Effective and Sparse Count-Sketch via k-means clustering

Anonymous submission

Abstract

Count-sketch is a popular matrix sketching algorithm that can produce a much smaller sketched matrix of an input data matrix $\mathbf{X}$ in $O(nnz(\mathbf{X}))$ time while preserving most of its properties. Therefore, count-sketch is widely used for addressing high-dimensionality challenge in machine learning. However, count-sketch has two main limitations: (1) The randomly generated sketching matrix used in count-sketch does not consider any intrinsic properties of $\mathbf{X}$. This data-oblivious method could produce a bad sketched matrix which results in low accuracy for subsequent machine learning tasks (e.g., classification); (2) For highly sparse input data, count-sketch could produce a dense sketched data matrix and make the subsequent machine learning tasks more computationally expensive than on the original sparse data $\mathbf{X}$. To a ddress these two limitations, we first show an interesting connection between count-sketch and k-means clustering by analyzing the reconstruction error of count-sketch. Based on our analysis, we propose to obtain the low-dimensional sketched matrix by applying k-means clustering on the columns of $\mathbf{X}$ and use the cluster centers as the low-dimensional sketched matrix. In addition, to produce a sparse sketched matrix, we propose to solve k-mean clustering using gradient descent with ϵ-$\mathcal{L}_1$ ball projection on each iteration. Our experimental results based on six benchmark datasets have demonstrated that our method achieves higher accuracy than the original count-sketch and other matrix sketching algorithms. Our results also demonstrate that our method produces a much sparser sketched data matrix than other methods and therefore the prediction cost of our method is smaller than other methods.

Introduction

Matrix sketching (Woodruff 2014) is a powerful dimensionality reduction method that can efficiently find a small matrix to replace the original large matrix while preserving most of its properties. For an input large data matrix $\mathbf{X} \in R^{n\times d}$ where n is the number of samples and d is the number of features, matrix sketching methods generate a sketch of $\mathbf{X}$ by multiplying it with a random sketching matrix $\mathbf{R} \in R^{d\times r}$ ($r \ll d$) with certain properties. Compared with traditional dimensionality reduction methods (e.g., Principal Component Analysis (PCA) (Jolliffe 2011)), matrix sketching methods can obtain the sketched matrix very efficiently with certain theoretical guarantees (Woodruff 2014). Therefore, matrix sketching has gained significant research attention and has been used widely for handling high-dimensional data in machine learning (Mahoney 2011; Ailon and Chazelle 2006; Bojarski et al. 2017; Choromanski, Rowland, and Weller 2017).

A typical way of applying matrix sketching in machine learning problems is *sketch and solve* (Dahiya, Konomis, and Woodruff 2018). For example, in a linear classification problem with training data $\{\mathbf{X}, \mathbf{y}\}$ where $\mathbf{X} \in R^{n\times d}$ is a large input feature matrix and $\mathbf{y} \in R^n$ is the corresponding label vector, a classification model can be trained by solving $\min_{\mathbf{w}\in R^d} \sum_{i=1}^{n} l(\mathbf{w}^T\mathbf{x}_i, y_i) + \lambda\|\mathbf{w}\|_2$ where $l(\cdot,\cdot)$ denotes a loss function (e.g., hinge loss). By using matrix sketching, we can first obtain a sketched data matrix $\widetilde{\mathbf{X}} \in R^{n\times r}$ by $\widetilde{\mathbf{X}} = \mathbf{XR}$ and then solve a much smaller problem $\min_{\mathbf{v}\in R^r} \sum_{i=1}^{n} l(\mathbf{v}^T\widetilde{\mathbf{x}}_i, y_i) + \lambda\|\mathbf{v}\|_2$. The expensive computation on the original large matrix $\mathbf{X} \in R^{n\times d}$ can be replaced by computation on the small sketched matrix $\widetilde{\mathbf{X}}R^{n\times r}$. This *sketch and solve* method has also been used to speed up other machine learning tasks, such as least squares regression (Dobriban and Liu 2019), low-rank approximation (Tropp et al. 2017; Clarkson and Woodruff 2017) and k-means clustering (Boutsidis, Zouzias, and Drineas 2010; Liu, Shen, and Tsang 2017).

Recent advances in randomized numerical linear algebra (Martinsson and Tropp 2020) have provided a solid theoretical foundation for matrix sketching. Various methods have been proposed to construct the random matrix $\mathbf{R}$. The early method (Dasgupta and Gupta 1999) constructs a dense random Gaussian matrix $\mathbf{R}$ where each element in $\mathbf{R}$ is generated from a Gaussian distribution $\mathcal{N}(0, \frac{1}{d})$. This method requires $O(ndr)$ time for computing the sketched matrix $\widetilde{\mathbf{X}} = \mathbf{XR}$ since the random Gaussian matrix $\mathbf{R}$ is dense. (Achlioptas 2003) proposed to generate a sparser random matrix $\mathbf{R}$ where each element in $\mathbf{R}$ is generated from $\{-1, 0, 1\}$ following a discrete distribution. It will reduce the computation complexity from $O(ndr)$ to $O(\frac{1}{3}ndr)$. In recent years, two famous fast random projection matrices were proposed for efficiently computing the projection $\widetilde{\mathbf{X}} = \mathbf{XR}$. The first method is called Subsampled Randomized Hadamard Transform (SRHT) which can achieve $O(ndlog(r))$ time for computing $\mathbf{XR}$ (Tropp 2011; Ailon and Liberty 2009). The second method is called count-sketch (Clarkson and Woodruff 2017) which can compute $\mathbf{XR}$ in $O(nnz(\mathbf{X}))$ time for any input $\mathbf{X}$ making the Count-

Sketch method particularly suitable for sparse input data. In this paper, we focus on improving the count-sketch algorithm in the context of classification.

Count-sketch constructs the random matrix $\mathbf{R}$ by a product of two matrices $\mathbf{D}$ and $\mathbf{\Phi}$, i.e., $\mathbf{R} = \mathbf{D}\mathbf{\Phi}$, where $\mathbf{D} \in R^{d\times d}$ is a random diagonal matrix where each diagonal values is uniformly chosen from $\{1, -1\}$ and $\mathbf{\Phi} \in R^{d\times r}$ is a very sparse matrix where each row has only one randomly selected entry equal to 1 and all other entries are 0. Previously, (Paul et al. 2014) applied count-sketch for linear SVM classification and showed that linear SVM trained on the sketched data matrix has comparable generalization ability as in the original space in the case of classification. However, we argue that there are two main limitations of count-sketch: (1) It is a data-oblivious method where the generation of sketching matrix $\mathbf{R}$ is totally independent of input data matrix $\mathbf{X}$. The sketched matrix may not be effective for the subsequent classification algorithm; (2) The sketched data matrix $\tilde{\mathbf{X}}$ could be much denser than the original input data $\mathbf{X}$. The dense matrix could make the subsequent classification algorithm on the sketched data more computationally expensive than on the original data $\mathbf{X}$ because the number of non-zero values in projected data could be larger than the number of non-zero values in the original data. Even though data-oblivious matrix sketching has been extensively studied, few studies focus on efficient data-dependent matrix sketching. Recently, (Xu et al. 2017) proposed to use the approximated singular value decomposition (SVD) as the projection subspace. (Lei and Lan 2020) proposed to improve SRHT by non-uniform sampling by exploiting data properties. However, both of them will produce a dense sketched matrix for sparse input data.

In this paper, we focus on addressing the aforementioned two limitations of count-sketch. To address the first limitation, we first show an interesting connection between count-sketch and k-means clustering by analyzing the reconstruction error of count-sketch. Based on our analysis, we propose to reduce the reconstruction error of count-sketch by using k-means clustering on the columns of $\mathbf{X}$. The resulting sparse cluster centers are used as the low-dimensional sketched data matrix. To address the second limitation, we propose to get sparse cluster centers by optimizing k-means objective function using gradient descent with ϵ-$\mathcal{L}_1$ ball projection in each iteration. Finally, we compare our proposed methods with the other five popular matrix sketching algorithms on six real-life datasets. Our experimental results clearly demonstrate that our proposed data-dependent matrix sketching methods achieve higher accuracy than count-sketch and other random matrix sketching algorithms. Our results also show our method produces a sparser sketched data matrix than count-sketch and other matrix sketching methods. The prediction cost of our method is smaller than other matrix sketching methods.

Preliminaries

Randomized Matrix Sketching

Given a data matrix $\mathbf{X} \in R^{n\times d}$ and a random sketching matrix $\mathbf{R} \in R^{d\times r}$ with $r \ll d$, a sketched matrix is produced by

$$\widetilde{\mathbf{X}} = \mathbf{X}\mathbf{R} \in R^{n\times r}. \qquad (1)$$

Note that the matrix $\mathbf{R}$ is randomly generated and is independent of the input data $\mathbf{X}$. As shown in the following Johnson-Lindenstrauss Lemma (JL lemma), randomized matrix sketching can preserve the pairwise distance of all data points using the sketched data matrix $\widetilde{\mathbf{X}}$.

Lemma 1 (JL lemma (Johnson and Lindenstrauss 1984)) *For any $0 < \epsilon < 1$ and any integer n, let $r = O(\log n/\epsilon^2)$ and $\mathbf{R} \in R^{d\times r}$ be a random orthonormal matrix. Then for any set $\mathbf{X}$ of n points in R^d, the following inequality about the pairwise distance between any two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathbf{X}$ holds true with high probability:*

$$(1-\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2 \le \|\mathbf{R}^T\mathbf{x}_i - \mathbf{R}^T\mathbf{x}_j\|_2 \le (1+\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2.$$

Count-Sketch

Among various methods for constructing the sketching matrix $\mathbf{R}$, count-sketch (or called sparse embedding) is well suited for sparse input data $\mathbf{X}$ since it can achieve $O(nnz(\mathbf{X}))$ time complexity for computing $\mathbf{X}\mathbf{R}$. Count-sketch (Clarkson and Woodruff 2017) constructs the random matrix $\mathbf{R} \in R^{d\times r}$ as $\mathbf{R} = \mathbf{D}\mathbf{\Phi}$ where $\mathbf{D}$ and $\mathbf{\Phi}$ are defined as follows,

- $\mathbf{D}$ is a $d \times d$ diagonal matrix with each diagonal entry independently chosen to be 1 or -1 with a probability of 0.5.
- $\mathbf{\Phi} \in \{0,1\}^{d\times r}$ is a $d \times r$ binary matrix with $\mathbf{\Phi}_{i,h(i)} = 1$, and all remaining entries are 0. h is a random map such that for any $i \in \{1, 2, \dots, d\}$, $h(i) = j$, for $j \in \{1, 2, \dots, r\}$ with a probability of $\frac{1}{r}$.

Note that random sketching matrix $\mathbf{R}$ in count-sketch is a very sparse matrix where each row has only one nonzero entry. This nonzero entry is uniformly chosen and the value is either 1 or -1 with a probability of 0.5. $\mathbf{X}\mathbf{R}$ can be computed in $O(nnz(\mathbf{X}))$ time because each nonzero entry in $\mathbf{X}$ is at most by multiplied by one nonzero entry in $\mathbf{X}\mathbf{R}$. (Paul et al. 2014) applied count-sketch for linear SVM classification and showed that linear svm trained on the sketched data can ensure comparable generalization ability as in the original space in the case of classification.

Methodology

Even though count-sketch has been successfully used for dimensionality reduction in linear SVM classification (Paul et al. 2014), we argue that this data-oblivious method has two limitations: (1) The sketching matrix $\mathbf{R} = \mathbf{D}\mathbf{\Phi}$ is randomly generated. It could result in bad sketched data when some important columns in $\mathbf{X}$ are not sampled using $\mathbf{R}$; (2) Count-sketch will not preserve the sparsity rate of the original data.

When using count-sketch in data classification, the first limitation could result in bad low-dimensional embedding and then produce a classification model with low accuracy. To illustrate this limitation, we show the classification accuracy of using count-sketch for dimensionality reduction on mnist dataset for ten different runs in Figure 1. As shown in

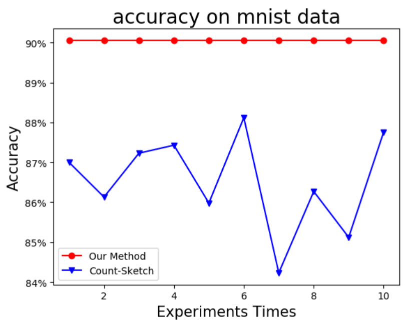


Figure 1: Classification accuracy of using count-sketch on different runs

Figure 1, count-sketch is not stable. It produces low classification accuracy in some runs. We also show the classification of our proposed method that will be introduced later in this figure. We can see that our proposed method obtains higher and stabler accuracy than count-sketch.

The second limitation of count-sketch is that the sketched matrix could be much denser than the original data when used with sparse input data. We checked the sparsity rate of mnist data before and after count-sketch. The original sparsity rate for mnist data is 80.78% and the sparsity rate is decreased to 1.72% in the sketched data. Therefore, the sketched data could be much denser than the original data and make the subsequent classification algorithm slower because the number of non-zero values in sketched data could be larger than the number of non-zero values in the original data. More results will be discussed in the experiment section.

Connection between Count-Sketch and k-means clustering

Since the construction of matrix $\mathbf{D}$ and $\mathbf{\Phi}$ in count-sketch is oblivious to the input data matrix $\mathbf{X}$, it could produce a bad sketched matrix (e.g., some important columns in $\mathbf{X}$ are not be sampled in $\mathbf{\Phi}$) and therefore results in low classification accuracy. In this paper, we seek to develop a data-dependent count-sketch method for addressing the two limitations of count-sketch. To motivate our method, we start by analyzing the reconstruction error of the count-sketch method and show an interesting connection between count-sketch and k-means clustering.

Let us define a diagonal scaling matrix $\mathbf{S} \in R^{r \times r}$ as

$$\mathbf{S}_{ii} = \frac{1}{\sum_{j=1}^{d} \Phi_{ji}} \tag{2}$$

Note that $(\mathbf{D\Phi S}^{\frac{1}{2}})^T(\mathbf{D\Phi S}^{\frac{1}{2}})$ equals to an identity matrix with size $r \times r$. The reconstruction error of count-sketch can be represented as

$$\|\mathbf{X} - \mathbf{X}(\mathbf{D\Phi S}^{\frac{1}{2}})(\mathbf{D\Phi S}^{\frac{1}{2}})^T\|_F^2 = \|\mathbf{X} - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2 \tag{3}$$

where $\|\mathbf{A}\|_F$ denotes the Frobenius norm of matrix $\mathbf{A}$. As shown in the following Proposition 1, the reconstruction error of count-sketch as shown in (3) is equivalent to the objective function of applying k-means clustering to cluster the d columns of $\mathbf{M} = \mathbf{XD}$ into r clusters.

Proposition 1 *The reconstruction error of count-sketch $\|\mathbf{X} - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2$ is equivalent to the objective function of applying k-means clustering on the columns of matrix product $\mathbf{M} = \mathbf{XD}$ if we treat $\mathbf{\Phi}$ as a learnable variable which denotes the cluster membership of each column in $\mathbf{M}$,*

$$\begin{aligned} \|\mathbf{X} - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2 &= \|\mathbf{M} - \mathbf{M\Phi S\Phi}^T\|_F^2 \\ &= \sum_{i=1}^{d} \|\mathbf{M}_{(:,i)} - \mathbf{c}_{I(\mathbf{M}_{(:,i)})}\|_2^2, \end{aligned} \tag{4}$$

where $\mathbf{M}_{(:,i)}$ denotes the i-th column of $\mathbf{M}$, $I(\mathbf{M}_{(:,i)})$ returns the index of the cluster that $\mathbf{M}_{(:,i)}$ belongs to and $\mathbf{c}_{I(\mathbf{M}_{(:,i)})}$ is the centroid of that cluster.

The proof of proposition 1 is in the appendix section. The proposition 1 provides an interesting connection between count-Sketch and k-means clustering. In the count-Sketch algorithm, the clustering membership indicator matrix $\mathbf{\Phi}$ is randomly generated which does not consider intrinsic data properties. It could result in bad embedding with high reconstruction error.

Improved count-sketch by k-means and $\mathcal{L}_1$ ball projection

As shown in (12), the reconstruction error of count-sketch can be improved by replacing the random cluster membership indicator matrix $\mathbf{\Phi}$ in the original count-sketch algorithm with a cluster membership indicator matrix produced by k-means algorithm on the columns of $\mathbf{M}$. Motivated by this observation, we propose to use k-means algorithm to learn the cluster membership indicator matrix $\mathbf{\Phi}$ from data to achieve lower reconstruction error. Therefore, the new cluster centers returned by k-means with $k = r$, which equals to $\mathbf{XD\Phi S}$, can be used as the new low-dimensional feature representation. As shown in proposition 1. This new sketched matrix will result in low reconstruction error than the original count-sketch method.

Apart from the reconstruction error, as mentioned earlier, another limitation of count-sketch is that it may not preserve the sparsity rate of the input data $\mathbf{X}$. In other words, the new data presentation $\widetilde{\mathbf{X}}$ could be dense even if the original data $\mathbf{X}$ is highly sparse data. This limitation could make the subsequent algorithm on projected data $\widetilde{\mathbf{X}}$ be even slower than just using the original data $\mathbf{X}$ without count-sketch. Therefore, instead of using the Lloyd's classic k-means algorithm (Lloyd 1982), we would like to develop a new method to obtain very sparse cluster centers. We propose to obtain sparse cluster centers by optimizing the objective of k-means as shown in (12) using gradient descent together with $\mathcal{L}_1$ ball projection (Duchi et al. 2008) in each update.

The gradient of the k-means objective function $\sum_{i=1}^{d} \|\mathbf{M}_{(:,i)} - \mathbf{c}_{I(\mathbf{M}_{(:,i)})}\|_2^2$ with respect to the j-th cluster center $\mathbf{c}_j$ is

$$\nabla \mathbf{c}_j = \sum_{i=1}^{d} -2\delta(I(\mathbf{M}_{(:,i)}), j)(\mathbf{M}_{(:,i)} - \mathbf{c}_j), \tag{5}$$

where $\delta(I(\mathbf{M}_{(:,i)}), j)$ is a binary function which returns 1 if $I(\mathbf{M}_{(:,i)})$ equals to j (i.e., the i-th column of $\mathbf{M}$ belongs to

Algorithm 1: ϵ-$\mathcal{L}_1$ ball projection (Sculley 2010)

```
Input: $\mathbf{c} \in R^n$, $\mathcal{L}_1$ ball radius $\lambda$, tolerance parameter $\epsilon$
Output: projected sparse vector $\mathbf{c} \in R^n$
if $\|\mathbf{c}\| \leq \lambda(1+\epsilon)$ return c
$l = 0$; $u = \|\mathbf{c}\|_\infty$; $r = \|\mathbf{c}\|_1$
while $r > \lambda(1+\epsilon)$ or $r < \lambda$ do      # Bisection to find $\theta$
    $\theta = \frac{l+u}{2}$
    $r = \sum_{i=1}^{n} \max(0, |c_i| - \theta)$
    if $r < \lambda$ then $u = \theta$ else $l = \theta$
end while
for $i = 1$ to $n$ do                          # $\mathcal{L}_1$ ball projection
    $c_i = \text{sign}(c_i)\max(0, |c_i| - \theta)$
end for
```

the j-th cluster). Otherwise it returns 0. In other words, the computation of gradient $\nabla \mathbf{c}_j$ only depends on columns that belong to the j-th cluster in the current iteration.

By using gradient descent, in each iteration, the cluster center $\mathbf{c}_j$ can be updated as

$$\mathbf{c}_j = \mathbf{c}_j - \eta \nabla \mathbf{c}_j, \tag{6}$$

where η is the learning rate. However, directly using (6) will not produce sparse cluster centers.

To obtain sparse cluster centers, we will use ϵ-$\mathcal{L}_1$ ball projection to make $\mathbf{c}_j$ be a sparse vector. The ϵ-$\mathcal{L}_1$ ball projection is proposed in (Sculley 2010) which is approximated extension of the exact $\mathcal{L}_1$ ball projection (Duchi et al. 2008). ϵ-$\mathcal{L}_1$ is very effective at getting sparse cluster centers as shown in (Sculley 2010). The basic idea of ϵ-$\mathcal{L}_1$ ball projection is to use bisection to find a value θ that projects a dense vector $\mathbf{c}_j$ to an $\mathcal{L}_1$ ball with a radius between λ and $(1+\epsilon)\lambda$. After θ is found, ϵ-$\mathcal{L}_1$ ball projection will map the i-th entry in $\mathbf{c}_j$ (denoted as c_{ji}) to

$$c_{ji} = \text{sign}(c_{ji})\max(0, |c_{ji}| - \theta). \tag{7}$$

As shown in (7), the resulting cluster centers $\mathbf{c}_j$s will be sparse vectors since $\max(0, |c_{ji}| - \theta)$ will make an element to 0 if its absolute value is smaller than θ. The whole procedure of ϵ-$\mathcal{L}_1$ ball projection is described in Algorithm 1.

By using Algorithm 1 in each iteration of optimizing k-means objective function by gradient descent, we will get sparse cluster centers.

Algorithm Implementation and Analysis

We summarize our proposed algorithm for improving the original count-sketch in algorithm and name it as **E**ffective and **S**parse **C**ount-**SK**etch (ESCK). Our proposed algorithm first obtain $\mathbf{M}$ as shown in step 1-2 which is the same as the original count-sketch. The contribution of our proposed algorithm is to replace the randomly generated cluster membership indicator matrix Φ in count-sketch with the learned cluster membership indicator matrix Φ. The r sparse cluster centers will be used as the low-dimensional data representation. As shown from step 3 to 12, sparse cluster centers are obtained by using gradient descent with ϵ-$\mathcal{L}_1$ ball projection to cluster d columns into r groups.

With respect to time complexity, step 2 only needs $O(nnz(\mathbf{X}))$ time because $\mathbf{D}$ is a diagonal matrix. The time complexity for steps 3 to 12 is upper bounded by $O(ndrt)$ where t is the number of iterations. For sparse input data, the time complexity in each iteration for updating cluster centers will smaller than $O(ndr)$ since both data and clusters are sparse. Empirically, the k-means algorithm using gradient descent converges very fast and only a few iterations are needed. In our experiments, we will show that our proposed method is only several times slower than count-sketch. However, the classification accuracy obtained by our method is much larger than count-sketch and other methods. Note that our proposed method can also return the learned cluster membership indicator matrix $\mathbf{\Phi}$. Therefore, our algorithm can be extended to an inductive setting and generate the feature mapping for new unseen data by using $\widetilde{\mathbf{X}} = \mathbf{XD\Phi S}$ which enjoys the same low computational complexity as the count-sketch method.

Algorithm 2: **E**ffective and **S**parse **C**ount-**SK**etch (ESCK)

```
Input: $\mathbf{X} \in \mathbf{R}^{n \times d}$, reduced dimension $r$, iteration $t$, parameter $\epsilon$, $\lambda$ for $\mathcal{L}_1$ ball projection;
Output: low-dimensional data representation $\widetilde{\mathbf{X}} \in R^{n \times r}$ and the learnt cluster membership indicator matrix $\mathbf{\Phi}$
 1: Generate a diagonal random sign matrix $\mathbf{D}$
 2: Compute $\mathbf{M} = \mathbf{XD}$
 3: Randomly pick $r$ columns from $\mathbf{M}$ as the cluster centers $\{\mathbf{c}_j\}_{j=1}^{r}$
 4: for $iter = 1$ to $t$ do
 5:     Create all zero matrix $\mathbf{\Phi} \in R^{d \times r}$
 6:     for $i = 1$ to $d$ do
 7:         $j = \text{argmin}_j \|\mathbf{M}_{(:,i)} - \mathbf{c}_j\|_2^2$
 8:         $\mathbf{\Phi}_{i,j} = 1$
 9:     end for
10:     Update each cluster centers using (6)
11:     Obtain sparse cluster centers using Algorithm 1
12: end for
13: return $\widetilde{\mathbf{X}} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_r]$ and $\mathbf{\Phi}$
```

Dataset	# of samples	# of features	# of classes	sparsity rate
usps	9,298	256	10	0%
mnist	60,000	780	10	80.78%
gisette	7,000	5,000	2	0.85%
real-sim	72,309	20,958	2	99.75%
rcv1-binary	20,242	47,236	2	99.84%
rcv1-multi	15,564	47,236	53	99.86%

Table 1: Summary of experimental datasets

Experiments

In this section, we compare our methods with different commonly-used random dimensionality reduction algorithms based on six real-life datasets. These datasets are downloaded from LIBSVM website(Chang and Lin 2011). The summarization of these six datasets is shown in Table 1. The sparsity rate as shown in the last column is the fraction

	Performance	usps (r=30)	mnist (r=100)	gisette (r=256)	real-sim (r=256)	rcv1-binary (r=256)	rcv1-multi (r=256)
PCA	Accuracy(%)	***93.61 ± 0.01***	**90.55 ± 0.03**	**94.52 ± 0.01**	-	-	***84.20 ± 0.01***
	Sparsity rate	0%	0%	0%			0%
	Prediction time(ms)	3.0	21.9	3.8			22.4
Gaussian	Accuracy(%)	90.60 ± 0.01	88.92 ± 0.03	90.70 ± 0.01	79.25 ± 0.06	81.63 ± 0.01	69.71 ± 0.01
	Sparsity rate	0%	0%	0%	0%	0%	0%
	Prediction time(ms)	2.9	20.8	3.0	31.2	8.0	22.4
Achlioptas	Accuracy(%)	90.17±0.03	87.70±0.02	89.80± 0.03	76.85±0.06	81.86±0.01	67.56±0.07
	Sparsity rate	0%	0.01%	0%	1.07%	0.03%	0.03%
	Prediction time(ms)	3.1	55.0	4.0	41.9	11.7	88.8
Count-Sketch	Accuracy(%)	90.74 ± 0.01	87.66 ± 0.01	90.37 ± 0.02	77.21 ± 0.06	80.19±0.02	69.38± 0.06
	Sparsity rate	0%	1.72%	0%	73.36%	73.65%	74.47%
	Prediction time(ms)	3.0	51.0	4.5	12.0	4.5	24.9
SRHT	Accuracy(%)	89.86 ± 1.66	87.14 ± 0.84	90.45 ± 0.87	78.37 ± 0.20	80.29 ± 0.69	68.50 ± 0.29
	Sparsity rate	0%	0%	0%	0%	0%	0%
	Prediction time(ms)	3.3	79.4	3.6	22.6	7.0	103.0
SRHT-topr	Accuracy(%)	90.68 ± 1.51	88.15 ± 0.77	92.45 ± 0.55	82.48 ± 0.19	82.14 ± 0.24	71.01 ± 0.77
	Sparsity rate	0%	0%	0%	0%	0%	0%
	Prediction time(ms)	4.2	75.2	3.3	46.6	6.1	101.0
ESCK-full	Accuracy(%)	**92.55± 0.02**	***90.60±0.02***	***95.13± 0.02***	***88.68±0.07***	***92.91±0.01***	**78.99 ± 0.01**
	sparsity rate	16.60%	43.10%	3.66%	89.57%	87.61%	88.44%
	Prediction time(ms)	1.5	15.8	2.5	4.0	1.0	13.5
ESCK-miniBatch	Accuracy(%)	92.18± 0.01	90.50±0.02	94.45 ± 0.03	**88.25±0.08**	**90.01±0.02**	77.13 ± 0.01
	sparsity rate	46.78%	40.29%	37.58%	97.47%	94.78%	95.37%
	Prediction time(ms)	2.0	16.9	1.3	1.0	1.0	6.0

Table 2: Experimental results of different random matrix sketching methods

of zeros in each input data matrix $\mathbf{X}$. As shown in Table 1, there are four sparse datasets (mnist, real-sim, rcv1-binary, rcv1-multi) and two dense datasets (usps, gisette). We evaluate the performance of the following seven matrix sketching methods:

- Gaussian: The sketching matrix is a random Gaussian Matrix (Dasgupta and Gupta 1999).
- Achlioptas: A sparser sketching matrix is randomly generated from a discrete distribution (Achlioptas 2003).
- Count-Sketch: original oblivious count-sketch method (Clarkson and Woodruff 2017).
- SRHT : The sketching matrix is generated by SRHT (Tropp 2011).
- SRHT-topr An improved variant of SRHT which is data-dependent (Lei and Lan 2020).
- ESCK-full: our proposed method that uses full batch gradient descent with ϵ-$\mathcal{L}_1$ ball projection.
- ESCK-miniBatch: our proposed method that uses mini-batch gradient descent with ϵ-$\mathcal{L}_1$ ball projection.

We also include the results of using PCA (Jolliffe and Cadima 2016) for dimensionality reduction. Due to the high computational complexity of PCA, we cannot get the results of PCA on real-sim and rcv1-binary datasets.

Experimental Setting. For the two dense datasets (usps and gisette), we have scaled the feature values to $[-1, 1]$ using min-max normalization. We use five-fold cross-validation to evaluate the accuracy of different random matrix sketching methods. The regularization parameter C in SVM is chosen from $\{10^{-5}, 10^{-4}, \ldots, 10^4, 10^5\}$. The ϵ parameter for ϵ-$\mathcal{L}_1$ ball projection is fixed to 0.1 and λ parameter is chosen from $\{10, 20, 30, 40\}$. Our experiments are performed on a desktop with Intel(R) Core(TM) i7-9700 CPU and @ 3.00GHz and 16.0 GB RAM.

Experimental Results. We report the classification accuracy, sparsity rate of the sketched matrix and prediction time of different algorithms in Table 2. The projected dimension r for each dataset is given in the first row of this table. The results for different settings of projected dimension r will be discussed later. The results of PCA in given in the second row. For random matrix sketching methods, the first four methods are data-oblivious random projection methods and the last three are data-dependent random projection methods. The best accuracy for each dataset is in bold and italic and the second-best accuracy for each dataset is in bold.

Table 2 shows that the data-dependent matrix sketching methods (i.e., SRHT-topr, ESCK-full and ESCK-miniBatch) get higher accuracy than data-independent matrix sketching methods. Among the random matrix sketching methods, the proposed ESCK-full algorithm achieves the best accuracy on all six datasets. The proposed method ESCK-miniBatch gets slightly lower accuracy than ESCK-full but gets higher accuracy than the other five matrix sketching methods. The results in Table 2 demonstrate that our proposed methods achieve better accuracy than other random matrix sketching methods.

With respect to the sparsity rate of the sketched data, as expected, Gaussian, Achlioptas, SRHT and SRHT-topr will produce dense data even if the input data is sparse. The original count-sketch method and our proposed methods can produce sparse embedding for highly sparse input data. The sparsity rate of the sketched data produced by our proposed methods is higher than the count-sketch. Furthermore, our proposed method could result in sparse embedding for dense

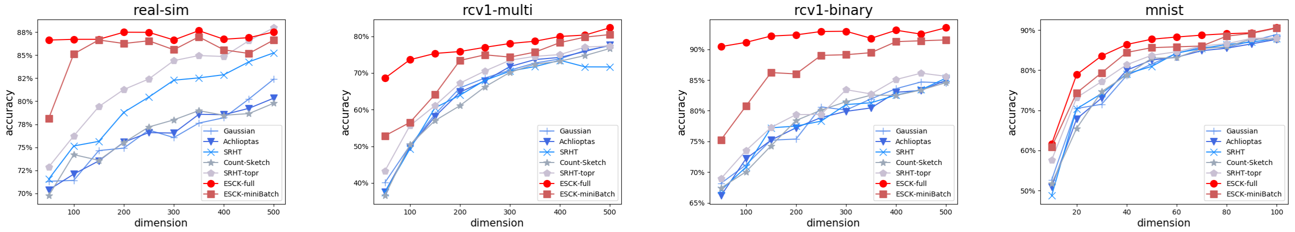
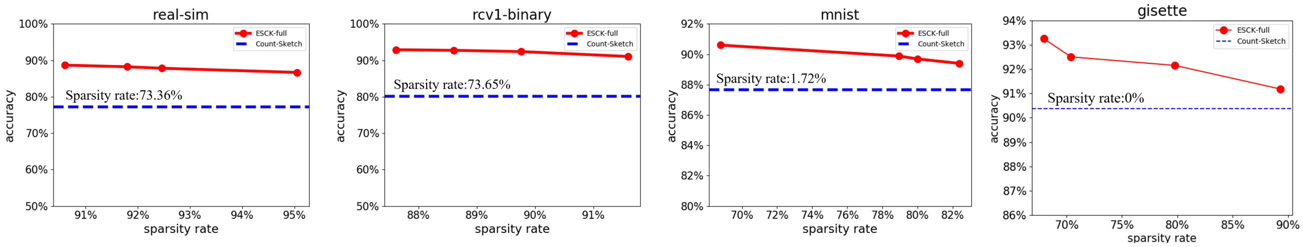


Figure 2: Impact of Projection Dimension r



(a) real-sim (b) rcv1-binary (c) mnist (d) gisette

Figure 3: Accuracies with Different Sparsity Rates

input data (e.g., usps and gisette). With respect to the prediction time, the prediction time of our methods is lower than other methods. We summarize the prediction costs for different algorithms on a single input data sample in Table 3. In this table, we decompose the prediction cost into (1) Project cost (i.e., computing $\widetilde{\mathbf{x}} = \mathbf{R}^T\mathbf{x}$) and classification cost (i.e., computing $\mathbf{v}^T\widetilde{\mathbf{x}}$). As shown in Table 3, both count-sketch and our proposed ESCK are very efficient for prediction.

Methods	Projection Cost	Classification Cost
Gaussian	$O(dr)$	$O(r)$
Achlioptas	$O(dr)$	$O(r)$
Count-Sketch	$O(nnz(\mathbf{x}))$	$O(nnz(\tilde{\mathbf{x}}))$
SRHT	$O(dlog(d))$	$O(r)$
SRHT-topr	$O(dlog(d))$	$O(r)$
ESCK	$O(nnz(\mathbf{x}))$	$O(nnz(\tilde{\mathbf{x}}))$

Table 3: Prediction cost for different methods on a single input $\mathbf{x}$

Impact of Projection Dimension r. In Figure 2, we show the results of different algorithms with different projection dimension r. As shown in the figure, our method ESCK-full consistently get better accuracy than other matrix sketching methods. The other two data-dependent matrix sketching methods ESCK-minibatch and SRHT-topr also get better than the four data-oblivious matrix sketching method. When the parameter r is small, the accuracy improvement of ESCK-full is large on real-sim and rcv1-binary datasets.

Impact of Sparse Sketched Matrix By tuning the λ parameter ϵ-$\mathcal{L}_1$ ball projection, our proposed method can result in a very sparse sketched matrix $\widetilde{\mathbf{X}}$. In this section, we explore how the sparsity rate of the sketched matrix affects the classification accuracy. In Figure 3, we show the sparsity rate and accuracy for count-sketch and ESCK-full. The blue dashed line shows the accuracy of count-sketch and the sparsity rate is annotated by the text above this line. The red line shows the accuracies of ESCK-full with different sparsity rate of the sketched matrix. As shown in Figure 3, the ESCK-full obtain better accuracy than count-sketch with a higher sparsity rate. As the sparsity rate increased, we can observe that accuracy could slightly decrease but still higher than count-sketch. On the mnist dataset, the count-sketch method generates a dense sketched matrix with a sparsity rate equals to 1.72% and the accuracy of the subsequent classifier is 87.65%. In comparison, the ESCK-full can generate a sparse sketched matrix with higher classification accuracy.

We have compared the embedding time of our proposed method with the original count-sketch during the training stage. We also investigate the effect of the diagonal random sign matrix $\mathbf{D}$ in our algorithm. The detailed results and discussion can be found in supplementary materials.

Conclusion

In this paper, we propose a novel data-dependent count-sketch algorithm that can produce more effective and sparse subspace embedding than the original count-sketch algorithm. Our new method applies k-means clustering algorithm to obtain the sketched data matrix. A sparse sketched data matrix is obtained by using gradient descent with ϵ-$\mathcal{L}_1$ ball projection to optimize the k-means clustering objective function. We compared our proposed algorithm with the other five matrix sketching algorithms. Our experimental results on six real-life datasets have demonstrated that our proposed methods achieve higher classification accuracies than count-sketch and other matrix sketching methods. Also, our proposed methods can produce a sketched matrix with high sparsity rate than other methods that can make the subsequent classification model more efficient than others.

References

Achlioptas, D. 2003. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4): 671–687.

Ailon, N.; and Chazelle, B. 2006. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 557–563.

Ailon, N.; and Liberty, E. 2009. Fast dimension reduction using Rademacher series on dual BCH codes. *Discrete & Computational Geometry*, 42(4): 615.

Arora, S.; Hazan, E.; and Kale, S. 2006. A fast random sampling algorithm for sparsifying matrices. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 272–279. Springer.

Bojarski, M.; Choromanska, A.; Choromanski, K.; Fagan, F.; Gouy-Pailler, C.; Morvan, A.; Sakr, N.; Sarlos, T.; and Atif, J. 2017. Structured adaptive and random spinners for fast machine learning computations. In *Artificial Intelligence and Statistics*, 1020–1029.

Boutsidis, C.; Zouzias, A.; and Drineas, P. 2010. Random projections for k-means clustering. In *Advances in Neural Information Processing Systems*, 298–306.

Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3): 1–27.

Choromanski, K. M.; Rowland, M.; and Weller, A. 2017. The unreasonable effectiveness of structured random orthogonal embeddings. In *Advances in Neural Information Processing Systems*, 219–228.

Clarkson, K. L.; and Woodruff, D. P. 2017. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6): 1–45.

Dahiya, Y.; Konomis, D.; and Woodruff, D. P. 2018. An empirical evaluation of sketching for numerical linear algebra. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1292–1300.

Dasgupta, S.; and Gupta, A. 1999. An elementary proof of the Johnson-Lindenstrauss lemma. *International Computer Science Institute, Technical Report*, 22(1): 1–5.

Ding, C.; He, X.; and Simon, H. D. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*, 606–610. SIAM.

Dobriban, E.; and Liu, S. 2019. Asymptotics for sketching in least squares regression. In *Advances in Neural Information Processing Systems*, 3675–3685.

Duchi, J.; Shalev-Shwartz, S.; Singer, Y.; and Chandra, T. 2008. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, 272–279.

Johnson, W. B.; and Lindenstrauss, J. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206): 1.

Jolliffe, I. 2011. *Principal component analysis*. Springer.

Jolliffe, I. T.; and Cadima, J. 2016. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065): 20150202.

Lei, Z.; and Lan, L. 2020. Improved Subsampled Randomized Hadamard Transform for Linear SVM. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 4519–4526. AAAI Press.

Liu, W.; Shen, X.; and Tsang, I. 2017. Sparse Embedded k-Means Clustering. In *Advances in Neural Information Processing Systems*, 3319–3327.

Lloyd, S. 1982. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2): 129–137.

Mahoney, M. W. 2011. Randomized Algorithms for Matrices and Data. *Foundations and Trends® in Machine Learning*, 3(2): 123–224.

Martinsson, P.-G.; and Tropp, J. 2020. Randomized numerical linear algebra: Foundations & algorithms. *arXiv preprint arXiv:2002.01387*.

Paul, S.; Boutsidis, C.; Magdon-Ismail, M.; and Drineas, P. 2014. Random Projections for Linear Support Vector Machines. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(4): 1–25.

Sculley, D. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, 1177–1178.

Tropp, J. A. 2011. Improved analysis of the subsampled randomized Hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02): 115–126.

Tropp, J. A.; Yurtsever, A.; Udell, M.; and Cevher, V. 2017. Practical sketching algorithms for low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 38(4): 1454–1485.

Woodruff, D. P. 2014. Sketching as a Tool for Numerical Linear Algebra. *Theoretical Computer Science*, 10(1-2): 1–157.

Xu, Y.; Yang, H.; Zhang, L.; and Yang, T. 2017. Efficient non-oblivious randomized reduction for risk minimization with improved excess risk guarantee. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2796–2802.

Appendix

Proof of Proposition 1

Proposition 1 *The reconstruction error of count-sketch* $\|\mathbf{X} - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2$ *is equivalent to the objective function of applying* k*-means clustering on the columns of matrix product* $\mathbf{M} = \mathbf{XD}$ *if we treat* $\mathbf{\Phi}$ *as a learnable variable which denotes the cluster membership of each column in* $\mathbf{M}$*,*

$$\begin{aligned}&\|\mathbf{X} - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2 = \|\mathbf{M} - \mathbf{M\Phi S\Phi}^T\|_F^2 \\ &= \sum_{i=1}^{d} \|\mathbf{M}_{(:,i)} - \mathbf{c}_{I(\mathbf{M}_{(:,i)})}\|_2^2,\end{aligned} \tag{8}$$

where $\mathbf{M}_{(:,i)}$ *denotes the* i*-th column of* $\mathbf{M}$*,* $I(\mathbf{M}_{(:,i)})$ *returns the index of the cluster that* $\mathbf{M}_{(:,i)}$ *belongs to and* $\mathbf{c}_{I(\mathbf{M}_{(:,i)})}$ *is the centroid of that cluster.*

We first rewrite the reconstruction error as $\|\mathbf{X} - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2 = \|\mathbf{X} - \mathbf{XDD}^T + \mathbf{XDD}^T - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2$. Note that $\mathbf{D}$ is a $d \times d$ diagonal matrix with each diagonal entry either 1 or -1, therefore $\mathbf{X} = \mathbf{XDD}^T$. Let us use $\mathbf{M}$ to denote $\mathbf{XD}$, we will have

$$\begin{aligned}&\|\mathbf{X} - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2 \\ =&\|\mathbf{XDD}^T - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2 \\ =&\|\mathbf{MD}^T - \mathbf{M\Phi S\Phi}^T\mathbf{D}^T\|_F^2\end{aligned} \tag{9}$$

Next, we will show that $\|\mathbf{MD}^T - \mathbf{M\Phi S\Phi}^T\mathbf{D}^T\|_F^2 = \|\mathbf{M} - \mathbf{M\Phi S\Phi}^T\|_F^2$ as follows,

$$\begin{aligned}&\|\mathbf{MD}^T - \mathbf{M\Phi S\Phi}^T\mathbf{D}^T\|_F^2 \\ &= trace((\mathbf{MD}^T - \mathbf{M\Phi S\Phi}^T\mathbf{D}^T) \\ &\qquad\qquad (\mathbf{MD}^T - \mathbf{M\Phi S\Phi}^T\mathbf{D}^T)^T) \\ &= trace((\mathbf{M} - \mathbf{M\Phi S\Phi}^T)\mathbf{D}^T\mathbf{D}(\mathbf{M} - \mathbf{M\Phi S\Phi}^T)^T) \\ &= trace((\mathbf{M} - \mathbf{M\Phi S\Phi}^T)(\mathbf{M} - \mathbf{M\Phi S\Phi}^T)^T). \\ &= \|\mathbf{M} - \mathbf{M\Phi S\Phi}^T\|_F^2.\end{aligned} \tag{10}$$

Combining (9) and (10), the reconstruction error of count-sketch can be rewritten as

$$\|\mathbf{X} - \mathbf{XD\Phi S\Phi}^T\mathbf{D}^T\|_F^2 = \|\mathbf{M} - \mathbf{M\Phi S\Phi}^T\|_F^2 \tag{11}$$

Based on the definition of matrix $\mathbf{\Phi}$, $\mathbf{\Phi}$ is a $d \times r$ indicator matrix which each row has only one non-zero entry. Therefore, $\mathbf{\Phi}$ can viewed as a cluster membership indicator matrix which corresponds to randomly assign d columns of matrix $\mathbf{M}$ into r clusters. The non-zero element $\mathbf{\Phi}_{ij} = 1$ in i-th row of $\mathbf{\Phi}$ denotes the i-th column in $\mathbf{M}$ is assigned to cluster j. Note that the i-th column of matrix product $\mathbf{M\Phi S\Phi}^T$ is the centroid of the cluster where the i-th column $\mathbf{M}_{(:,i)}$ belongs to. Therefore

$$\|\mathbf{M} - \mathbf{M\Phi S\Phi}^T\|_F^2 = \sum_{i=1}^{d} \|\mathbf{M}_{(:,i)} - \mathbf{c}_{I(\mathbf{M}_{(:,i)})}\|_2^2, \tag{12}$$

where $I(\mathbf{M}_{(:,i)})$ returns the index of the cluster that the i-th column $\mathbf{M}_{(:,i)}$ belongs to and $\mathbf{c}_{I(\mathbf{M}_{(:,i)})}$ is the centroid of that cluster. By treating $\mathbf{\Phi}$ as a learnable variable which denotes the cluster membership, the reconstruction error of count-sketch is the same as the objective function of k-means algorithm on the columns of $\mathbf{M}$ as shown in (12).

Note that our proposition 1 is different to results in (Ding, He, and Simon 2005). (Ding, He, and Simon 2005) aim to explain the connection between non-negative matrix factorization and kernel k-means. However, in our paper, we aim to analyze the reconstruction error of count-sketch and using k-mean algorithm as a more effective column sampling methods

Additional Experimental Results

Embedding Time of Different Algorithms in the Training Stage We compare the embedding time of our proposed method with the original count-sketch during the training stage. The results are shown in Table 4. As expected, our proposed methods will be several times slower than the original count-sketch since we need to perform k-means clustering on the columns of $\mathbf{M}$. ESCK-miniBatch is faster than ESCK-full.

	embedding time during training stage		
	Count-Sketch	ESCK-full	ESCK-miniBatch
usps	1.2s	5.3s	1.1s
mnist	5.1s	26.4s	6.5s
gisette	2.6s	10.2s	4.5s
real-sim	2.6s	38.5s	13.5s
rcv1-bianry	1.9s	12.6s	5.3s
rcvl-multi	1.1s	8.5s	4.1s

Table 4: Comparison of the embedding during the training stage

The Effect of the Diagonal Random Sign Matrix $\mathbf{D}$

As mentioned in the paper, count-sketch (Clarkson and Woodruff 2017) constructs the random matrix $\mathbf{R} \in R^{d \times r}$ as $\mathbf{R} = \mathbf{D\Phi}$ where $\mathbf{D}$ is a $d \times d$ diagonal matrix with each diagonal entry independently chosen to be 1 or -1 with a probability of 0.5. Previous studies (Arora, Hazan, and Kale 2006; Clarkson and Woodruff 2017) have proved that the matrix $\mathbf{R}$ constructed in this way can guarantee with the property that the structure in the high dimension space can also approximately preserved in the randomly projected subspace. In the theoretical analysis of count-sketch (Clarkson and Woodruff 2017), the expectation of the pairwise distance between two samples in the projected subspace will not be equal to the pairwise distance in the input space without using $\mathbf{D}$. Here we empirically investigate the effect of the diagonal random sign matrix $\mathbf{D}$ in our method. Our empirical results are shown in Table 5. Our results show that the effect of $\mathbf{D}$ is very subtle in our method since our method is a deterministic sampling method while count-sketch is a random sampling method. We leave the theoretical analysis of the effect of $\mathbf{D}$ in our future work.

	Performance	usps	mnist	gisette	real-sim	rcv1-binary	rcv1-multi
		(r=30)	(r=100)	(r=256)	(r=256)	(r=256)	(r=256)
ESCK-full	Accuracy(%)	92.55± 0.02	90.60±0.02	95.13± 0.02	88.68±0.07	92.91±0.01	78.99 ± 0.01
	Sparsity rate	16.60%	43.10%	3.66%	89.57%	87.61%	88.44%
	Prediction time(ms)	1.5	15.8	2.5	4.0	1.0	13.5
ESCK-full without $\mathbf{D}$	Accuracy(%)	91.60± 0.02	90.45±0.01	95.06± 0.01	88.54±0.07	92.89±0.01	77.67 ± 0.02
	Sparsity rate	50.81%	56.37%	3.66%	90.60%	87.61%	88.49%
	Prediction time(ms)	0.9	12.3	2.4	4.0	1.2	13.9
ESCK-miniBatch	Accuracy(%)	92.18± 0.01	90.50±0.02	94.45 ± 0.03	88.25±0.08	90.01±0.02	77.13 ± 0.01
	Sparsity rate	46.78%	40.29%	37.58%	97.47%	94.78%	95.37%
	Prediction time(ms)	2.0	16.9	1.3	1.0	1.0	6.0
ESCK-miniBatch without $\mathbf{D}$	Accuracy(%)	91.90± 0.01	89.53±0.02	93.83 ± 0.03	88.04±0.08	90.30±0.02	77.61 ± 0.01
	Sparsity rate	3.33%	40.07%	0.08%	91.78%	91.96%	89.16%
	Prediction time(ms)	1.9	16.6	3.9	4.1	0.9	12.5

Table 5: Experimental results on the effect of diagonal random sign matrix $\mathbf{D}$