Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Scalable Sequential Spectral Clustering

Yeqing Li[1], **Junzhou Huang**[1*], **Wei Liu**[2]
[1]University of Texas at Arlington, Texas, USA
[2]Didi Research, Bejing, China
{yeqing.li@mavs.uta.edu, jzhuang@uta.edu, weiliu@didichuxing.com}

Abstract

In the past decades, Spectral Clustering (SC) has become one of the most effective clustering approaches. Although it has been widely used, one significant drawback of SC is its expensive computation cost. Many efforts have been devoted to accelerating SC algorithms and promising results have been achieved. However, most of the existing algorithms rely on the assumption that data can be stored in the computer memory. When data cannot fit in the memory, these algorithms will suffer severe performance degradations. In order to overcome this issue, we propose a novel sequential SC algorithm for tackling large-scale clustering with limited computational resources, *e.g.*, memory. We begin with investigating an effective way of approximating the graph affinity matrix via leveraging a bipartite graph. Then we choose a smart graph construction and optimization strategy to avoid random access to data. These efforts lead to an efficient SC algorithm whose memory usage is independent of the number of input data points. Extensive experiments carried out on large datasets demonstrate that the proposed sequential SC algorithm is up to a thousand times faster than the state-of-the-arts.

Introduction

Clustering is one of the most important problems in computer vision and pattern recognition. In the past decades, spectral clustering methods, *e.g.*, (Shi and Malik 2000; Von Luxburg 2007; Ng et al. 2002; Zelnik-Manor and Perona 2004; Yu and Shi 2003; Chang et al. 2015), which are based on eigendecompositions over graph Laplacians of data, have shown superior performance to traditional methods like K-Means. The effectiveness of spectral clustering methods is mainly due to its capability of capturing nonlinear cluster structures, while many other clustering methods rely on Euclidean geometry and explicit or implicit assumptions of convex shapes of clusters (Yan, Huang, and Jordan 2009). However, conventional spectral clustering algorithms are not applicable to large-scale data because of their quadratic computational complexity. This is mainly due to the expensive cost of the similarity graph construction and the eigendecomposition computation. Given n data points and each data point represented by a feature vector with d dimensions, the time complexity of computing a pairwise similarity matrix is $O(n^2d)$ and that of the worst-case eigendecomposition of a full graph Laplacian is $O(kn^2)$, where k is the number of classes. Given the rapid growth in the amount of user-generated data, such a quadratic complexity makes SC inapplicable in handling big data encountered in real-world problems.

Many approaches have been proposed to reduce the computational cost of spectral clustering. They can be roughly divided into two classes. The algorithms of the first class focus on reducing the computational cost of eigendecomposition. Given a similarity matrix of n data points, these algorithms accelerate the eigendecomposition by faster approximation approaches (Fowlkes et al. 2004; Chen et al. 2006; Liu et al. 2007; Khoa and Chawla 2012; Liu et al. 2013b; Zhang and Kwok 2010). Although promising results have been achieved, the construction of the similarity matrix still takes quadratic running time, which limits their usage. The algorithms of the second class try to reduce the computational cost of graph construction and eigendecomposition simultaneously by approximating the original graph (Shinnou and Sasaki 2008; Sakai and Imiya 2009; Yan, Huang, and Jordan 2009; Chen and Cai 2011; Li et al. 2015). In general, algorithms in the second class are more practical than those in the first class since they reduce the cost of all stages of spectral clustering. However, these algorithms are all in-memory algorithms, which assume that all data and intermediate results can be accessed in main memory. This assumption significantly limits the scale of data that they can handle.

It is a challenging research problem to efficiently handle data beyond the memory capacity. In this context, we could split the algorithm running time into two parts: **1)** time for processing data in memory; **2)** time for transferring data from/to disk. Most of the previous large-scale spectral clustering algorithms assume that the transfer time is negligible, which makes them impractical for big data. Other researchers try to reduce the second part by developing algorithms in distributed systems (Chen et al. 2011; Miao et al. 2008). However, not only is writing programs on a distributed system a challenging task, but also deploying and tuning a distributed program require specialized skills.

[*]Corresponding author: Junzhou Huang. Email: jzhuang@uta.edu. This work was partially supported by NSF IIS-1423056, CMMI-1434401, CNS-1405985.

This paper aims to develop large-scale spectral clustering algorithms for ordinary users with only limited computing resources. We consider the situation that data cannot be completely stored in memory. Therefore, it is inevitable to swap data back and forth between memory and disk. The key idea of our approach is to divide the entire data set into small blocks and sequentially process each block, which can keep the memory footprint small and the computational cost low. Inspired by the current progress of large-scale graph construction involved in semi-supervised learning (Liu, He, and Chang 2010; Liu, Wang, and Chang 2012), we propose a fast and scalable algorithm for spectral clustering named Sequential Spectral Clustering (SeqSC). Specifically, we approximate the original similarity graph by a bipartite graph and use two key components to drastically reduce the computational time and data transfer time. The first component is a novel sequential K-Means algorithm framework, while the second component is a sequential eigendecomposition approach. The proposed SeqSC is able to run in close-to-linear time with respect to the number of data points without losing the clustering accuracy. This requires scanning the raw data on disk for only a few passes. To our best knowledge, this is the first exploration of a large-scale spectral clustering in the case with limited resources. Extensive experiments are conducted in a real environment without enough memory for massive data, where the proposed algorithm is up to several orders of magnitudes faster than the state-of-the-art approaches.

Background and Notations

In this section, we will briefly introduce the notations and the spectral clustering algorithm. Suppose $X = [x_1, \dots, x_n]^T \in \mathbb{R}^{n \times d}$ denotes the data matrix, where n is the number of data points and d is the dimension of features. Each data point $x_i \in \mathbb{R}^d$ belongs to one of K classes $C = \{c_1, \dots, c_K\}$. Given the whole dataset X, each data point is represented as a vertex on the graph and each edge represents the affinity relation of one pair of vertices. In practice, the k-NN graph is usually used. Specifically, x_i and x_j are connected if at least one of them is among the k nearest neighbors in the given metric (usually Euclidean distance). The weight of the edge between x_i and x_j is defined as:

$$w_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|}{2\sigma^2}\right), & \text{if } x_i \text{ and } x_j \text{ are connected} \\ 0, & \text{otherwise,} \end{cases}$$

where σ is the bandwidth parameter. Note that while we use Gaussian Kernel for our example, this approach is also applicable to other types of kernels. Thus, $W = \{w_{ij}\} \in \mathbb{R}^{n \times n}$ is the adjacent matrix of the graph and it is a symmetric undirected graph. Let $D \in \mathbb{R}^{n \times n}$ be the degree matrix, the i-th diagonal element of which is $d_{ii} = \sum_{j=1}^{n} w_{ij}$. Let L denotes the normalize graph Laplacian matrix, then it is defined as:

$$L = I - D^{-1/2} W D^{-1/2} \tag{1}$$

The objective function of normalized spectral clustering (Ng et al. 2002) is defined as:

$$\min_{G^T G = I} \operatorname{Tr}(G^T L G), \tag{2}$$

where $G \in \mathbb{R}^{n \times K}$ is the class indicator matrix of all data. One solution of G in Eq. (2) is the K smallest eigen vectors of L. The whole process takes at least quadratic time in terms of n, which is impractical for the large-scale problem.

Methodology

In order to address the scalability problem of SC, we propose Sequential SC (SeqSC) approach by sequentializing the two key steps of SC: graph construction and eigendecomposition.

Graph Construction

As discussed above, the construction of the graph W takes quadratic space and time because of the computation of pairwise distance of data points. This process is easy to sequentialize. Specifically, we can keep only one sample of data x_i in the memory and then load all the other data from the disk sequentially and compute the distances from x_i to all the other data points. This algorithm is equivalent to sequentially computing each row of W. However, there are still two problems of this approach. First of all, the time complexity of this approach is still quadratic. Second, it requires scanning the data for multiple times (usually n divided by some constant times), which may take even longer than the computation itself for large data set.

To reduce the time complexity, one must eliminate the need of computing pairwise distances. A common way to achieve this goal is to find a low-rank approximation to W by using the bipartite graph. The intuition of this approach is instead of computing direct distance between two points x_i and x_j, we construct some local "station" u_k within their neighborhoods and then approximate the distance between x_i and x_j by $dist(x_i, x_j) \approx dist(x_i, u_i) + dist(u_i, x_j)$. This approximation is the distance of traveling from one point to another through a nearby transit point u_k. The number of stations should be much smaller than the data size so that the number of distances that one need to compute is much less than n^2. Therefore, instead of computing W, we employ the approach of anchor graph (Liu, He, and Chang 2010). Here, a small set of anchor points $U = [u_1, \dots, u_m]^T \in \mathbb{R}^{m \times d}$ is used to capture the manifold structure. These anchor points are usually chosen by running a lightweight clustering algorithms such as K-Means on raw data.

With the generated points, the k-NN graph is constructed between the raw data and the anchor points. Connections are only allowed between raw data points and anchor points. This constraint results in a bipartite graph between raw data X and anchor points U. The weight of each edge is defined as

$$Z_{ij} = \frac{K(x_i, u_j)}{\sum_{k \in \Phi_i} K(x_i, u_k)}, \forall j \in \Phi_i, \tag{3}$$

where $K()$ is a given kernel function and $\Phi_i \subset \{1 \dots m\}$ denotes the indexes of p nearest neighbours of x_i in U.

The affinity matrix becomes $W = \begin{bmatrix} 0 & Z \\ Z^T & 0 \end{bmatrix} \in \mathbb{R}^{(n+m) \times (n+m)}$. The degree matrix becomes $D =$

$\begin{bmatrix} D_r & 0 \\ 0 & D_c \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$,where D_r is a diagonal matrix whose diagonal elements are row sums of Z and D_c is a diagonal matrix whose diagonal elements are the column sums of Z. Since Z is by definition row normalized, we have $D_r = I_n$, where I_n is the n by n identity matrix. Let $\hat{Z} = ZD_c^{-1/2}$. Thus, we have transformed a unstructured ordinary graph to a bipartite graph.

Although the time complexity of distance computing is reduced, this approach still need to go through the data set multiple times for using K-Means to generate the anchor points. This is still prohibitive for processing large dataset, which dues to the fact that when the data set is too large to hold it all in memory, the transfer time will soon occupy most portion of the total running time.

The Proposed Sequential K-Means

Here, we propose a new approach to construct the graph, which preserves low computational time and has low memory footprint. The core of this approach is the sequential K-Means algorithm.

The proposed Sequential K-Means algorithm (SeqKM) is given in Alg. 1. The basic idea of SeqKM is that we first randomly select a small subset of data and run K-Means++ algorithm on it and then sequentially process the whole data set using stochastic gradient descent (SGD) (Bottou and Bengio 1995). The K-Means++ (Arthur and Vassilvitskii 2007) guarantees the initial centers have relatively good quality, while the SGD further adjusts the centers by looking at the whole data set. The computation of SeqKM is $O(msd + nd)$, where m is the number of clusters, s is the number of samples. This is much lower than ordinary K-Means. More importantly, it only requires scanning data at most twice (one for random sampling and one for SGD optimization). In fact, with proper data storage, sub-sampling can be implemented without scanning the whole dataset, then only one pass is needed.

Algorithm 1 Sequential K-Means (SeqKM)

```
 1: Input: Number of clusters $k$, data set $X \in \mathbb{R}^{n\times d}$, sample size $s$
 2: Output: Cluster labels of each data points, centers $C$ and cluster labels of all anchor points.
 3: v = 0 // Per-center count
 4: M = Randomly select $s$ samples from X;
 5: C = Run K-Means++ algorithm on $M$ and get $k$ centers
 6: for $i = 1$ to $n$ do
 7:     $j = arg\min_j \|x_i - C_j\|^2 \quad \forall j \in 1,\dots,k$
 8:     $v[j] = v[j] + 1$
 9:     $\eta = \frac{1}{v[j]}$
10:     $C_j = (1-\eta)C_j + \eta x_i$
11: end for
```

The SeqKM can find a good balance between the quality and speed in the graph construction step, which is of independent interest. With SeqKM, the graph construction can be easily implemented as a sequential on-disk algorithm, which eliminates the potential bottleneck of introducing high transfer time.

The Proposed Sequential Spectral Clustering

After constructing the normalized bipartite graph matrix $\hat{Z} \in \mathbb{R}^{n\times m}$, we need to compute the left and right singular vectors of $\hat{Z}$ as shown in Eq. (4) in the clustering step. The solution of SC are the smallest eigenvectors G of L. This is equivalent to computing largest singular vectors of $\hat{Z}$, that is:

$$svd(\hat{Z}) = A\Sigma B^T, \tag{4}$$

where $svd(\cdot)$ denotes the Singular Value Decomposition (SVD), $\Sigma = diag(\sigma_1, \dots, \sigma_m) \in \mathbb{R}^{m\times m}$ is a diagonal matrix with the singular values of $\hat{Z}$ on its diagonal, $\sigma_1 \geq \sigma_2 \geq \dots, \geq \sigma_m \geq 0$, $A \in \mathbb{R}^{n\times m}$ is the left singular vectors, $B \in \mathbb{R}^{m\times m}$ is the right singular vectors and $G = [A^T, B^T]^T \in \mathbb{R}^{(n+m)\times m}$. Finally, we can get the cluster labels of each data point by running the K-Means algorithm on G.

The above approach is appealing since it reduces the computational cost of spectral clustering on n data point to close to a linear function of n. However, when the amount of data exceeds the capacity of memory, the performance of this approach will suffer from severe degradation. This is due to the need of swapping data back and forth between the memory and the disk.

Fortunately, Z is a tall thin matrix and we can implement a sequential SVD algorithm by using this fact. Note that the right singular vectors B is also the eigenvectors of matrix $\hat{Z}^T\hat{Z} \in \mathbb{R}^{m\times m}$, which is a simple inner product of matrix Z and can be computed sequentially as $\hat{Z}^T\hat{Z} = \sum_{i=1}^n z_i^T z_i$. After obtaining B, A can be computed by $A = \hat{Z}B\Sigma^{-1}$. Hence, we can compute A sequentially by the following relation:

$$a_i = \hat{z}_i B\Sigma^{-1}, \tag{5}$$

where z_i is the i-th row of Z and a_i is the i-th row of A.

Algorithm 2 Sequential Singular Value Decomposition (SSVD)

```
 1: Input: Data matrix $Z \in \mathbb{R}^{n\times m}$, Number of singular values/vectors $k$
 2: Output: Singular vectors $A \in \mathbb{R}^{n\times k}$, $B \in \mathbb{R}^{m\times k}$, and Singular Values $\Sigma \in \mathbb{R}^{k\times k}$
 3: $S = 0$
 4: for $i = 1$ to $n$ do
 5:     $S = S + z_i^T z_i$ // Sequentially compute $Z^T Z$
 6: end for
 7: $[B\Sigma] = eig(S, k)$ // $eig()$ is eigendecomposition
 8: $\Sigma = \Sigma^{1/2}$
 9: $R = \Sigma^{-1}B$
10: for $i = 1$ to $n$ do
11:     $A_i = z_i R$
12: end for
```

We summarize the Sequential Singular Value Decomposition (SSVD) algorithm in Alg. 2. The basic idea is that we compute the Gram Matrix $Z^T Z$ as a form of sum over data (i.e. $Z^T Z = \sum z_i^T z_i$) and obtain B and then sequentially compute each row of A by projection. In our context, we always have $m \ll n$. Therefore, $Z^T Z$ and B are both relatively small dimensional matrix, which is time and space efficient to compute and store.

With SeqKM and SSVD, we propose our Sequential Spectral Clustering (SeqSC) algorithm in Alg. 3. In SSC, we also perform the graph construction step and normalization step sequentially. At each time step, we only need one small piece (e.g. one row) of data and small amount of intermediate information, the memory usage can be kept in very low level even the data set is large. In fact, the memory usage is independent to the size of the data set.

Algorithm 3 Sequential Spectral Clustering (SeqSC)

```
 1: Input: Data matrix $X \in \mathbb{R}^{n \times d}$, Cluster number $k$, Anchor points number $m$.
 2: Output: Cluster labels of each data points, all anchor points $U$ and cluster labels of all anchor points.
 3: Generate $m$ anchor points $U$ using SeqKM (Alg. 1);
 4: D = 0
 5: for $i = 1$ to $n$ do
 6:     Compute $z_i$ $X_i$ and $U$ using Eq. (3)
 7:     D = D + $z_i$
 8: end for
 9: D = diag(D) // Convert $D$ to diagonal matrix
10: for $i = 1$ to $n$ do
11:     $\hat{z}_i = z_i D^{-1/2}$ // Compute $\hat{Z}$ sequentially
12: end for
13: Compute $[A, \Sigma, B] = SSVD(\hat{Z}, k)$ // Alg. 2
14: Apply SeqKM (Alg. 1) on $A$ to get the cluster labels
```

Computational Complexity and Memory Use. The proposed SeqSC consists three steps: 1) Generating the anchor points U by SeqKM clustering; 2) Constructing bipartite graph Z; 3) Running spectral clustering. Generating the anchor points takes $O(ksd + nmd)$ time. The construction of an bipartite graph takes $O(nmd)$ time. In the third step, the singular decomposition takes $O(nm^2)$ time. The total running time will be $O(nm^2)$. (In fact, this complexity could be further reduced by sub-selection technique (Li et al. 2014; Li, Chen, and Huang 2014). We will leave this improvement to the future work.) For memory usage, the first call of SeqKM in Algorithm 3 takes $O((s + m)d) \approx O(sd)$ space, while the second call takes $O((s + k)m) \approx O(sm)$. The graph construction and graph normalization take $O(m^2 + md)$ space, while the SSVD step takes $O(mk)$ space. Therefore, the overall space complexity is $O((m + d)m))$. It is obvious that the time complexity of the proposed algorithm is much lower than the quadratic time of original spectral clustering and comparable to other SC algorithms based on bipartite graph, e.g. (Chen and Cai 2011), (Liu et al. 2013a). Furthermore, the space complexity of existing SC algorithms is at least $O(n * \max(m, d))$. For very large n, this will cause performance problem because of disk swapping. In contrast, the space complexity of the proposed method is irrelevant to n, which can avoid the swapping problem. These advantages will be validated in the experimental section.

Experiments

In this section, we evaluate the proposed Sequential Spectral Clustering (SeqSC) approach on three benchmarks datasets. All our experiments are conducted on a desktop computer with a 3.0GHz Intel Pentium 4 CPU and 1GB RAM, MatLab 7.14 (32bit).

Datasets and Experimental Environment

We conduct experiments on three data sets MNIST, CovType and MNIST8m. They all have ground truth for measuring clustering quality.

MNIST[1]. This dataset consists of 70,000 images of handwritten digits from 0 to 9. We use 784 dimension original pixel values to represent each image.

CovType. This dataset consists of 581,012 for predicting the forest cover type from cartographic variables. Each sample belongs to one of seven types (classes).

MNIST8m[2]. This data set consists of 8,100,000 images of handwritten digits from 0 to 9 (Loosli, Canu, and Bottou 2007). The feature is the same as MNIST dataset: 784 dimension original pixel values. We randomly select a subset of 500,000 samples.

Experimental Protocols

We conduct three parts of experiments to evaluate the effectiveness and efficiency of our proposed algorithms. We compare the SeqSC to several state-of-the-art SC algorithms. For all compared algorithms, we download code from the authors' website and follow the parameter settings in their papers. All the code are written in MatLab. And we use the code from (Chen et al. 2011) to sequentially generate k-NN ($k = 50$) so that the out-of-memory exception will not be triggered during the graph construction phase. For parameters of our algorithm, we fix $m = 200$ and construct 5-NN anchor graph. For the first part, we compare the following algorithms:

- **SC** is the spectral clustering on normalized Laplacian (Ng et al. 2002). We use the implementation that is available online (Chen et al. 2011)[3].
- **KASP** is proposed in (Yan, Huang, and Jordan 2009) and is an approximate spectral clustering algorithm based on K-Means. We implement this algorithm using MatLab for fair comparison.
- **Nyström** (Fowlkes et al. 2004) uses Nyström transform to approximate the eigenfunction problem. We use the implementation that is available online (Chen et al. 2011)[4].

[1]http://yann.lecun.com/exdb/mnist/

[2]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html

[3]http://alumni.cs.ucsb.edu/~wychen/sc.html

[4]http://alumni.cs.ucsb.edu/~wychen/sc.html

- **LSC** (Chen and Cai 2011) is a Landmark-based spectral clustering algorithm. We download the MatLab code from the authors' website[5].
- **ESCG** (Liu et al. 2013a) uses shortest path algorithm to generate bipartite graph. We download the Matlab code from the authors' website[6].
- **SeqSC** is the proposed algorithm described in Alg. 3.

The second part of experiments aims to evaluate the performance of proposed SeqKM algorithm as a component of SeqSC. That is we replace SeqKM with other alternatives to evaluate the effects on the clustering accuracy and the running time. **1) SeqSC(RS)** is the variation of the proposed SeqSC that uses random sampling to generating anchor points instead of SeqKM. **2) SeqSC(KK)** is the variation of the proposed SeqSC that uses only the results of K-Means++ on sub-sampling data. That means we only use traditional K-Means on subsampling data to compute the cluster centers and assign each data point to the closest centers (We call it **SubKM** below). **3) SeqSC(SK)** is the variation of the proposed SeqSC that uses SeqKM to generate anchor points and SubKM to compute the cluster labels. **4) SeqSC(KS)** is the variation of the proposed SeqSC that uses SubKM to generate anchor points and SeqKM to compute the cluster labels. **5) SeqSC(SS)** is the full version of the proposed SeqSC algorithm.

For data access, we split data into blocks so that we can check the performance of transferring data. For all experiments, we fix each block to contain 5000 data points. Note that the number 5000 here is not critical as long as it contains data larger than the hard disk transfer block size and less than the memory capacity, which is a wide range. For all batch algorithms, i.e. all of the compared algorithms, we load all of the testing data into memory before the algorithm is run, while for the SeqSC and all its variations, we sequentially load each block when we need to compute it. We measure the clustering quality by the **accuracy** (Chen and Cai 2011), which compares the clustering labels generated by the algorithms to the labels given by the data set. We also report the **running time** of each algorithm using wall-clock time. We evaluate the **accuracy** (Chen and Cai 2011) and **running time** of all the algorithms.

Clustering Quality

The first part of our experiments is to compare our algorithm with other state-of-the-art algorithms. We conduct this part of experiments on two datasets: MNIST and CovType. The results are shown in Figure 1 to Figure 3b. Since the running time of different algorithms may vary greatly, they are shown in log-scale.

Figure 1 and Figure 2 show the compared results of all the algorithms on MNIST dataset in terms of accuracy and running time respectively. In Figure 1, we can observe that LSC and SC are roughly the best among all the baseline algorithms. These results are consistent with previous works

[5]http://www.cad.zju.edu.cn/home/dengcai/Data/ReproduceExp.html

[6]http://jialu.cs.illinois.edu/

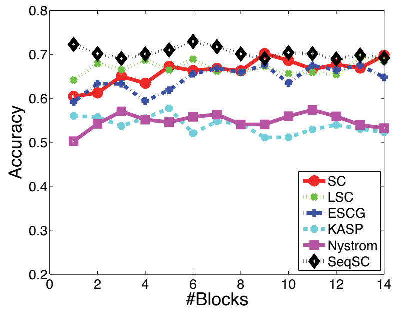


Figure 1: Clustering accuracy comparison on MNIST dataset. Accuracy vs #Blocks.

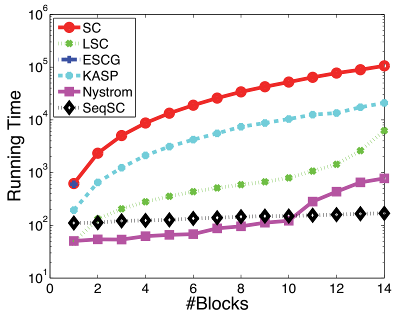
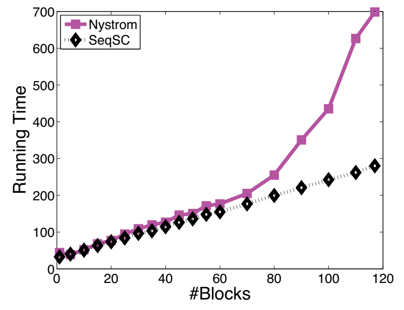


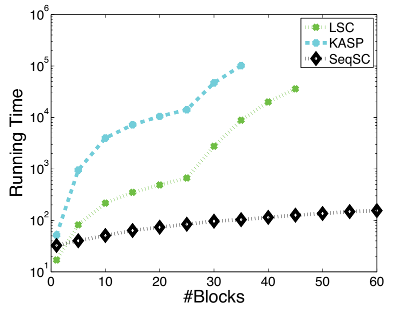
Figure 2: Running time comparison on MNIST dataset. Running Time vs. #Blocks.

(Chen and Cai 2011; Liu et al. 2013a). The proposed SeqSC has consistently outperformed all the competitors in various data size from 1 block ($n = 5000$) to 14 blocks ($n = 70000$). The comparison results of the running time are shown in Fig. 2. We can observe that the proposed method is the fastest of all. We can observe that SC and ESCG take almost the same amount of time. This is due to the fact that most of the running time is used to construct the k-NN graph. And since the resulted graph is very sparse, the time for the eigendecomposition in SC is relatively short (Liu et al. 2013a). The following ones are KASP and LSC, whose running time grows fast as the data size grows. The running time of the Nyström algorithm is very close to the proposed SeqSC, which are among the fastest algorithms of all. However, as the size of data grows, the running time Nyström increase more rapidly. Note that although the running time of the Nyström algorithm is comparable to the proposed algorithm, their accuracy is not, which has already been shown in Figure 1. This is due to the fact that although running time is reduced when Nyström uses random subset of data points to approximate the large-scale singular value decomposition, the sampled data points are not effective in preserving the neighboring information of the data set. This information loss results in the performance degradation of the Nyström algorithm. All these results demonstrate the benefit of the proposed method.

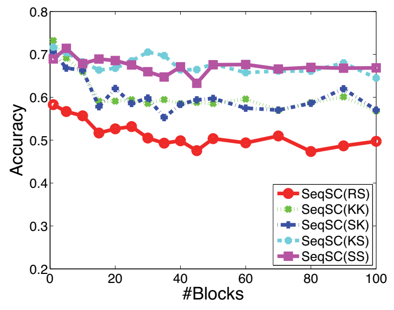
Figure 3a and 3b show the results on CovType dataset,



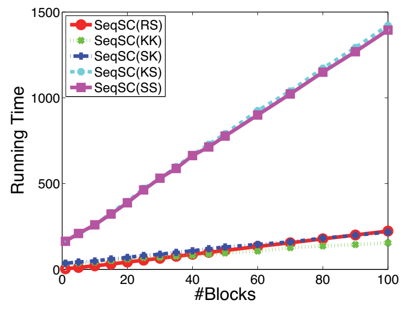
(a) Time vs. #Blocks @ CovType.



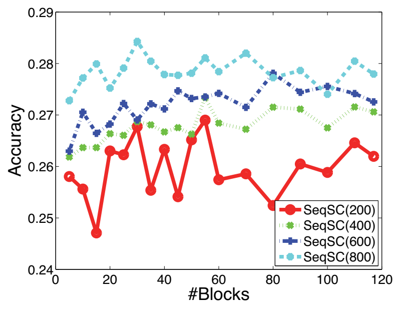
(b) Time vs. #Blocks @ CovType.



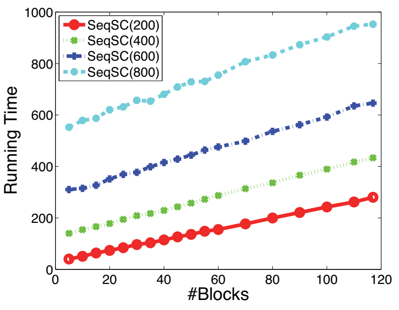
(c) Acc vs. #Blocks @ MINIST8m.



(d) Time vs. #Blocks @ MINIST8m.



(e) Time vs. #Blocks @ CovType.



(f) Time vs. #Blocks @ CovType.

Figure 3: Clustering Accuracy (Acc) and running time. . (a) Compared with Nyström (b) Compared with LSC and KASP. (c)-(d) Compare different K-Means approaches; (e)-(f) Compare different number of anchor points.

which is much larger than the MNIST dataset. We omit the results of SC and ESCG because they are too slow to be run on dataset of this scale. In these figures, we can observe that the proposed SeqSC still outperforms the baseline algorithms. The LSC and the KASP only produce results for part of the data due to the out-of-memory error. Although the Nyström algorithm can produce results for all the data, its accuracy is lower while its running time grows faster than the proposed algorithm (Figure 3a and 3b). These results again validate the performance and scalability of the proposed algorithm.

Influence of the SeqKM

We further evaluate the performance of different variations of the proposed algorithm on MNIST8m dataset. The results are shown in Figure 3c. We can observe several things from Figure 3c. Firstly, the quality of the generated anchor points is important. The algorithms that are based on K-Means generated anchor points are much better than those based on random generated anchor points. Secondly, SeqKM also outperforms the SubKM in the stage of generating cluster labels. This is due to the fact that SeqKM finds a good balance between the clustering quality and performance of the proposed algorithm. The SeqKM algorithm tries to find better initialization from local data and use one pass on the whole data set to adjust the final results, which avoid being trapped in the local solution.

Figure 3d shows the running time comparison of different variations of SeqSC. This result shows that under the framework of our sequential clustering, the running time of every version of SeqSC grows linearly with respect to the amount of data. This result again validates the scalability of the proposed algorithms.

Parameter Selection

The third part of our experiment evaluates the performance of the proposed algorithm under different parameters. The most important parameter is the number of anchor points p, which controls the quality of the bipartite graph. We conduct experiments on CovType dataset of SeqSC with p varies from 200 to 800. We can see from Figure 3e that the accuracy increases which p increases. Note that although the running time increases along with p, it is still linear with respect to n. The optimal balance between running time and accuracy depends on computational resources at hand. Still and all, promising results have been achieved by the proposed algorithm even with moderate choice of parameters.

Conclusion

In this paper, we proposed a sequential spectral clustering algorithm, named by SeqSC, for coping with large-scale clustering. The SeqSC algorithm counts on two key components: the proposed Sequential K-Means (SeqKM) and Sequential Singular Value Decomposition (SSVD) algorithms. The benefits of the proposed SeqSC are three-fold: **(1)** The proposed SeqKM algorithm extends the previous online K-Means algorithm by adopting a better initialization, which can discover high-quality cluster centers in one pass; **(2)** the proposed SeqKM algorithm enables the graph construction to be executed in a sequential manner; **(3)** by employing SSVD, clustering can be performed sequentially and the entire time complexity is nearly linear in the number of input data points. Such benefits make our SeqSC algorithm accurate and scalable enough to tackle massive datasets. The experimental results on three public datasets demonstrated that the proposed SeqSC algorithm outperforms the state-of-the-arts in large-scale clustering with up to 1000 times acceleration.

References

Arthur, D., and Vassilvitskii, S. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 1027–1035. Society for Industrial and Applied Mathematics.

Bottou, L., and Bengio, Y. 1995. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7*. Citeseer.

Chang, X.; Nie, F.; Ma, Z.; Yang, Y.; and Zhou, X. 2015. A convex formulation for spectral shrunk clustering. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Chen, X., and Cai, D. 2011. Large scale spectral clustering with landmark-based representation. In *AAAI*.

Chen, B.; Gao, B.; Liu, T.-Y.; Chen, Y.-F.; and Ma, W.-Y. 2006. Fast spectral clustering of data using sequential matrix compression. In *Machine Learning: ECML 2006*. Springer. 590–597.

Chen, W.-Y.; Song, Y.; Bai, H.; Lin, C.-J.; and Chang, E. Y. 2011. Parallel spectral clustering in distributed systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33(3):568–586.

Fowlkes, C.; Belongie, S.; Chung, F.; and Malik, J. 2004. Spectral grouping using the nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26(2):214–225.

Khoa, N. L. D., and Chawla, S. 2012. Large scale spectral clustering using resistance distance and spielman-teng solvers. In *Discovery Science*, 7–21. Springer.

Li, Y.; Chen, C.; Liu, W.; and Huang, J. 2014. Subselective quantization for large-scale image search. *Proc. AAAI Conference on Artificial Intelligence (AAAI)*.

Li, Y.; Nie, F.; Huang, H.; and Huang, J. 2015. Large-scale multi-view spectral clustering via bipartite graph. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Li, Y.; Chen, C.; and Huang, J. 2014. Transformation-invariant collaborative sub-representation. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, 3738–3743. IEEE.

Liu, T.-Y.; Yang, H.-Y.; Zheng, X.; Qin, T.; and Ma, W.-Y. 2007. Fast large-scale spectral clustering by sequential shrinkage optimization. In *Proceedings of the 29th European conference on IR research*, 319–330. Springer-Verlag.

Liu, J.; Wang, C.; Danilevsky, M.; and Han, J. 2013a. Large-scale spectral clustering on graphs. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 1486–1492. AAAI Press.

Liu, J.; Wang, C.; Gao, J.; and Han, J. 2013b. Multi-view clustering via joint nonnegative matrix factorization. In *Proc. of SDM*, volume 13, 252–260. SIAM.

Liu, W.; He, J.; and Chang, S.-F. 2010. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 679–686.

Liu, W.; Wang, J.; and Chang, S.-F. 2012. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE* 100(9):2624–2638.

Loosli, G.; Canu, S.; and Bottou, L. 2007. Training invariant support vector machines using selective sampling. In Bottou, L.; Chapelle, O.; DeCoste, D.; and Weston, J., eds., *Large Scale Kernel Machines*. Cambridge, MA.: MIT Press. 301–320.

Miao, G.; Song, Y.; Zhang, D.; and Bai, H. 2008. Parallel spectral clustering algorithm for large-scale community data mining. In *The 17th WWW workshop on social web search and mining (SWSM)*.

Ng, A. Y.; Jordan, M. I.; Weiss, Y.; et al. 2002. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 2:849–856.

Sakai, T., and Imiya, A. 2009. Fast spectral clustering with random projection and sampling. In *Machine Learning and Data Mining in Pattern Recognition*. Springer. 372–384.

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22(8):888–905.

Shinnou, H., and Sasaki, M. 2008. Spectral clustering for a large data set by reducing the similarity matrix size. In *LREC*.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and computing* 17(4):395–416.

Yan, D.; Huang, L.; and Jordan, M. I. 2009. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 907–916. ACM.

Yu, S. X., and Shi, J. 2003. Multiclass spectral clustering. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 313–319. IEEE.

Zelnik-Manor, L., and Perona, P. 2004. Self-tuning spectral clustering. In *Advances in neural information processing systems*, 1601–1608.

Zhang, K., and Kwok, J. T. 2010. Clustered nyström method for large scale manifold learning and dimension reduction. *Neural Networks, IEEE Transactions on* 21(10):1576–1587.