-center clustering, k-median clustering and k-means clustering.

2 k-center Clustering

In the k-center problem, the input is a set of vertices V with distances $d : V \times V \to \mathbb{R}$ between them, and the goal is to select k centers such that the maximum distance to the closest center is minimized over V.

The problem is NP-hard, and even cannot be approximated within any factor $\alpha \geq 1$ unless the distance d is a metric satisfying triangle inequality. Hence, we consider the Metric-k-center problem and develop an approximation algorithm.

Suppose there exists k centers in V such that the maximum distance that any vertex in V needs to travel to reach one of the centers is OPT. We will develop an algorithm in which each vertex will have to travel at most $2OPT$ instead, that is we will develop a 2-approximation algorithm for the metric-k center.

2.1 Greedy Algorithm

1. Pick any vertex $v \in V$ arbitrarily and declare it as the first center c_1.
2. For $i = 2$ to k do
   (a) Select the vertex from V that is farthest from the already chosen centers, that is maximizes the minimum distance to any of the chosen center–as the new ith center.

We will now show that this is a 2-approximation.

Theorem 1. *The greedy algorithm returns a 2-approximation for the k-center problem.*

Proof. Consider when the ℓth center c_ℓ is chosen. $\mathcal{C}_{\ell-1} = \{c_1, c_2, .., c_{\ell-1}\}$ denote the already chosen sets. It must happen that minimum distance from c_ℓ to $\mathcal{C}_{\ell-1}$ is lower than the minimum distance c_i to $\mathcal{C}_{i-1}$. If not, consider the following contradicting scenario.

Suppose minimum distance from c_ℓ to $\mathcal{C}_{\ell-1}$ is to c_j. Let $j \leq i-1$. Then while choosing the ith center, the algorithm will prefer c_ℓ over c_i as c_ℓ is farther than c_i from $\mathcal{C}_{i-1}$. So suppose

$j \geq i$. But then minimum distance of c_ℓ to $\mathcal{C}_{i-1}$ is only higher–so higher than c_i, and again the algorithm will prefer c_ℓ over c_i as c_ℓ is farther than c_i from $\mathcal{C}_{i-1}$. This gives a contradiction.

Now, after the k centers have been selected–consider the point $c_{k+1} \in V$ that is farthest from $\mathcal{C}_k$. If that distance is $\leq 2OPT$, we have a 2-approximation. So, suppose that distance is $> 2OPT$. That means by the previous argument, all pairwise distances between c_i and c_j, $i \neq j \in [1, k+1]$ is $> 2OPT$.

Then among these $k+1$ points, two must belong to the same optimum cluster and are thus within distance at most OPT from one of the optimum centers. By triangle inequality, then distance between them is at most $2OPT$ giving a contradiction. □

3 k-median Clustering

The k-center clustering is highly susceptible to outliers. To overcome this the clustering objective one can try to minimize the sum of the distances to the minimum cluster center. This is known as the k-median clustering.

Hence, in the k-median clustering, the goal is to select k centers $c_1, c_2, .., c_k$ from $\mathcal{C}$ such that $\sum_{v \in V} \min_{i=1}^{k} d(v, c_i))$ is minimized.

Again the problem is NP-hard, and here we give a simple *local search* algorithm that can be proven to give a good approximation factor.

3.1 Local Search

The following is a very intuitive algorithm to solve the k-median problem.

1. Start with any set of k-vertices from V as your centers $\mathcal{C}$, let $Cost(\mathcal{C})$ denote the value of objective function for $\mathcal{C}$ as solution, that is $Cost(\mathcal{C}) = \sum_{v \in V} \min_{c \in \mathcal{C}} d(v, c)$.

2. If there exists $x \in V \setminus \mathcal{C}$ and $Y \in \mathcal{C}$ such that swapping X and Y improves the solution then swap them. That is if $Cost(\mathcal{C} - Y + X) < Cost(\mathcal{C})$ then $Swap(X, Y)$.

3. Else return.

It can be shown that the above local search algorithm converges and gives a solution within 5 times of the optimum cost. There are better local search algorithms known that instead of swapping one vertex at a time, consider multiple vertices swapping.

4 k-means Clustering

We now look at the k-means clustering which is one of the oldest and popular clustering algorithms. In the k-means problem, instead of minimizing the sum of distances to the nearest center, we try to minimize the sum of the squared distances to the nearest center. k-means clustering is defined over the $\mathbb{R}^d$ space for Euclidean distance. Moreover, the center does not need to be a point in V itself. Given a set of points, we know if we select their mean as the cluster center that will minimize the total squared distance! (why?)—which is the reason behind the choice of name.

Solving the problem exactly is NP-Hard. 25 years ago, Llyod proposed another simple local search algorithm which till date is the most popular algorithm for k-means clustering. A 2002 survey of data mining techniques states that it "is by far the most popular clustering algorithm used in scientific and industrial applications."

4.1 Llyod's Algorithm

1. Begin with k arbitrary centers, typically chosen uniformly at random from the data points.
2. Assign each point to its nearest cluster center
3. Recompute the new cluster centers as the center of mass of the points assigned to the clusters
4. Repeat Steps 1-3 until the process converges.

It can be shown that the objective function is monotonically decreasing, which ensures no configuration is repeated. However, the convergence could be slow: k^n possible clusterings.

Moreover, the algorithm can converge to a local minima without any guarantee on the objective function value.

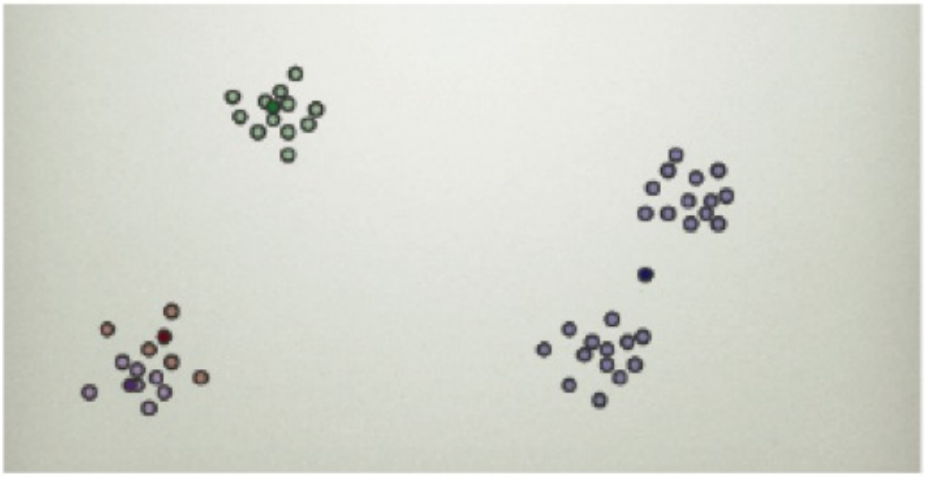


Figure 1: k-means get stuck in a local minima

4.2 Drawbacks of k-means clustering

- Euclidean distance is used as a metric and variance is used as a measure of cluster scatter.
- The number of clusters k is an input parameter: an inappropriate choice of k may yield poor results. That is why, when performing k-means, it is important to run diagnostic checks for determining the number of clusters in the data set.
- Convergence to a local minimum may produce results far away from optimal clustering

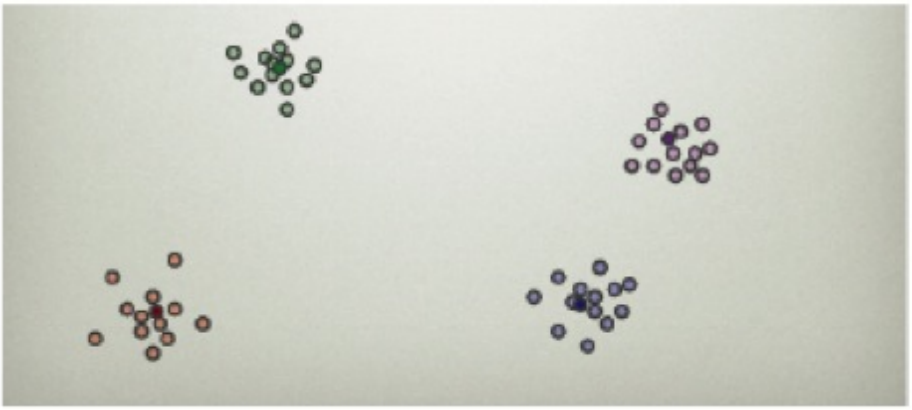


Figure 2: Alternative to select initial k clusters in k-means

4.3 k-means++

By selecting initial centers wisely k-means++ generalizes k-means and ensures the converged local minima has an objective value close to the actual optimum objective.

To overcome the scenario depicted in Figure **??**, one can try the following fix, where we always select the center that is farthest away from the currently opened centers.

However, such an assignment will be highly susceptive to outliers. For example, see Figure **??**.

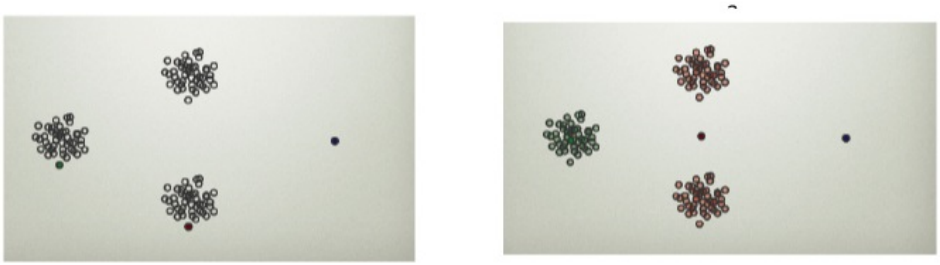


Figure 3: Susceptibility to Outliers

k-means++ interpolates between selecting centers randomly and selecting centers only by distance. Specifically, let $D(x)$ denote the minimum distance of x to the nearest chosen cluster centers. Then select x as a new center with probability $\frac{D(x)^2}{\sum_{v \in V} D(v)^2}$.

It can be shown by this small change k-means++ returns a solution which is within an $O(\log k)$-factor of the optimum objective value.

5 Flavor of the Analysis of k-means++ (Optional)

For the k-means problem, we are given an integer k and a set of n data points $\mathcal{X} \in \mathbb{R}^d$. The goal is to select k centers $\mathcal{C}$ to minimize the following objective.

$$\phi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} ||x - c||^2$$

That is we want to select k centers to minimize the sum of squared distances of all points to their respective nearest center.

k-means problem is NP-Hard. Lloyd's algorithm is a popular heuristic that is employed to solve k-means problem. However, depending on the distribution of the data points, it is possible that Lloyd's algorithm converges to a local optimum that is far from the global optimum. k-means++ is a way to ensure that Llyod's algorithm does not converge to a local optimum which is far off. In other words, even with k-means++, it is possible that the algorithm converges to a local optimum, but that local optimum is close to the global optimum.

5.1 Details of k-means++ Algorithm

Let $D(x)$ denote the distance of x from the closest center.

1. Take one center c_1, chosen uniformly at random from $\mathcal{X}$.
2. Take a new center c_i, choosing $x \in \mathcal{X}$ with probability $\frac{D(x)}{\sum_{x \in \mathcal{X}} D(x)^2}$
3. Repeat step 2 until we have taken k centers altogether.
4. Now proceed as with the standard k-means algorithm.

5.2 Analysis

Here is a standard result from Linear Algebra.

Lemma 1. *Let S be a set of points with center of mass $c(S)$, and let z be an arbitrary point. Then*

$$\sum_{x \in S} ||x - z||^2 - \sum_{x \in S} ||x - c(S)||^2 = |S||c(S) - z||^2$$

Theorem 2. *If C is constructed with k-means++, then the corresponding objective function ϕ satisfies*

$$E[\phi] \leq 8(\ln k + 2)\phi_{OPT}$$

.

The above claim holds after the first iteration. Later iterations can only improve the bound as ϕ decreases.

We will only show a partial result. We will show if cluster centers are chosen from each cluster then k-means++ is 8-competitive.

Theorem 3. *Let $\mathcal{C}$ contains the current centers that have already been chosen. Let A be a cluster in the optimal solution not represented in $\mathcal{C}$.*

If we add a random center to $\mathcal{C}$ from A, chosen with D^2 weighing then

$$E[\phi(A)] \leq 8\phi_{OPT}(A)$$

.

Proof.

$$E[\phi(A)] = \sum_{a_0 \in A} \frac{D(a_0)^2}{\sum_{a \in A} D(a)^2} \sum_{a \in A} \min\left(D(a)^2, ||a - a_0||^2\right)$$

Let z be the closest center to some $a \in A$. Using triangle inequality, we have

$$D(a_0) \leq ||a_0 - z||_2 \leq D(a) + ||a - a_0||_2$$

Therefore,

$$D(a_0)^2 \leq (D(a) + ||a - a_0||_2)^2 \leq 2D(a)^2 + 2||a - a_0||^2$$

Summing over all $a \in A$, we get

$$|A|D(a_0)^2 \leq 2\sum_{a\in A} D(a)^2 + 2\sum_{a\in A} ||a - a_0||^2$$
$$\text{or}\,, D(a_0)^2 \leq \frac{2}{|A|}\sum_{a\in A} D(a)^2 + \frac{2}{|A|}\sum_{a\in A} ||a - a_0||^2$$

Using the above in the expression for $E[\phi(A)]$ we get

$$\begin{aligned}
&E[\phi(A)] \\
&= \sum_{a_0\in A} \frac{D(a_0)^2}{\sum_{a\in A} D(a)^2} \sum_{a\in A} \min\left(D(a)^2, ||a - a_0||^2\right) \\
&\leq \sum_{a_0\in A} \frac{\frac{2}{|A|}\sum_{a\in A} D(a)^2}{\sum_{a\in A} D(a)^2} \sum_{a\in A} \min\left(D(a)^2, ||a - a_0||^2\right) + \sum_{a_0\in A} \frac{\frac{2}{|A|}\sum_{a\in A} ||a - a_0||^2}{\sum_{a\in A} D(a)^2} \sum_{a\in A} \min\left(D(a)^2, ||a - a_0||^2\right) \\
&\leq \sum_{a_0\in A} \frac{\frac{2}{|A|}\sum_{a\in A} D(a)^2}{\sum_{a\in A} D(a)^2} \sum_{a\in A} ||a - a_0||^2 + \sum_{a_0\in A} \frac{\frac{2}{|A|}\sum_{a\in A} ||a - a_0||^2}{\sum_{a\in A} D(a)^2} \sum_{a\in A} D(a)^2 \\
&= \frac{4}{|A|}\sum_{a_0\in A}\sum_{a\in A} ||a - a_0||^2
\end{aligned}$$

Now use Lemma **??**.

$$\begin{aligned}
E[\phi(A)] &= \frac{4}{|A|}\sum_{a_0\in A}\sum_{a\in A} ||a - a_0||^2 \\
&= \frac{4}{|A|}\sum_{a_0\in A}\sum_{a\in A} ||a - c(A)||^2 + |A|(||a_0 - c(A)||^2) \\
&= 8\sum_{a\in A} ||a - c(A)||^2 = 8\phi_{OPT}(A)
\end{aligned}$$

This completes the proof.

□

5.3 More Local Search Algorithm & Hybrid Algorithm

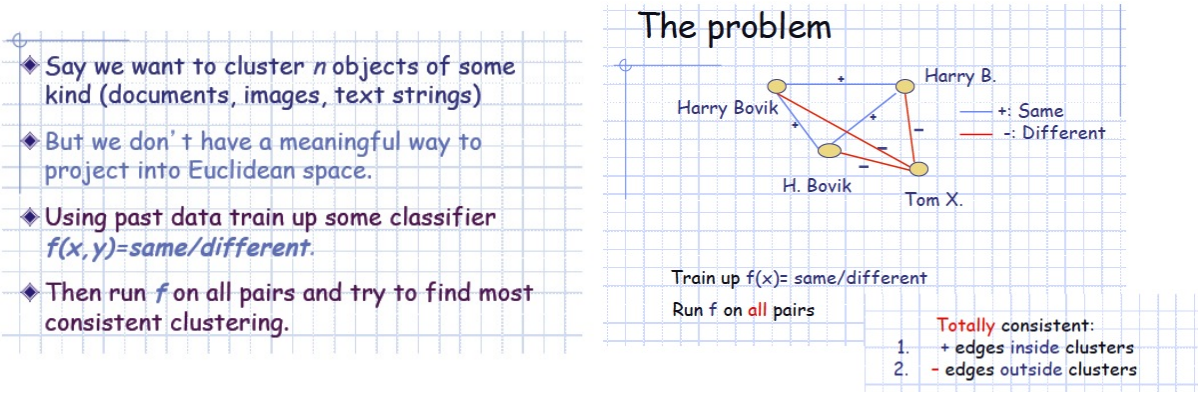
One may ask if we can design a local search algorithm in the same spirit as the k-median clustering. Note that in the k-median clustering, the centers come from the input points whereas in k-means clustering, they can be any points in $\mathbb{R}^d$. Still, we can run the same local search algorithm we learnt for k-median, and can get a $(9+\epsilon)$-approximation [**?**][1]. In fact, [**?**] shows that by combining the local search with Llyod's algorithm, we can get a very good performance. The algorithm in a nut shell is as follows:

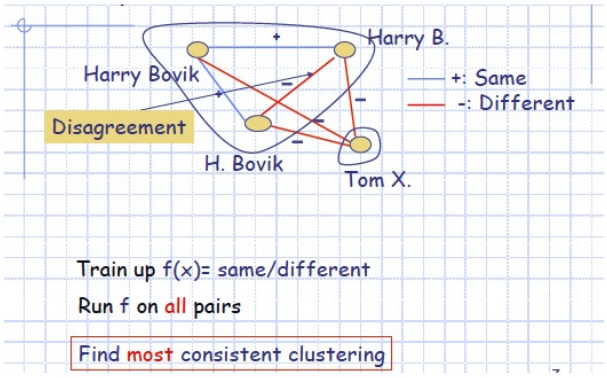
- Start with k initial centers chosen at random.
- Run Llyod's algorithm to obtain the new centers.
- Now do a local search by swapping centers to obtain an improvement, and repeat the last two steps until no such improvements can be found.

[1] the local search described in [**?**] swaps multiple vertices at a time to get the desired approximation.

6 Clustering with unknown k: Correlation Clustering

A major drawback of the clustering algorithms that we have learnt so far is that they need to know k in advance, that is the number of clusters. In this section, we describe a clustering algorithm that does not need any prior knowledge of k. The setting is as follows.





Definition (Correlation Clustering)**.** Given a complete graph on n vertices, with each edge labeled $+$ or $-$, find partition of vertices as consistent as possible with edge labels.

- **Minimize Disagreement** minimize the total number of intra-cluster $-$ edges and inter-cluster $+$ edges.
- **Maximize Agreement** maximize the total number of intra-cluster $+$ edges and inter-cluster $-$ edges.

Correlation clustering is NP-Hard. Therefore, unless $P = NP$, we cannot hope for any polynomial time exact algorithm for it. We therefore resort to developing approximation algorithm.

6.1 Correlation Clustering on Noisy Input

We can think of the classifier to flip the correct label for every pair with some probability $p < \frac{1}{2}$.

Let us for simplicity assume that the classifier always correctly labels the $-$ edges, and makes error with probability p on $+$ edges. Take $p = \frac{1}{2} - \delta$ for some constant $\delta > 0$. Therefore, if we see $+$-labeled edges, they are genuinely $+$. However, an observed $-$-edge may very well have an actual $+$-label.

Let $N^+(v)$ denote the positive neighbors of any vertex v (including v itself), that is v is associated with every vertex in $N^+(v)$ with an edge labeled $+$. Then, clearly, if u and v are in different clusters $|N^+(u) \cap N^+(v)| = \phi$.

Now suppose u and v are in the same cluster. Let w be a node in the same cluster, different from both u and v. Then

$$\Pr[label(u,w) = + \text{ and } label(v,w) = +] = (1-p)^2 = (\frac{1}{2} + \delta)^2$$

$$E[|N^+(u) \cap N^+(v)|] \geq (1-p)^2|C| = (\frac{1}{2} + \delta)^2|C| \geq \frac{|C|}{4}$$

where C is the true cluster containing u and v. By the Chernoff bound, if $|C| = \omega(\log n)$, then with high probability $|N^+(u) \cap N^+(v)| > 0$.

With the simpler bounds that you learnt in the class, consider $|C| \geq 96 \log n$, then

$$\Pr[|N^+(u) \cap N^+(v)| \leq \frac{|C|}{8}] = \Pr[|N^+(u) \cap N^+(v)| \leq \frac{|C|}{4}(1 - \frac{1}{2})] \leq e^{-\frac{|C|}{4}\frac{1}{8}} = \frac{1}{n^3}$$

By union bound over at most $\binom{n}{2}$ vertex pairs,

$$\Pr[\exists u \text{ and } v \text{ s.t they belong to the same cluster with no common positive neighbor}] \leq \frac{1}{n}$$

The above analysis suggests the following algorithm.

- While all items are not clustered
  - Create a new cluster C by picking an unclustered vertex u, and including u and all vertices v such that $|N^+(u) \cap N^+(v)| > 0$ in C. Call all the vertices in C as clustered.

Notes. Interestingly, an algorithm very close to the spirit can be designed when noise is adversarial rather than random to return a good approximate answer. We first consider a random permutation of vertices. Instead of choosing an arbitrary vertex u inside of the while loop, we pick the vertex first in the permutation that is yet to be clustered. We next consider its entire positive neighborhood as a single cluster. This falls under the category of agnostic learning.

Exercise 1. *Design an algorithm for correlation clustering when the noise can be two-sided that is both $+$ and $-$ labeled edges can be flipped with probability $\frac{1}{2}$.*

7 Clustering with an Oracle

In order to run correlation clustering, we would need to get labelled data for every $\binom{n}{2}$ pairs which can be very costly to obtain. Now, we ask the question, is it possible to obtain only a few labelled data and yet be able to recover all the underlying clusters?

Suppose for the time-being, whenever, we ask for labels of a particular pair, the classifier returns the correct answer. How many such labeled data, do you need to obtain the clusters exactly. Answer: nk. Why?

1. For every unclustered vertex u
   (a) For every non-empty cluster C formed so far
      i. Pick a vertex v from C and ask for the label of (u, v). If the label is $+$, include u in C and move to (1).
2. If none of the labels asked had label $+$ in (i) for u, create a new cluster $\{u\}$.

Exercise 2 (Extra-credit)**.** *What if the classifier returns wrong answer with probability $p < \frac{1}{2}$? Design an algorithm that asks minimum number of queries and returns the clusters with high probability.*

References

[1] Kanungo, Tapas, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. "A local search approximation algorithm for k-means clustering." In Proceedings of the eighteenth annual symposium on Computational geometry, pp. 10-18. ACM, 2002.