Supplementary Materials for "SparseKmeans: Efficient K-means Clustering For Sparse Data"

Khoi Nguyen Pham Dang*
MBZUAI
khoinguyen.phamdang@mbzuai.ac.ae

He-Zhe Lin*
MBZUAI
b07902028@csie.ntu.edu.tw

Chih-Jen Lin
National Taiwan Univ. / MBZUAI
cjlin@csie.ntu.edu.tw

A STOPPING CONDITION OF K-MEANS ALGORITHM

SparseKmeans follows the implementation in scikit-learn to stop the iteration when the new centroids are close enough to their corresponding old ones. Specifically, the procedure is terminated when

$$\sum_{j=1}^{K} \|\boldsymbol{c}_j - \hat{\boldsymbol{c}}_j\|^2 \le \tau \times \frac{1}{n}\sum_{k=1}^{n} \sigma_k^2, \tag{A.1}$$

where $\boldsymbol{c}_j$'s and $\hat{\boldsymbol{c}}_j$'s are current and previous centroids, τ is a pre-defined parameter, and σ_k^2 is the variance of the k-th component in each $\boldsymbol{x}_i$, respectively.

In SparseKmeans, we store the current centroids and previous centroids as two matrices C and $\hat{C}$:

$$C = \begin{bmatrix} - & \boldsymbol{c}_1^T & - \\ - & \boldsymbol{c}_2^T & - \\ & \vdots & \\ - & \boldsymbol{c}_K^T & - \end{bmatrix} \text{ and } \hat{C} = \begin{bmatrix} - & \hat{\boldsymbol{c}}_1^T & - \\ - & \hat{\boldsymbol{c}}_2^T & - \\ & \vdots & \\ - & \hat{\boldsymbol{c}}_K^T & - \end{bmatrix}.$$

To calculate $\|\boldsymbol{c}_j - \hat{\boldsymbol{c}}_j\|^2$ in (A.1), we first take the subtraction between C and $\hat{C}$:

$$C - \hat{C} = \begin{bmatrix} \boldsymbol{c}_1^T - \hat{\boldsymbol{c}}_1^T \\ \boldsymbol{c}_2^T - \hat{\boldsymbol{c}}_2^T \\ \vdots \\ \boldsymbol{c}_K^T - \hat{\boldsymbol{c}}_K^T \end{bmatrix},$$

and then calculate the squared norms of $C - \hat{C}$

$$\begin{bmatrix} \|\boldsymbol{c}_1^T - \hat{\boldsymbol{c}}_1^T\|^2 \\ \|\boldsymbol{c}_2^T - \hat{\boldsymbol{c}}_2^T\|^2 \\ \vdots \\ \|\boldsymbol{c}_K^T - \hat{\boldsymbol{c}}_K^T\|^2 \end{bmatrix} \tag{A.2}$$

Summing all elements in (A.2), we have the left-hand side of (A.1).

B K-MEANS++ INITIALIZATION

The Initialization of $\boldsymbol{c}_j$'s are often by choosing a subset of samples. In scikit-learn, they use K-means++ initialization [1] as its default setting. Compared to randomly choosing K points as the initial centroids, K-means++ reduces the number of iterations for convergence and minimizes the risk of poor clustering results.

K-means++ sequentially selects the initial points in a more optimal way by prioritizing points that are far from the previously chosen centroids. Therefore, the moment of choosing $\boldsymbol{c}_j$, we already have $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{j-1}$. We then maintain a distance vector $\boldsymbol{d'} \in \mathbb{R}^m$ with

$$d'_i = \min_{j'=1,\ldots,j-1} \|\boldsymbol{x}_i - \boldsymbol{c}_{j'}\|^2,$$

which is the squared distance between $\boldsymbol{x}_i$ and the nearest point that has been chosen as a centroid. Then, based on $\boldsymbol{d'}$, we pick L candidates for $\boldsymbol{c}_j$, where points further from the already chosen centroids are more likely to be selected. After selecting L candidates, we assign the one that yields the least total squared distance between data and centroids (called "potential") as $\boldsymbol{c}_j$. The complete procedure of K-means++ is as follows.

(1) Uniformly choose $\boldsymbol{c}_1$ from $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ and initialize $d'_i = \|\boldsymbol{x}_i - \boldsymbol{c}_1\|^2$.
(2) For $j = 2, \ldots, K$, do the following to select $\boldsymbol{c}_j$.
   (a) Sample $L = 2+\log_2 K$ candidate points from $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$ using the distribution
   $$\Pr(\boldsymbol{x}_i) = \frac{d'_i}{\sum_{i'=1}^{m} d'_{i'}}.$$
   Denote $S = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_\ell\}$ as the sampled points.
   (b) Calculate the potential for assigning each $s \in S$ as $\boldsymbol{c}_j$. Find the candidate point in S that minimizes the potential as $\boldsymbol{c}_j$, i.e.,
   $$\boldsymbol{c}_j \in \arg\min_{\boldsymbol{s}\in S} \sum_{i=1}^{m} \min\{d'_i, \|\boldsymbol{x}_i - \boldsymbol{s}\|^2\}. \tag{B.1}$$
   (c) Update
   $$d'_i \leftarrow \min\{d'_i, \|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2\}, i = 1, \ldots, m.$$

C DETAILS OF ELKAN'S CLUSTER ASSIGNMENT

C.1 The Maintenance of $u(\boldsymbol{x}_i)$ and $l(\boldsymbol{x}_i, \boldsymbol{c}_j)$

- For each $\boldsymbol{x}_i$ in cluster-ℓ_i, $u(\boldsymbol{x}_i)$ is an upper bound of $\|\boldsymbol{x}_i - \boldsymbol{c}_{\ell_i}\|$. Since the centroids are updated after the cluster assignment, we need to update $u(\boldsymbol{x}_i)$ correspondingly. Suppose $\hat{\boldsymbol{c}}_{\ell_i}$ the centroid for cluster-ℓ_i before the centroid updates and $\hat{u}(\boldsymbol{x}_i)$ is an upper bound of $\|\boldsymbol{x}_i - \hat{\boldsymbol{c}}_{\ell_i}\|$. To update $u(\boldsymbol{x}_i)$ from $\hat{u}(\boldsymbol{x}_i)$, by the triangle inequality, we have
  $$\|\boldsymbol{x}_i - \hat{\boldsymbol{c}}_{\ell_i}\| + \|\hat{\boldsymbol{c}}_{\ell_i} - \boldsymbol{c}_{\ell_i}\| \ge \|\boldsymbol{x}_i - \boldsymbol{c}_{\ell_i}\|.$$
  Therefore, after centroid updates, $u(\boldsymbol{x}_i)$ can be obtained via
  $$u(\boldsymbol{x}_i) \leftarrow \hat{u}(\boldsymbol{x}_i) + \|\boldsymbol{c}_{\ell_i} - \hat{\boldsymbol{c}}_{\ell_i}\|. \tag{C.1}$$
- For each $\boldsymbol{x}_i$ and any $j = 1, \ldots, K$, $l(\boldsymbol{x}_i, \boldsymbol{c}_j)$ is a lower bound of $\|\boldsymbol{x}_i - \boldsymbol{c}_j\|$. Similar to the situation in $u(\boldsymbol{x}_i)$, since $\boldsymbol{c}_j$ is changed afer the centroid updates, we need to update $l(\boldsymbol{x}_i, \boldsymbol{c}_j)$ for all i. Suppose $\hat{\boldsymbol{c}}_j$ is the centroid of cluster-j before the centroid updates and $\hat{l}(\boldsymbol{x}_i, \hat{\boldsymbol{c}}_j)$ is a lower bound of $\|\boldsymbol{x}_i - \hat{\boldsymbol{c}}_j\|$. By the triangle inequality and the definition of $\hat{l}(\boldsymbol{x}_i, \hat{\boldsymbol{c}}_j)$, we have
  $$\|\boldsymbol{x}_i - \boldsymbol{c}_j\| \ge \|\boldsymbol{x}_i - \hat{\boldsymbol{c}}_j\| - \|\hat{\boldsymbol{c}}_j - \boldsymbol{c}_j\| \ge \hat{l}(\boldsymbol{x}_i, \hat{\boldsymbol{c}}_j) - \|\hat{\boldsymbol{c}}_j - \boldsymbol{c}_j\|.$$

*Both authors contributed equally to this research.

Algorithm C.1: The original version of Elkan's cluster assignment in [2].

1 Compute $D \in \mathbb{R}^{K\times K}$ with $D_{j,j'} = \|\boldsymbol{c}_j - \boldsymbol{c}_{j'}\|/2$
2 **for** $i = 1, \ldots, m$ **do**
3 **if** $u(\boldsymbol{x}) \le \min_{j\neq\ell_i} D_{j,\ell_i}/2$ **then**
4 **continue**
5 *bound_tight* $\leftarrow$ False
6 **for** $j = 1, \ldots, K$ **do**
7 **if** $j \neq \ell_i$ **and** $u(\boldsymbol{x}_i) > D_{j,\ell_i}/2$ **and** $u(\boldsymbol{x}_i) > l(\boldsymbol{x}_i, \boldsymbol{c}_j)$ **then**
  // filter clusters that $\boldsymbol{x}_i$ must not belong to
8 **if** *bound_tight* **is** False **then**
9 $d_i \leftarrow \|\boldsymbol{x}_i - \boldsymbol{c}_{\ell_i}\|$
10 $u(\boldsymbol{x}_i) \leftarrow d_i$, $l(\boldsymbol{x}_i, \boldsymbol{c}_{\ell_i}) \leftarrow d_i$
11 *bound_tight* $\leftarrow$True
12 Go to Line 7
13 $l(\boldsymbol{x}_i, \boldsymbol{c}_j) \leftarrow \|\boldsymbol{x}_i - \boldsymbol{c}_j\|$
14 **if** $d_i > \|\boldsymbol{x}_i - \boldsymbol{c}_j\|$ **then**
15 $\ell_i \leftarrow j$, $u(\boldsymbol{x}_i) \leftarrow \|\boldsymbol{x}_i - \boldsymbol{c}_j\|$

Therefore, after centroid updates, $l(\boldsymbol{x}_i, \boldsymbol{c}_j)$ can be updated by

$$l(\boldsymbol{x}_i, \boldsymbol{c}_j) \leftarrow \max\left\{\hat{l}(\boldsymbol{x}_i, \hat{\boldsymbol{c}}_j) - \|\boldsymbol{c}_j - \hat{\boldsymbol{c}}_j\|, 0\right\}. \tag{C.2}$$

C.2 Elkan's K-means Assignment in the Orignal Work [2]

In the main paper, we give the sketch of Elkan's K-means assignment in the original work of [2]. Here we provide a detailed version in Algorithm C.1. In particular, there are two additional settings:

(1) (Line 3 to Line 4) To avoid the for-loop over j in Algorithm 1, Elkan [2] uses considers the following condition

$$u(\boldsymbol{x}) \le \frac{1}{2}\min_{j:j\neq\ell_i} \|\boldsymbol{c}_j - \boldsymbol{c}_{\ell_i}\|. \tag{C.3}$$

If (C.3) holds, then condition (6) must hold for all $j \neq \ell_i$, which means $\boldsymbol{x}$ must remain in its original cluster.

(2) To further reduce the number of distance calculations, when the distance $\|\boldsymbol{x}_i - \boldsymbol{c}_j\|$ may need to be calculated, we update $u(\boldsymbol{x}_i)$ to be the tighest upper bound $\|\boldsymbol{x}_i - \boldsymbol{c}_{\ell_i}\|$. We illustrate the idea in the following figure:

$$\overbrace{1, \ldots, (j^*-1),}^{bound\_loose} \underbrace{\quad j^* \quad}_{\text{Line 8 invoked}} \overbrace{, (j^*+1), \ldots, K}^{bound\_tight}. \tag{C.4}$$

Suppose j^* is the first cluster satisfying neither (6) nor (7). For $j = 1, \ldots, j^*-1$, we use $u(\boldsymbol{x}_i)$ to check condition (6) and (7). When $j = j^*$, we need both $\|\boldsymbol{x}_i - \boldsymbol{c}_{\ell_i}\|$ and $\|\boldsymbol{x}_i - \boldsymbol{c}_{j^*}\|$ to determine whether $\boldsymbol{x}_i$ should stay in cluster-ℓ_i or go to cluster-j^*. Under this situation, we update $u(\boldsymbol{x}_i)$ to $\|\boldsymbol{x}_i - \boldsymbol{c}_{\ell_i}\|$, we go to Line 7 and recheck the conditions. Then, from $j = j^*+1, \ldots, K$, we maintain $u(\boldsymbol{x}_i) = \|\boldsymbol{x}_i - \boldsymbol{c}_{\ell_i}\|$ in Line 15 while ℓ_i may be changed. Since we use the smallest possible value of $u(\boldsymbol{x}_i)$, the number of distance calculations can be reduced.

D ADDITIONAL IMPLEMENTATION DETAILS FOR UPDATING CENTROIDS

In Section 3.1, we mentioned that when updating centroids, we need to maintain a vector $\boldsymbol{d}$ with $d_i = \|\boldsymbol{x}_i - \boldsymbol{c}_{\ell_i}\|$, i.e., the distance to the nearest centroid. This allows us to replace the $u(\boldsymbol{x}_i)$ in (6) and (7) with d_i without additional costs. Here we explain why $\boldsymbol{d}$ is needed in centroid updates.

The main reason is, after cluster assignment, it is possible that none of $\boldsymbol{x}_i$'s belongs to some cluster-j, so we cannot update $\boldsymbol{c}_j$ by (2). Suppose we have E empty clusters after the cluster assignment. Then scikit-learn would find E samples with the top-E largest d_i values, and reassign them to these empty clusters.

E DATASETS

This section describes the preprocessing method applied to the four data sets across all experiments. For binary classification data set Url[1], we directly use the data instances for clustering. Other data sets including Amazon-670K[2], Wiki-500K[3] and Amazon-3M[4] are multi-label data sets, in which each instance can be associated with several labels. In the context of this work, to avoid confusion with clustering labels, we refer to the original labels of these mult-label data sets as meta-labels. Instead of clustering on data instances, we preprocessed the data to obtain a set of representations for meta-labels. These representations are then normalized and used as input for the K-means clustering algorithms. To clarify, let k be the total number of data instances, $\boldsymbol{z}_i$ be the TF-IDF representation of i^{th} instance, m be the total number of meta-labels and $\boldsymbol{x}_j$ be the representation of j^{th} meta-label. We call $Y \in \mathbb{R}^{k\times m}$ as a binary matrix, such that:

$$Y_{ij} = \begin{cases} 1 & \text{if } \boldsymbol{z}_i \text{ is associated with the meta-label } j^{th} \\ 0 & \text{otherwise} \end{cases}. \tag{E.1}$$

To obtain a meta-label representation, we compute the sum over all data instances corresponding to that meta-label.

$$\boldsymbol{x}_j = \frac{\sum_{i:Y_{ij}=1} \boldsymbol{z}_i}{\|\sum_{i:Y_{ij}=1} \boldsymbol{z}_i\|}. \tag{E.2}$$

The process could be formalized as matrix multiplication:

$$X = Y^T Z \tag{E.3}$$

following by a normalization for each row $\boldsymbol{x}_i$, in which $X \in \mathbb{R}^{m\times n}$ and $Z \in \mathbb{R}^{k\times n}$ are matrices of meta-label representations and instances representations, respectively.

REFERENCES

[1] David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. 1027–1035.

[2] Charles Elkan. 2003. Using the triangle inequality to accelerate k-means. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML)*. 147–153.

[1] https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/binary/url_combined_normalized.bz2
[2] https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/multilabel/Amazon-670K_tfidf_train_ver2.svm.bz2
[3] https://drive.google.com/open?id=1bGEcCagh8zaDV0ZNGsgF0QtwjcAm0Afk
[4] https://drive.google.com/open?id=187vt5vAkGI2mS2WOMZ2Qv48YKSjNbQv4