Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

**Compressed *K*-Means for Large-Scale Clustering**

**Xiaobo Shen,**t‡ **Weiwei Liu,**‡ **Ivor Tsang,**‡ **Fumin Shen,**b **Quan-Sen Sun**t\*

t School of Computer and Engineering, Nanjing University of Science and Technology

‡ Centre for Artiﬁcial Intelligence, University of Technology Sydney

b School of Computer Science and Engineering, University of Electronic Science and Technology of China {njust.shenxiaobo, liuweiwei863, fumin.shen}@gmail.com, ivor.tsang@uts.edu.au, sunquansen@njust.edu.cn

**Abstract**

Large-scale clustering has been widely used in many ap- plications, and has received much attention. Most existing clustering methods suffer from both expensive computation and memory costs when applied to large-scale datasets. In this paper, we propose a novel clustering method, dubbed compressed k-means (CKM), for fast large-scale clustering. Speciﬁcally, high-dimensional data are compressed into short binary codes, which are well suited for fast clustering. CKM enjoys two key beneﬁts: 1) storage can be signiﬁcantly re- duced by representing data points as binary codes; 2) distance computation is very efﬁcient using Hamming metric between binary codes. We propose to jointly learn binary codes and clusters within one framework. Extensive experimental re- sults on four large-scale datasets, including two million-scale datasets demonstrate that CKM outperforms the state-of-the- art large-scale clustering methods in terms of both computa- tion and memory cost, while achieving comparable clustering accuracy.

**Introduction**

Clustering is a fundamental technique in machine learning and pattern recognition. The aim of clustering is to parti- tion a data set into different groups with similar data points being assigned into one group. Until now many cluster- ing algorithms (Hartigan and Wong 1979; Ng et al. 2001; Li et al. 2009; Wang et al. 2011b; Ding et al. 2015) have been proposed, including the widely used k-means clustering (Hartigan and Wong 1979; Arthur and Vassilvitskii 2007; Ding et al. 2015) and spectral clustering (Shi and Malik 2000; Ng et al. 2001).

There has been a dramatic growth in the volume of data with the advent of Internet in the recent decades. For instance, Flickr has more than 5 billion images avail- able, YouTube receives more than 100 hours of videos up- loaded per minute. Conventional clustering methods such as spectral clustering cannot be directly applied to large- scale datasets due to their high computation cost. Recently, increased attention has been paid to large-scale clustering (Chen and Cai 2011; Chen et al. 2011; Li et al. 2015;

\* Corresponding author.

Copyright oc 2017, Association for the Advancement of Artiﬁcial Intelligence (www.aaai.org). All rights reserved.

Gong et al. 2015; Zhang and Lu 2016), which aims to de- velop clustering methods with highscalability. For example, several variants, e.g., Nystrm (Chen et al. 2011), large-scale clustering (LSC) (Chen and Cai 2011), large-scale multi- view spectral clustering (Li et al. 2015) have been proposed to reduce the high computation of spectral clustering.

In contrast, k-means clustering has been more often ap- plied to large-scale clustering because of its simplicity and general applicability. Two steps are employed in each it- eration: updating the cluster centers, and updating the as- signments of each point. The computation cost of k-means in each iteration is O(nkd), where n, k, d are the size of the dataset, the number of clusters, and the dimensional- ity, respectively. For such large values, even a single iter- ation is very slow. For example, we show in the experi- ment on MINIST8M dataset where n = 8.1M, d = 784, k = 1000, k-means takes around 6500s to update in each iteration. In addition, 50.80G is needed to store this dataset. Basically, it is very challenging to directly apply k- means over this scale of dataset in a single machine. Sev- eral variants (Elkan 2003; Arthur and Vassilvitskii 2007; Hamerly 2010; Drake and Hamerly 2012; Ding et al. 2015; Bachem et al. 2016) of k-means have been proposed to im- prove the clustering efﬁciency. They can decrease the num- ber of iterations, but the running time of each iteration and memory usage are unchanged. Therefore, two main chal- lenges remain to be solved in large-scale clustering: 1) how to reduce the storage of huge data, and 2) how to reduce the computational cost of the clustering methods.

Binary code learning (Wang et al. 2016) has gained in- creased interests in many large-scale applications. The ba- sic idea of binary code learning is to encode the origi- nal high-dimensional data into a set of short binary codes with similarity preservation. The advantage of binary cod- ing is that it can perform an effective search in Hamming space at very low cost of both storage and computation. Many binary code learning methods (Gionis et al. 1999; Gong and Lazebnik 2011; Norouzi, Fleet, and Salakhutdi- nov 2012; Zhou et al. 2016; Song, Liu, and Meyer 2016; Shen et al. 2017) have been developed, all of which are pro- posed to facilitate large-scale retrieval and classiﬁcation, in- stead of clustering. A pioneering work (Gong et al. 2015) to perform clustering over binary codes was recently pro- posed. This is a naive two-step approach, which generates

binary codes and performs clustering separately. Apparently, the binary codes may not be optimal for clustering in this two-step approach, and we show in our experiments that it performs poorly on some datasets. To the best of our knowl- edge, learning binary codes for clustering has been less well studied and remains a very challenging area.

Inspired by the great success of binary code learning, we propose a novel compressed k-means (CKM) for large- scale clustering. CKM aims to simultaneously learn binary codes and clusters. The advantages of CKM over conven- tional clustering methods lie in the low cost of computation and storage. Taking the MINIST8M dataset as an example, CKM reduces the storage around 392 times from 50.80G to 129.60M; meanwhile, the running time of CKM in one iteration is 25s, which is nearly 260 times faster than con- ventional k-means. The main contributions of this work are summarized below:

• We propose a novel binary coding based clustering method, named compressed k-means (CKM),which com- presses high-dimensional data into short binary codes. CKM can be applied to large-scale clustering at low com- putational and storage costs.

• The proposed CKM is formulated to jointly perform bi- nary coding function learning and clustering, such that the learned binary codes are optimal for clustering. Inspired by the recent advances in structural prediction (Yu and Joachims 2009; Norouzi, Fleet, and Salakhutdinov 2012), an upper bound of the empirical loss of the optimization algorithm is presented, for which an efﬁcient optimization algorithm is devised.

• Extensive experiments on four large-scale datasets, demonstrate that the proposed CKM is faster and has lower memory usage than the state-of-the-art large-scale clustering methods.

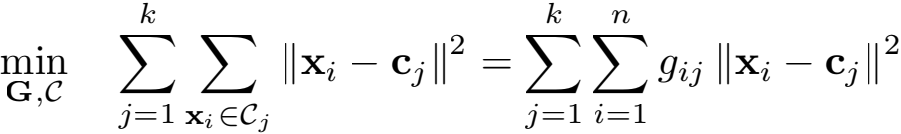
**Notations and Background**

Given a dataset X = {**x**i ∈ Rd  **X** = [**x**1, . . . , **x**n], and the goal of clustering is to group the data

points {**x**1, . . . , **x**n } into k clusters {Cj }=1 , such that simi-

lar data points can be grouped together. We use the partition matrix **G** = [**g**1, . . . , **g**n] ∈ {0, 1}k ×n to represent the clus- tering results. Let gij = 1 if **x**i belongs to cluster Cj and gij = 0 otherwise; we call **G** the cluster indicator matrix because each column,i.e., **g**i (1 ≤ i ≤ n), has one and only one element equal to 1 to indicate the cluster membership, while the remaining elements are 0. We denote the set of such indicator matrices as Ψ .

The k-means method is the most popular clustering method because of its simplicity. It has been identiﬁed as one of the top 10 algorithms in data mining (Wu et al. 2008). Formally, k-means aims to minimize the following objective function:



s.t. **G** ∈ Ψk ×n (1)

where Ⅱ · Ⅱ denotes l2 norm of a vector, and **c**j is the j-th centroid of the dataset. Because **G** ∈ Ψ is a cluster indica- tor matrix,k-means is a combinatorial optimization problem that is generally difﬁcult to resolve.

A simple yet popular algorithm for ﬁnding a local opti- mum of the k-means problem starts with a random set of k centers and is as follows: each data point is repeatedly assigned to its nearest center, and the centers are recom- puted given the point assignments. This local search, called *Lloyd’s* iteration, continues until a stable set of centers is obtained. In each iteration,k-means requires the time com- plexity of O(nkd). For a large-scale dataset in which both n and k are large, k-means will be computationally expen- sive even with medium-length d. It is also very challenging to store the whole dataset and cluster centers in a single ma- chine. Generally, directly applying k-means on a large-scale dataset is inefﬁcient.

A challenging problem naturally raises how to efﬁciently perform k-means on large-scale datasets. In the following sections, we introduce the binary coding technique to ad- dress this issue.

**Compressed** K**-means**

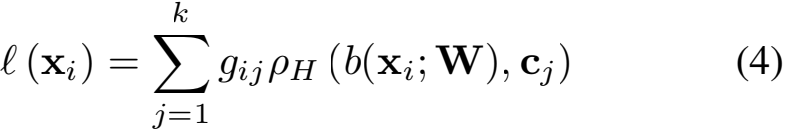
**Problem Formulation**

Given the data matrix **X** ∈ Rd×n , we aim to learn a mapping b(**x**) that projects d-dimensional real-valued input **x** ∈ Rd onto a r-dimensional binary code **h** ∈ H ≡ {−1, 1}r , where k-means clustering can beefﬁciently performed. The map- ping, referred to as binary coding function, is deﬁned as:

b (**x**; **W**) = sign (f (**x**, **W**)) (2) where sign(·) is the element-wise sign function, and f (**x**, **W**) : Rd → Rr is a real-valued transformation, where **W** ∈ Rd×r. A variety of mathematical forms off can be used for domain speciﬁc practical applications. In this work, we consider a linear transformation f (**x**) = **W**T**x** for its simplicity.

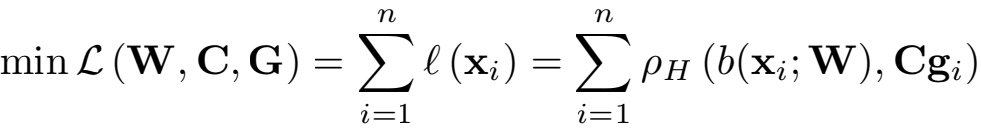
The distance between binary codes **h**, **e** ∈ H can bede- noted as:

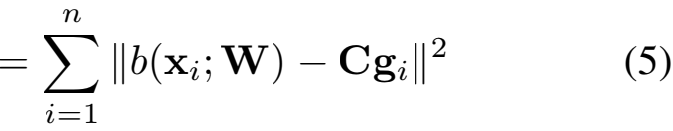
ρH (**h**, **e**) = Ⅱ**h** − **e**Ⅱ2 (3) Fork-means clustering in the Hamming space, we deﬁne the loss function of the i-th data point:



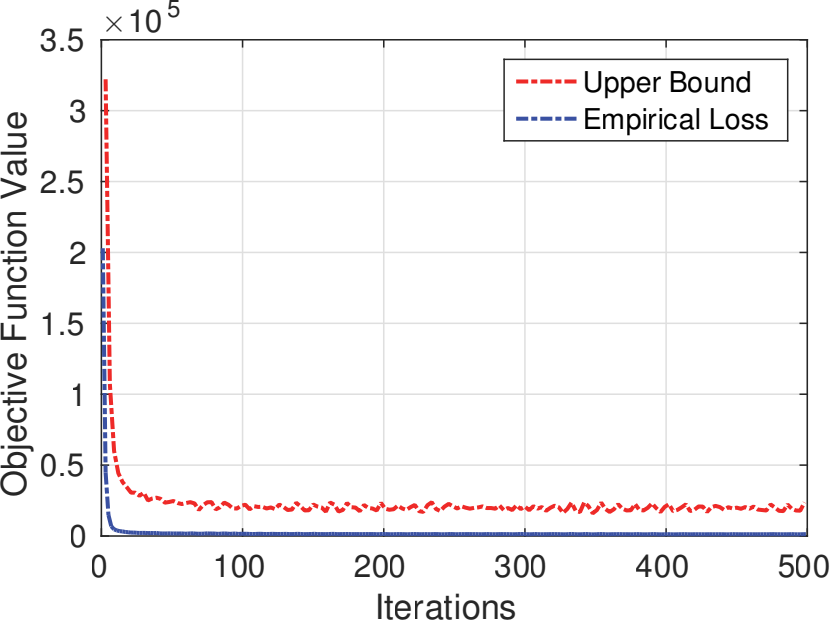
= ρH (b(**x**i ; **W**), **Cg**i )

where **C** = [**c**1, . . . , **c**k ] ∈ {−1, 1}r×k. Thus the objective function of the proposed compressed k-means (CKM) is de- ﬁned as:





s.t. Ⅱ**w**j Ⅱ ≤ ν, ∀j ∈ {1, . . . , k} , and **C** ∈ {−1, 1}r×k , **G** ∈ Ψk ×n

where ν ∈ R+ is a positive regularization parameter that is used to constrain the scale of **W**. This is because the scale of **W** does not affect the objective function value. We impose the binary constraints on the cluster centers **C**. In constrst to the conventional k-means, the data points and cluster cen- ters are both hashed into binary codes, which enables us to perform fast distance calculation in clustering.

Direct global optimization of L is challenging because 1) the objective function is highly non-convex; 2) b(**x**; **W**) is a discrete mapping. In the following section, we will present the technique to address these difﬁculties.

**Upper Bound on Empirical Loss**

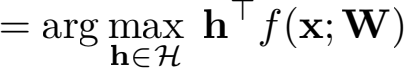
Inspired by latent structural SVMs (Yu and Joachims 2009; Liu and Tsang 2015), we develop the optimization technique to optimize an upper bound of L. We ﬁrst re-express the binary coding function as a form of structured prediction (Norouzi, Fleet, and Salakhutdinov 2012):

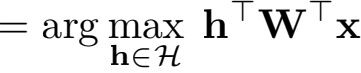
Figure 1: The upper bound and empirical loss with respect to different iterations on RCV1 dataset.

With simple matrix transformations, the solution to **e**i can easily be obtained:

**e**i = sign (−2**Cg**i + α**W**T**x**i ) (10) **Cluster Learning.** We next optimize the sub-objective func- tion of clustering in the Hamming space. The sub-objective function with respect to **C**, **G** is as follows:

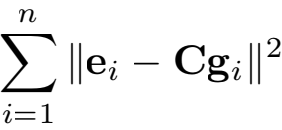
b (**x**; **W**) = sign (f (**x**; **W**)) (6)





Here, (6) holds because that the optimal code should be +1 for the positive entries of **W**T**x**, and -1 otherwise.

Based on the structure prediction form of binary coding function, we present a theorem on the upper bound of the loss function of the i-th data point.

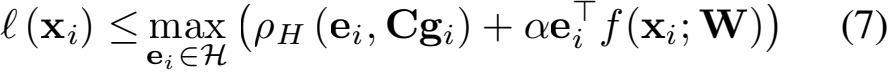


**C** ∈ {−1, 1}r×k , **G** ∈ Ψk ×n

,i s.t.

(11)

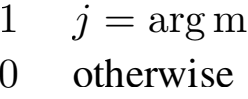
**Theorem 1.** *For arbitrary* α > 0 *, the loss function of the* i*-th data point,i.e.,* l(**x**i )*, is upper bounded by:*

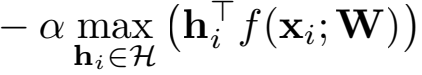


The optimization of (11) is similar to conventional k-means. To obtain the locally optimal {**C**, **G**}, it is necessary to it- eratively update one variable while ﬁxing the other variable until convergence. Below we show that, in each iteration, {**C**, **G**} can be solved via the following theorem.

**Theorem 2.** *In each iteration,* {**c**j, **g**i } *that minimizes the optimization problem in* (11) *is given by:*

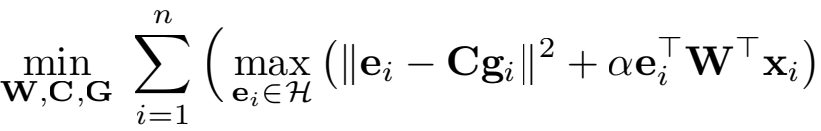
**c**j =sign (Σgij =1 **e**i ) (12)

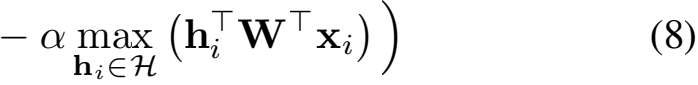
gij = {inl Ⅱ**e**i − **c**l Ⅱ (13)



*Proof.* This upper bound is easily derived via structural SVM. I

Based on Theorem 1, we can obtain the following *surro- gate* objective function:





*where* i = 1, . . . ,n*,* j = 1, . . . ,k*.*

*Proof.* The proof can be easily obtained similar to the con-

s.t. Ⅱ**w**j Ⅱ ≤ ν, ∀j ∈ {1, . . . , k} , and **C** ∈ {−1, 1}r×k , **G** ∈ Ψk ×n

ventional k-means.

**Optimization**

**Binary Coding Function Learning.** Optimizing the ob- jective function with respect to **W** is difﬁcult because it is a convex-concave problem. In this work, inspired by (Norouzi, Fleet, and Salakhutdinov 2012), we employ stochastic gradient descent (SGD) to update **W**. In each iter- ation, we randomly sample a data point,i.e., **x**, and then take a step in the direction that decreases the objective function value. The updating rule of **W** can be represented as

**W** = **W** − η(**e** − **h**)T (14)

**Loss-augmented Inference.** To evaluate and use the surro- gate objective in (8) for optimization, we must solve a *loss- augmented inference* problem to ﬁnd the binary code that maximizes the sum of the score and loss term:

x Ⅱ**e**i − **Cg**i Ⅱ2 + α**eW**T**x**i (9)

s.t. **e**i ∈ {−1, 1}r × 1

2529

**Algorithm 1** Compressed k-means

**Input:** Training set **X** ∈ Rd×n; code length r; cluster num-

berk; parameters α, ν . **Output: W**, **C**, **G**.

1: Initialize **W** via Locality Sensitive Hashing (LSH); 2: Initialize **B** = sign(**W**T**X**);

3: Initialize **C** by randomly selecting k binary codes; 4: **repeat**

5: Update **e**i via (10), i = 1, . . . , n;

6: Iteratively update **C**, **G** via (12), (13); 7: Update **W** using (14) on a small batch;

8: Project **W** back to the feasible region via (15); 9: **until** *convergence*

where η is the learning rate, which is set as 0.001 in this work, **h**, **e** are obtained by the loss inference (6), (10), re- spectively. As there is a norm constraint on **W**, we need to project **W** back to the feasible region, thus we perform the following operation

**w**i = min {1, √ν /Ⅱ**w**i Ⅱ }**w**i (15)

where i = 1, . . . ,r.

In our implementation, we propose a two-stage scheme. We start the optimization algorithm by initializing **W** as a random Gaussian matrix by Locality Sensitive Hashing (LSH) (Gionis et al. 1999). In the ﬁrst stage, we jointly learn the binary coding function and clustering on a ran- domly selected subset with n/ = βn data points, where β is the proportion of the selected data points. In the second stage, we perform clustering on the binary codes of the entire dataset. Here the subset selection is used to speed up the bi- nary function learning. We will show in the experiment that it is enough to learn a good binary function on a small subset for large-scale clustering. In addition, we use mini-batches rather than single data point to compute the gradient, and the batch size is set as 128. The detailed optimization procedure of CKM is described in Algorithm 1.

The theoretical convergence of this update rule has been explored (Hazan, Keshet, and McAllester 2010). In this work, we empirically verify that the update rule lowers both the upper bound and the empirical loss, and converges to a local minimum. Fig. 1 shows the curves of the upper bound and the empirical loss, from where we can clearly see that both converge within a few iterations.

**Computational Complexity and Memory Usage**

The computational complexity of the proposed CKM con- sists of the following parts. Loss-augmented inference re- quires O(dr) for updating each data point. Updating binary coding function requires O(drp), where p is the mini-batch size. The most time-consuming part lies on cluster learn- ing, which takes O(nk) Hamming distance calculation for r-bit codes. In constrast, k-means takes O(nk) Euclidean distance calculation for d-dimensional real-valued vectors. Thus, CKM is more efﬁcient thank-means, especially when r 冬 d.

Table 1: Statistics of four large-scale datasets.

|  |  |  |  |
| --- | --- | --- | --- |
| Datasets | #Sample | #Feature | #Classes |
| RCV1 | 193844 | 1979 | 103 |
| CovType | 581012 | 54 | 7 |
| ILSVRC2012 | 1331167 | 4096 | 1000 |
| MNIST8M | 8100000 | 784 | 10 |

To calculate memory usage,CKM needs to store the trans- formation matrix **W**, which counts for the storage of O(rd) real-valued numbers. The data points and cluster centroids are stored by CKM at the cost of O((n + k)r) bits. k- means requires the storage of O((n + k)d) real-valued num- bers. Therefore, CKM has much lower memory cost than k-means.

**Experiments**

In this section, we evaluate the proposed clustering method on four large-scale datasets. All the computations reported in this study are performed on a Red Hat Enterprise 64-Bit Linux workstation with 18-core Intel Xeon CPU E5-2680 2.80 GHz and 256 GB memory.

**Datasets**

We conduct experiments on four large-scale datasets, whose statistics are summarized in Table 1.

• **RCV1**1 : a subset (Chen et al. 2011) of an archive of 804414 manually categorized newswire stories from Reuters Ltd. It has 193844 documents in 103 categories. Following previous studies (Wang et al. 2011a), we re- move the keywords (features) appearing less than 100 times in the corpus, which results in 1979 (out of 47236) keywords in our experiment.

• **CovType**2 : consists of 581012 instances for predicting forest cover type from cartographic variables. Each sam- ple belongs to one of seven types (classes).

• **ILSVRC2012**3 : a subset of ImageNet (Deng et al. 2009). It contains 1000 object categories and more than 1.2 mil- lion images. As in (Lin, Shen, and van den Hengel 2015), we use the 4096-dimensional features extracted by the convolution neural networks (CNN) model (Krizhevsky, Sutskever, and Hinton 2012) to represent the images.

• **MNIST8M**4 : consists of around 8.1 million images of handwritten digits from 0 to 9. The feature is the same as MNIST dataset: 784-dimensional original pixel values.

**Comparison Methods**

To demonstrate the effectiveness of the proposed CKM, we compare it with ﬁve state-of-the-art large-scale clustering methods, consisting of two k-means methods, two spectral clustering methods, and a naive two-step clustering method

1<http://alumni.cs.ucsb.edu/>~wychen/sc.html

2<https://archive.ics.uci.edu/ml/>

3<http://www.image-net.org/challenges/LSVRC/2012/>

4<http://www.csie.ntu.edu.tw/>~cjlin/libsvmtools/datasets/

Table 2: Running time (in seconds) of clustering on four large-scale datasets.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset | k-means | | k-means++ | | Nystrm | | LSC-K | | LSH+bk-means | | CKM | |
| Time | Speedup | Time | Speedup | Time | Speedup | Time | Speedup | Time | Speedup | Time | Speedup |
| RCV1 | 191s | 1 × | 254s | 0.75 × | 61s | 3.13 × | 206s | 0.93 × | 4s | 47.75 × | 16s | 11.94× |
| CovType | 17s | 1 × | 11s | 1.55 × | 39s | 0.44× | 65s | 0.26× | 1s | 17.00× | 3s | 5.67 × |
| ILSVRC2012 | 8523s | 1 × | 18469s | 0.46× | 2626s | 3.25 × | 22173s | 0.38 × | 89s | 95.76× | 250s | 34.09× |
| MINIST8M | 9718s | 1 × | 2578s | 3.77 × | 1418s | 6.85 × | 5107s | 1.90× | 101s | 96.22× | 248s | 39.19× |

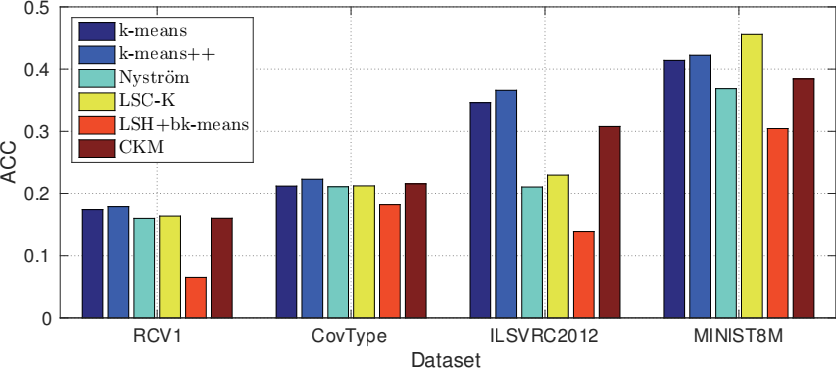
Table 3: Memory usage of k-means and CKM on four large- scale datasets. ‘Mem.’ denotes memory usage. ‘Red.’ de- notes the times of memory reduction compared to k-means.

Figure 2: Clustering accuracy (ACC) on four large-scale data sets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | k-means | | CKM | |
| Mem. | Red. | Mem. | Red. |
| RCV1 | 3.07GB | 1 × | 3.10MB | 988 × |
| CovType | 0.25GB | 1 × | 2.32MB | 432× |
| ILSVRC2012 | 43.62GB | 1 × | 21.30MB | 2048 × |
| MINIST8M | 50.80GB | 1 × | 129.60MB | 392× |

**Evaluation Metric**

We evaluate clustering quality by Accuracy (Chen and Cai 2011). Given the data point **x**i , letoi and si be its resultant clustering label and ground-truth label, respectively. The ac-

using LSH plus bk-means (Gong et al. 2015). The details of these methods are given below.

• k**-means**: is the conventional k-means method based on Euclidean distance. It can be seen as a baseline method.

• k**-means++** (Arthur and Vassilvitskii 2007): a variant of k-means, which provides a good initialization that is prov- ably close to the optimal solution.

• **Nystr****m** (Chen et al. 2011): a parallel large-scale spec- tral clustering method based on Nystrmapproximation. The code is available online5 , and we choose the Matlab version with orthogonalization.

• **LSC-K** (Chen and Cai 2011): the landmark-based large- scale spectral clustering method using k-means for land- mark selection. We download the Matlab code from the authors’ website6 .

• **LSH+**bk**-means** (Gong et al. 2015): ﬁrst uses Locality Sensitive Hashing (LSH) (Gionis et al. 1999) to gener- ate a random Gaussian matrix, by which data points are hashed into binary codes. bk-means (Gong et al. 2015) is then applied to the generated binary codes for clustering. This naive two-step method can be viewed as a baseline for binary coding based clustering methods.

We empirically set the number of landmarks in LSC-K and Nystrm as 500 according to the parameter setting in (Chen and Cai 2011), and the number of neighbors in LSC- K as 6. For the binary coding based methods, the binary code length r is set as 32 for CovType, and 128 for the other three high-dimensional datasets. For the proposed CKM, we em- pirically set the ratio of the selected subset β as 0.01, param- eter α as 10, and ν as 1.

curacy is deﬁned as ACC = Σ=1δ(,map(ri)) , where δ(a, b)

denotes the delta function that returns 1 if a = b and 0 oth- erwise, and map(ri ) is the best mapping function for per- muting the cluster labels to match the ground-truth labels. A larger ACC value indicates better clustering performance.

In addition, we also conduct the comparisons in terms of computation and memory costs.

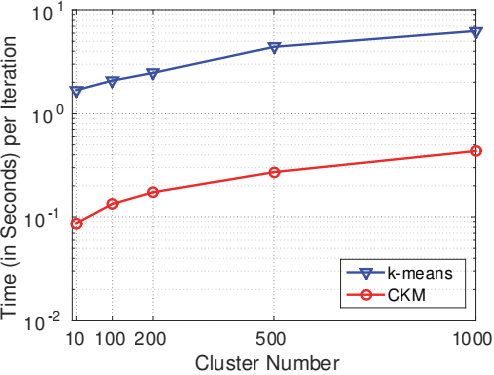
**Results**

**Accuracy:** The accuracy results of all the methods on four datasets are reported in Fig. 2. We ﬁnd several interesting points as follows: 1) Among the comparisons, k-means is stable on four datasets, and k-means++ generally outper- forms k-means. LSC-K performs best on MINIST8M, but fails to work well on ILSVRC2012. LSC-K performs bet- ter than Nystrm. 2) The proposed CKM clearly outper- forms LSH+bk-means. LSH+bk-means is a naive two-step method, thus the generated binary codes may not be optimal for clustering. The accuracy of LSH+bk-means is very low on RCV1, ILSVRC2012. 3) CKM generally achieves com- parable accuracy to the best results.

**Time:** Table 2 shows the running time of all the methods on four datasets. We can see from this table that 1) LSH+bk- means is the fastest among all the methods. It is faster than CKM, because CKM needs the additional time to learn the binary coding function, while the binary function is LSH is random. 2) The proposed CKM takes the second place. It is more efﬁcient than conventional clustering methods. In particular, CKM is nearly 39 times faster than k-means on MINIST8M. 3) k-means++ is generally faster thank-means on CovType and MINIST8M, but slower than k-means on RCV1 and ILSVRC2012. Among the spectral clustering methods, Nystrmis faster than LSC-K.

5<http://alumni.cs.ucsb.edu/>~wychen/sc.html 6<http://www.cad.zju.edu.cn/home/dengcai/>

2531



ACC

0.45

0.4

0.35

0.3

0.25

0.2

0.15

(a) RCV1 (c) ILSVRC2012 (d) MNIST8M

Figure 3: Running time (in seconds) of k-means and CKM for one iteration on (a) RCV1, (b) CovType, (c) ILSVRC2012, (d) MNIST8M. Y axis is in log scale.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | |  |  | |
|  |  | |  |  | |
|  |  | |  |  | |
|  | RCV1 |  |  |
|  |  |  |
|  |  |  | CovType ILSVRC | 2012 |  |
|  |
|  |  |  | MINIST8 | M |  |
|  | |  |  | |
|  |  | |
|  |  | |  |  | |
|  | |  | |

0.01 0.05 0.1 0.5 1

β

Figure 4: Clustering accuracy with respect to different β . X axis is in log scale.

In addition, we compare the running time of k-means and CKM in one iteration. The running time of the two methods in one iteration with respect to different numbers of clusters is shown in Fig. 3. As can be seen, CKM is clearly much faster than k-means among all the cases. This is because CKM uses Hamming metric for distance calculation, which is more efﬁcient than the conventional Euclidean distance calculation ink-means.

**Memory Usage:** Table 3 reports the memory usage of k- means and CKM. We clearly observe that compared with k-means, CKM signiﬁcantly reduce the memory storage of data. Particularly, CKM only needs to store 21.30M of bi- nary codes to represent ILSVRC2012, which is nearly 2048 times memory reduction to k-means. This result implies that CKM can perform clustering over very large-scale dataset in a single machine.

**Sensitivity Study:** We now provide a more careful analysis of the proposed CKM on the sensitivity to the key parame- ters in the clustering tasks. The ratio β is ranged from [0.01, 0.05, 0.1, 0.5, 1], and clustering accuracy with respect to dif- ferent β is shown in Table 4. From Table 4, we observe that the clustering accuracy slightly improves as β increases, that



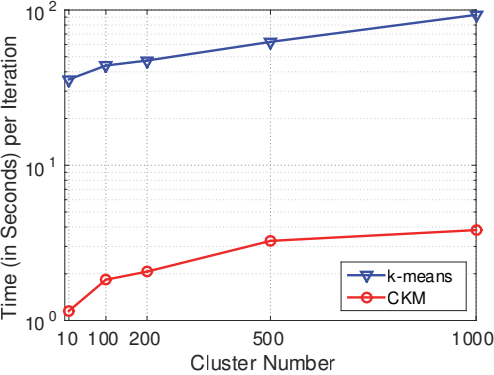
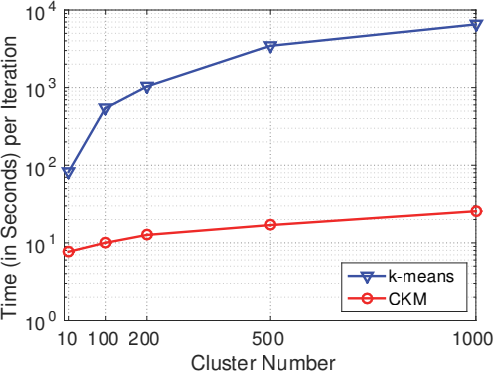
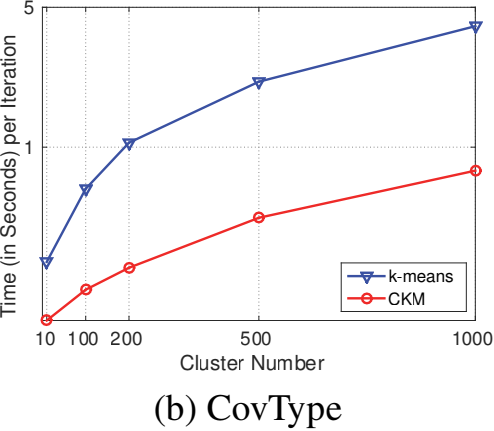
Figure 5: Illustration of cluster samples of ILSVRC2012 dataset generated by the proposed CKM. Each row illus- trates several representative images of one cluster.

is, β does not heavily inﬂuence the clustering performance. This reveals that the binary coding function in CKM is well learned even on a small subset of large-scale datasets. The sensitivtiy results on other paramaters are presented in sup- plementary materials.

**Case Study:** We present a case study in which the proposed CKM is applied to a large-scale image clustering applica- tion. Fig. 5 shows sample clusters of ILSVRC2012 gener- ated by CKM. Each row illustrates several representative images of one cluster. We observe from this ﬁgure that sim- ilar images are well clustered. This case study suggests that CKM works well in practical large-scale clustering applica- tions.

**Conclusion**

This work focuses on the challenging problem of fast clus- tering over large-scale datasets. We propose a novel com- pressed k-means (CKM) to generate optimal binary codes for clustering. Compared to existing clustering methods, CKM enjoys both computational and memory efﬁciency. Extensive experiment results on four large-scale datasets, including two million-scale datasets, suggests that CKM is able to cluster very fast with limited memory, yet achieves



2532

comparable accuracy to the state-of-the-art methods.

**Acknowledgments**

This research is supported by the National Science Foun- dation of China (Grant No. 61273251, 61673220), the ARC Future Fellowship FT130100746, and ARC grant LP150100671.

**References**

Arthur, D., and Vassilvitskii, S. 2007. k-means++: The ad- vantages of careful seeding. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035.

Bachem,O.; Lucic,M.; Hassani,S. H.; and Krause,A. 2016. Approximate k-means++ in sublinear time. In *AAAI*, 1459– 1467.

Chen, X., and Cai, D. 2011. Large scale spectral clustering with landmark-based representation. In *AAAI*, 313–318.

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

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei- Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255.

Ding, Y.; Zhao, Y.; Shen, X.; Musuvathi, M.; and Mytkow- icz, T. 2015. Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup. In *ICML*, 579– 587.

Drake, J., and Hamerly, G. 2012. Accelerated k-means with adaptive distance bounds. In *5th NIPS workshop on opti- mization for machine learning*, 42–53.

Elkan, C. 2003. Using the triangle inequality to accelerate k-means. In *ICML*, volume 3, 147–153.

Gionis, A.; Indyk, P.; Motwani, R.; et al. 1999. Similarity search in high dimensions via hashing. In *VLDB*, 518–529. Gong, Y., and Lazebnik, S. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 817–824.

Gong, Y.; Pawlowski, M.; Yang, F.; Brandy, L.; Bourdev, L.; and Fergus, R. 2015. Web scale photo hash clustering on a single machine. In *CVPR*, 19–27.

Hamerly, G. 2010. Making k-means even faster. In *SDM*, 130–140.

Hartigan,J. A., and Wong,M. A. 1979. Algorithm as 136: A k-means clustering algorithm. *Applied Statistics* 28(1):100– 108.

Hazan, T.; Keshet, J.; and McAllester, D. A. 2010. Direct loss minimization for structured prediction. In *NIPS*, 1594– 1602.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classiﬁcation with deep convolutional neural net- works. In *NIPS*, 1097–1105.

Li,Y.-F.; Tsang,I. W.; Kwok,J. T.-Y.; and Zhou,Z.-H. 2009. Tighter and convex maximum margin clustering. In *AIS- TATS*, 344–351.

Li, Y.; Nie, F.; Huang, H.; and Huang, J. 2015. Large-scale multi-view spectral clustering via bipartite graph. In *AAAI*, 2750–2756.

Lin, G.; Shen, C.; and van den Hengel, A. 2015. Supervised hashing using graph cuts and boosted decision trees. *TPAMI* 37(11):2317–2331.

Liu, W., and Tsang, I. W. 2015. Large margin metric learn- ing for multi-label prediction. In *AAAI*, 2800–2806.

Ng, A. Y.; Jordan, M. I.; Weiss, Y.; et al. 2001. On spectral clustering: Analysis and an algorithm. In *NIPS*, 849–856. Norouzi, M.; Fleet, D. J.; and Salakhutdinov, R. R. 2012. Hamming distance metric learning. In *NIPS*, 1061–1069. Shen, X.; Shen, F.; Sun, Q.-S.; Yang, Y.; Yuan, Y.-H.; and Shen, H. T. 2017. Semi-paired discrete hashing: Learn- ing latent hash codes for semi-paired cross-view retrieval. *TCYB*.

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *TPAMI* 22(8):888–905.

Song, D.; Liu, W.; and Meyer, D. A. 2016. Fast structural binary coding. In *IJCAI*, 2018–2024.

Wang, H.; Nie, F.; Huang, H.; and Makedon, F. 2011a. Fast nonnegative matrix tri-factorization for large-scale data co- clustering. In *IJCAI*, 1553–1558.

Wang, Y.; Jiang, Y.; Wu, Y.; and Zhou, Z.-H. 2011b. Local and structural consistency for multi-manifold clustering. In *IJCAI*, 1559–1564.

Wang,J.; Liu,W.; Kumar, S.; and Chang, S.-F. 2016. Learn- ing to hash for indexing big data-a survey. *Proceedings of the IEEE* 104(1):34–57.

Wu, X.; Kumar, V.; Quinlan, J. R.; Ghosh, J.; Yang, Q.; Mo-

toda, H.; McLachlan, G. J.; Ng, A.; Liu, B.; Philip, S. Y.; et al. 2008. Top 10 algorithms in data mining. *Knowledge and Information Systems* 14(1):1–37.

Yu, C.-N. J., and Joachims, T. 2009. Learning structural svms with latent variables. In *ICML*, 1169–1176.

Zhang, R., and Lu, Z. 2016. Large scale sparse clustering. In *IJCAI*, 2336–2342.

Zhou, J. T.; Xu, X.; Pan, S. J.; Tsang, I. W.; Qin, Z.; and Goh, R. S. M. 2016. Transfer hashing with privileged infor- mation. In *IJCAI*, 2414–2420.