# Provable Variable Selection for Streaming Features

Jing Wang [1] Jie Shen [2] Ping Li [3]

## Abstract

In large-scale machine learning applications and high-dimensional statistics, it is ubiquitous to address a considerable number of features among which many are redundant. As a remedy, online feature selection has attracted increasing attention in recent years. It sequentially reveals features and evaluates the importance of them. Though online feature selection has proven an elegant methodology, it is usually challenging to carry out a rigorous theoretical characterization. In this work, we propose a provable online feature selection algorithm that utilizes the online leverage score. The selected features are then fed to $k$-means clustering, making the clustering step memory and computationally efficient. We prove that with high probability, performing $k$-means clustering based on the selected feature space does not deviate far from the optimal clustering using the original data. The empirical results on real-world data sets demonstrate the effectiveness of our algorithm.

## 1. Introduction

For retailers, brick-and-mortar stores and internet-based stores, various recommendation methods are proposed in an attempt to sell products. The recommendation model is usually updated in a timely manner or it includes new valuable features of products which are not previously available. For example, during the Apple WWDC 2018 keynote, Apple has introduced new features of their platforms to fight "fingerprinting", a technique which tracks users based on identifying computers. With the available of new features, a feature selection model is employed to determine whether the new features will drive sales of products in the future and only related features will be included in the recommendation model. Hence, in real-world applications, features are usually revealed in a continuous stream. It is necessary to evaluate new features immediately and output intermediate result. The feature evaluation process in a stream is called online feature selection (Perkins & Theiler, 2003; Zhou et al., 2005; Wu et al., 2010). We first formulate this problem.

Suppose that there are $n$ samples but initially we do not observe all of the features. We call the sequence $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots \in \mathbb{R}^n$ is a feature stream, with each $\boldsymbol{a}_i \in \mathbb{R}^d$ being the $i$th feature, or the $i$th covariate of all samples. Note that in our setting, the feature $\boldsymbol{a}_i$ is revealed at time stamp $i$. If $\boldsymbol{a}_i$ is selected, we update the observation matrix $\boldsymbol{A}$ as follows:

$$\boldsymbol{A} \leftarrow [\boldsymbol{A} \quad \theta_i \boldsymbol{a}_i]. \tag{1.1}$$

where the parameter $\theta_i \neq 0$ is chosen in an online manner.

In the literature, a large number of online methods have been proposed based on statistical measurements or optimization techniques (Perkins & Theiler, 2003; Zhou et al., 2005; Wang et al., 2015). For example, Perkins & Theiler (2003) added a new feature which contributes to a predictor learning and optimization analysis into the model. Zhou et al. (2005) proposed an adaptive complexity penalty method to evaluate a new feature based on its $p$-value. Wu et al. (2010) utilized the Markov blanket to measure the relationship between a new feature and the selected feature subset. Yet successful, most of the results in this line of research are empirical in nature.

On the other hand, feature selection method can be categorized into either supervised or unsupervised. For instance, Shen & Li (2017) recently proposed a non-convex supervised approach for variable selection with favorable iteration complexity. Unsupervised methods, however, are with great practical importance to many areas such as Cardiology, as annotated data is usually precious and limited due to genetic privacy issue and the medical background requirement for annotators.

**Summary of Contributions**. In this paper, we consider the high-dimensional regime that the number of features is much larger than the sample size, and the features are revealed in an online manner. We propose an unsupervised algorithm termed Online leverage scores for Feature Selection

[1]Cornell University, New York, NY 10021, USA. [2]Rutgers University, Piscataway, NJ 08854, USA. [3]Baidu Research, Bellevue, WA 98004, USA. Jing Wang <jiw2026@med.cornell.edu>, Ping Li <pingli98@gmail.com>. Correspondence to: Jie Shen <js2007@rutgers.edu>.

*Proceedings of the $35^{th}$ International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

(OFS). Our main technique is to approximately compute the broadly used leverage score in each iteration, and determine the importance of each feature in real time. We prove that the reduced feature space is a good approximation to the original one in some sense to be clarified. Furthermore, we apply $k$-means clustering on the set of selected features, and show that the clustering performance does not degrade a lot. Computationally, our algorithm enjoys low time complexity and little memory usage, which makes it a perfect fit for big data analytics.

## 2. Related Work

Feature selection is a primary technique in machine learning to address "the curse of dimensionality". In the last decades, a number of methods have been proposed (Guyon & Elisseeff, 2003; Donoho & Jin, 2008). In this section, we give a brief review of existing approaches in terms of batch situation and online situation.

**Batch Methods.** Existing batch feature selection methods can be roughly divided to unsupervised, supervised and semi-supervised approaches. The supervised methods utilize the target variable to guide the feature evaluation process, such as Fisher score, Least Absolute Shrinkage and Selection Operator (Lasso) (Tibshirani, 1996) and minimum Redundancy Maximum Relevance (Peng et al., 2005). Unsupervised feature selection methods mainly depend on latent data distribution analysis (He et al., 2005), such as spectral analysis (Zhao & Liu, 2007; Cai et al., 2010) and Kullback-Leibler Divergence between neighborhood distributions (Wei & Philip, 2016). The semi-supervised feature selection algorithms make benefits of both aforementioned approaches, such as combining Gaussian Field and Harmonic functions (Kong & Yu, 2010; Zhu et al., 2003).

Feature selection methods are also characterized as wrapper, embedded and filter model. The wrapper model evaluates feature subsets by their performance on a specific algorithm, such as SVM or Naive Bayes for classification tasks (Forman, 2003) and $k$-means for clustering tasks (Guyon et al., 2002; Xu et al., 2014). The embedded model seeks the desired feature subset by solving a regularized optimization objective function with certain constraints (Zhang, 2009; Yang & Xu, 2013). Examples of this approach include Least Angle Regression (Efron et al., 2004) and group Lasso (Zhang et al., 2016). The optimization process forces most coefficients small or exact zero. The features corresponding to nonzero coefficients are selected.

The filter model utilizes certain statistical measurements, such as the Hilbert-Schmidt Independence Criterion (HSIC), leverage score (Boutsidis et al., 2009) and kernel-based measures of independence (Chen et al., 2017). Specifically, the statistical leverage score is an important measurement for unsupervised feature selection. It characterizes the outstanding features that have more affect towards the result of a statistical procedure. There are multiple variants of the statistical leverage score, such as the normalized leverage score (Boutsidis et al., 2009), the truncated version of leverage score (Gittens & Mahoney, 2013) and the kernel ridge leverage score (Alaoui & Mahoney, 2015). The ridge leverage score is used to select features for $k$-means clustering (Boutsidis et al., 2009) and has proved to attain $(2+\epsilon)$-approximate partition. Specifically, the ridge leverage score of the $i$th column of data matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ is defined as (Alaoui & Mahoney, 2015),

$$l_i = \boldsymbol{a}_i^\top (\boldsymbol{A}\boldsymbol{A}^\top + \lambda \boldsymbol{I})^{-1}\boldsymbol{a}_i, \tag{2.1}$$

where $\lambda > 0$ is a parameter, $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ is the identity matrix. However, it is expensive as it requires $\mathcal{O}\left(n^3 + n^2 d\right)$ running time and $\mathcal{O}\left(nd\right)$ memory storage. A number of recent papers focus on sampling some columns of $\boldsymbol{A}$ and approximate the linear kernel of $\boldsymbol{A}$ (Li et al., 2013; Alaoui & Mahoney, 2015; Cohen et al., 2016; Musco & Musco, 2017). However, none of these techniques have been applied for feature selection of streaming features.

**Online Methods.** Motivated by the fact that features are available in a stream in real-world applications, online feature selection has attracted a lot of attention (Perkins & Theiler, 2003; Zhou et al., 2005; Wu et al., 2010; Wang et al., 2013). The batch-mode algorithms cannot handle this situation well as the global feature space is required in advance. Examples of online feature selection approaches either utilize statistical measurements, such as alpha-investing (Zhou et al., 2005) and mutual information (Wu et al., 2010) or rely on optimization techniques, such as stochastic gradient grafting (Perkins & Theiler, 2003; Wang et al., 2015). All existing mentioned methods come with no theoretical guarantees of the selected feature subset for clustering task.

### 2.1. Notation

We use bold lower-case letters, e.g. $\boldsymbol{v} \in \mathbb{R}^d$ to denote a column vector. $\|\boldsymbol{v}\|_2$ is used to denote the $\ell_2$-norm of the vector. Capital letters such as $\boldsymbol{X}$ are used to denote matrices, and its transpose is denoted by $\boldsymbol{X}^\top$. The capital letter $\boldsymbol{I}_{n\times n}$ is reserved for the identity matrix where $n$ indicates its size. For an invertible matrix $\boldsymbol{X}$, we write its inverse as $\boldsymbol{X}^{-1}$. Otherwise, we use $\boldsymbol{X}^\dagger$ for the pseudoinverse. For a square matrix $\boldsymbol{X}$, we write its trace as $\mathrm{Tr}\left(\boldsymbol{X}\right)$, which is the sum of its diagonal elements. The $i$th column and $j$th row of the matrix $\boldsymbol{X}$ are denoted by $\boldsymbol{x}_i$ and $(\boldsymbol{x}^j)^\top$, respectively. Suppose that the rank of matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ is $k \leq \min\{m, n\}$. The singular value decomposition of $\boldsymbol{X}$

is given by

$$\boldsymbol{X} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_k] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^\top \\ \vdots \\ \boldsymbol{v}_k^\top \end{bmatrix}$$

where the singular values in descending order $\sigma_1 \geq \cdots \geq \sigma_k > 0$, $\boldsymbol{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_k] \in \mathbb{R}^{n \times k}$ contains the left singular vectors and $\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k]$ contains the right singular vectors. In this paper, we will use the Frobenius norm $\|\boldsymbol{X}\|_F := \sqrt{\sum_{i=1}^k \sigma_i^2}$ and the spectral norm $\|\boldsymbol{X}\|_2 := \max_{1 \leq i \leq k} \sigma_i = \sigma_1$.

For a sequence of random variables $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$, we write $\mathbf{E}_{j-1}[\boldsymbol{X}_j]$ for the expectation of $\boldsymbol{X}_j$ conditioning on $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{j-1}\}$.

## 3. Main Results

In this section, we propose an online algorithm for feature selection, where the goal is to approximate the original data with much fewer attributes in some sense. To the end, we make use of the leverage score that, from a high level, reflects the importance of each feature.

Suppose that the data matrix is $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, i.e., $n$ samples lying in a $d$-dimensional ambient space. The statistical leverage score of the $i$th column (i.e., feature) of $\boldsymbol{A}$ is defined as

$$l_i^* = \boldsymbol{a}_i^\top (\boldsymbol{A}\boldsymbol{A}^\top)^\dagger \boldsymbol{a}_i. \tag{3.1}$$

It is well known that sampling an $n \times \mathcal{O}\left(\epsilon^{-2} n \log n\right)$ matrix $\tilde{\boldsymbol{A}}$ with probabilities proportional to the respective leverage scores yields a $(1+\epsilon)$-spectral approximation to $\boldsymbol{A}$ (Spielman & Srivastava, 2011), in the sense that for all $\boldsymbol{x}$

$$\left\|\tilde{\boldsymbol{A}}^\top \boldsymbol{x}\right\|_2 \approx \left\|\boldsymbol{A}^\top \boldsymbol{x}\right\|_2, \text{ or more precisely}$$
$$(1-\epsilon)\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x} \leq \boldsymbol{x}^\top \tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top \boldsymbol{x} \leq (1+\epsilon)\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{x}, \text{ or}$$
$$(1-\epsilon)\boldsymbol{A}\boldsymbol{A}^\top \preceq \tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top \preceq (1+\epsilon)\boldsymbol{A}\boldsymbol{A}^\top.$$

In the online setting, however, we are not able to access all the data to compute the leverage score. The key idea of our algorithm is that when a new feature arrives, we approximate its leverage score based on the obtained features, which can further be used to guide the selection process.

To be more concrete, at time stamp $i$, suppose the observed data matrix is $\tilde{\boldsymbol{A}}_{i-1}$ and the new feature $\boldsymbol{a}_i$ is revealed, we need to determine whether $\boldsymbol{a}_i$ is kept or discarded. A natural way for the sake is to compute the approximate leverage score of $\boldsymbol{a}_i$ as follows:

$$l_i = \boldsymbol{a}_i^\top (\tilde{\boldsymbol{A}}_{i-1}\tilde{\boldsymbol{A}}_{i-1}^\top)^\dagger \boldsymbol{a}_i. \tag{3.2}$$

**Algorithm 1** Online Feature Selection

```
Require: Initial data matrix $\tilde{\boldsymbol{A}}_0$, sampling rate $c = 8\epsilon^{-2}\log n$, approximation parameter $\epsilon \in (0,1)$.
1: for $i = 1, \cdots$ do
2:     Reveal the $i$th feature $\boldsymbol{a}_i$.
3:     Compute the online leverage score
           $\tilde{l}_i = \min((1+\epsilon)\boldsymbol{a}_i^\top(\tilde{\boldsymbol{A}}_{i-1}\tilde{\boldsymbol{A}}_{i-1}^\top)^\dagger \boldsymbol{a}_i, 1)$.
4:     Compute the probability,
           $p_i = \min(c\tilde{l}_i, 1)$.
5:     With probability $p_i$, update
           $\tilde{\boldsymbol{A}}_i = [\tilde{\boldsymbol{A}}_{i-1} \quad \boldsymbol{a}_i/\sqrt{p_i}]$.
       Otherwise,
           $\tilde{\boldsymbol{A}}_i = \tilde{\boldsymbol{A}}_{i-1}$.
6: end for
```

Intuitively, if $\tilde{\boldsymbol{A}}_{i-1}$ is a good approximation to $\boldsymbol{A}$, $l_i$ indicates the importance of $\boldsymbol{a}_i$ as $l_i^*$ does. And what we will show is that, it is the case after we reveal a few attributes.

It is known that if the entire feature space is available, each leverage score is upper bounded by 1. However the estimates based on $\tilde{\boldsymbol{A}}_{i-1}$ can be arbitrary because $\tilde{\boldsymbol{A}}_{i-1}$ is a submatrix of $\boldsymbol{A}$ which leads to $\tilde{\boldsymbol{A}}_{i-1}\tilde{\boldsymbol{A}}_{i-1}^\top \preceq \boldsymbol{A}\boldsymbol{A}^\top$. For our analysis, we technically require that each $l_i$ is not larger than 1. Hence, we will make use of a modified quantity

$$\tilde{l}_i = \min\left((1+\epsilon)\boldsymbol{a}_i^\top (\tilde{\boldsymbol{A}}_{i-1}\tilde{\boldsymbol{A}}_{i-1}^\top)^\dagger \boldsymbol{a}_i, 1\right). \tag{3.3}$$

Note that $\epsilon > 0$ is some pre-defined accuracy parameter, and the above suggests we are using a conservative estimate of the leverage score. To see this, consider $\tilde{\boldsymbol{A}}_{i-1} = \boldsymbol{A}$, then $\tilde{l}_i \geq l_i^*$. It is essential in the online setting in that we may lose many important features with an aggressive strategy.

Then, the sampling probability is computed as

$$p_i = \min\left(8\epsilon^{-2}\log n \cdot \tilde{l}_i, 1\right). \tag{3.4}$$

With the scaling factor of $\tilde{l}_i$ above, it is not hard to see that for a small approximation error $\epsilon$, one has to select the current feature with high probability, which conforms the intuition – an exact estimation of $\boldsymbol{A}$ requires selecting all the features. We summarize our method in Algorithm 1.

### 3.1. Analysis

We first show that with high probability, the data matrix produced by our algorithm is a good approximation to $\boldsymbol{A}$.

**Theorem 1.** *Consider Algorithm 1. Let $\tilde{\boldsymbol{A}}$ be the output when it terminates. It holds with high probability that*

$$(1-\epsilon)\boldsymbol{A}\boldsymbol{A}^\top \preceq \tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top \preceq (1+\epsilon)\boldsymbol{A}\boldsymbol{A}^\top.$$

*Proof.* Let $\boldsymbol{A}_i = (\boldsymbol{a}_1\ \boldsymbol{a}_2 \dots \boldsymbol{a}_i)$. Define $\boldsymbol{Y}_0$ as the zero matrix and for all $i \geq 1$, let

$$\boldsymbol{Y}_{i-1} = (\boldsymbol{A}\boldsymbol{A}^\top)^{\dagger/2}(\tilde{\boldsymbol{A}}_{i-1}\tilde{\boldsymbol{A}}_{i-1}^\top - \boldsymbol{A}_{i-1}\boldsymbol{A}_{i-1}^\top)(\boldsymbol{A}\boldsymbol{A}^\top)^{\dagger/2}.$$

Let $\boldsymbol{u}_i = (\boldsymbol{A}\boldsymbol{A}^\top)^{\dagger/2}\boldsymbol{a}_i$. If $\|\boldsymbol{Y}_{i-1}\|_2 \geq \epsilon$, we set $\boldsymbol{X}_i = \boldsymbol{0}$. Otherwise, set

$$\boldsymbol{X}_i = \begin{cases} (1/p_i - 1)\boldsymbol{u}_i\boldsymbol{u}_i^\top, & \text{if } \boldsymbol{a}_i \text{ is sampled in } \tilde{\boldsymbol{A}}, \\ -\boldsymbol{u}_i\boldsymbol{u}_i^\top, & \text{otherwise.} \end{cases}$$

Thus, $\boldsymbol{X}_i = \boldsymbol{Y}_i - \boldsymbol{Y}_{i-1}$.

Consider the case $\|\boldsymbol{Y}_{i-1}\|_2 < \epsilon$. We get

$$\begin{aligned}
\tilde{l}_i &= \min((1+\epsilon)\boldsymbol{a}_i^\top(\tilde{\boldsymbol{A}}_i\tilde{\boldsymbol{A}}_i^\top)^\dagger\boldsymbol{a}_i, 1) \\
&\geq \min((1+\epsilon)\boldsymbol{a}_i^\top(\boldsymbol{A}_i\boldsymbol{A}_i^\top + \epsilon\boldsymbol{A}\boldsymbol{A}^\top)^\dagger\boldsymbol{a}_i, 1) \\
&\geq \min((1+\epsilon)\boldsymbol{a}_i^\top((1+\epsilon)(\boldsymbol{A}\boldsymbol{A}^\top))^\dagger\boldsymbol{a}_i, 1) \\
&= \boldsymbol{a}_i^\top(\boldsymbol{A}\boldsymbol{A}^\top)^\dagger\boldsymbol{a}_i \\
&= \boldsymbol{u}_i^\top\boldsymbol{u}_i.
\end{aligned}$$

Thus, $p_i \geq \min(c\boldsymbol{u}_i^\top\boldsymbol{u}_i, 1)$. If $p_i = 1$, then $\boldsymbol{X}_i = \boldsymbol{0}$. Otherwise, we have $p_i \geq c\boldsymbol{u}_i^\top\boldsymbol{u}_i$. Moreover, we get

$$\|\boldsymbol{X}_i\|_2 \leq 1/c$$

and

$$\begin{aligned}
&\mathbf{E}_{i-1}\left[\boldsymbol{X}_i^2\right] \\
&\preceq p_i \cdot (1/p_i - 1)^2(\boldsymbol{u}_i\boldsymbol{u}_i^\top)^2 + (1-p_i)\cdot(\boldsymbol{u}_i\boldsymbol{u}_i^\top)^2 \\
&= (\boldsymbol{u}_i\boldsymbol{u}_i^\top)^2/p_i \\
&\preceq \boldsymbol{u}_i\boldsymbol{u}_i^\top/c.
\end{aligned}$$

Let $\boldsymbol{W}_i = \sum_{k=1}^i \mathbf{E}_{k-1}\left[\boldsymbol{X}_k^2\right]$. We have

$$\|\boldsymbol{W}_i\|_2 \leq \left\|\sum_{k=1}^i \boldsymbol{u}_i\boldsymbol{u}_i^\top/c\right\|_2 \leq 1/c.$$

Applying Lemma 4 gives

$$\begin{aligned}
\Pr(\|\boldsymbol{Y}_n\|_2 \geq \epsilon) &\leq n\cdot\exp\left(\frac{-\epsilon^2/2}{1/c + \epsilon/(3c)}\right) \\
&\leq n\cdot\exp(-c\epsilon^2/4) \\
&= 1/n.
\end{aligned}$$

This implies that with high probability

$$\left\|(\boldsymbol{A}\boldsymbol{A}^\top)^{\dagger/2}(\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top)(\boldsymbol{A}\boldsymbol{A}^\top)^{\dagger/2} - \boldsymbol{I}\right\|_2 \leq \epsilon.$$

We thus have

$$(1-\epsilon)\boldsymbol{A}\boldsymbol{A}^\top \preceq \tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top \preceq (1+\epsilon)\boldsymbol{A}\boldsymbol{A}^\top,$$

completing the theorem. □

Now we turn to control the number of features selected by Algorithm 1. We will use the result in (Cohen et al., 2015) shown below.

**Lemma 1.** *Let $\boldsymbol{A}$ be an $n\times d$ matrix, $\epsilon \in (0,1)$, $c = 1/\epsilon$, $\tilde{l}_1, \cdots, \tilde{l}_d$ be over-estimated leverage scores, i.e., $\tilde{l}_i \geq \boldsymbol{a}_i^\top(\boldsymbol{A}\boldsymbol{A}^\top)^\dagger\boldsymbol{a}_i$ for all $1 \leq i \leq d$. Let $p_i = \min\{c\tilde{l}_i, 1\}$. Construct $\tilde{\boldsymbol{A}}$ by independently sampling each column $\boldsymbol{a}_i$ of $\boldsymbol{A}$ with probability $p_i$ and rescale it by $1/\sqrt{p_i}$ if it is included in $\tilde{\boldsymbol{A}}$. Then, with high probability, $\tilde{\boldsymbol{A}}$ is the $(1+\epsilon)$-spectral approximation of $\boldsymbol{A}$ and the number of columns in $\tilde{\boldsymbol{A}}$ is $\mathcal{O}\left(\epsilon^{-2}\sum_{i=1}^d \tilde{l}_i \log n\right)$.*

By Lemma 1, in order to control the number of selected features, we need to bound the sum of online leverage scores.

**Lemma 2.** *After running Algorithm 1, it holds with high probability that*

$$\sum_{i=1}^d \tilde{l}_i = \mathcal{O}\left(n\log(\|\boldsymbol{A}\|_2)\right).$$

*Proof.* We define

$$\delta_i = \log\det(\tilde{\boldsymbol{A}}_i\tilde{\boldsymbol{A}}_i^\top) - \log\det(\tilde{\boldsymbol{A}}_{i-1}\tilde{\boldsymbol{A}}_{i-1}^\top).$$

The sum of $\delta_i$ can be bounded by the logarithm of the ratio of the determinants of $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top$. By the matrix determinant lemma, we have

$$\begin{aligned}
&\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right] \\
&= p_i\cdot e^{l_i/8}(1 + \boldsymbol{a}_i^\top(\tilde{\boldsymbol{A}}_{i-1}\tilde{\boldsymbol{A}}_{i-1}^\top)^{-1}\boldsymbol{a}_i/p_i)^{-1} \\
&\quad + (1-p_i)\cdot e^{l_i/8} \\
&\leq (1 + l_i/4)\cdot(p_i(1 + \boldsymbol{a}_i^\top(\tilde{\boldsymbol{A}}_{i-1}\tilde{\boldsymbol{A}}_{i-1}^\top)^{-1}\boldsymbol{a}_i/p_i)^{-1} \\
&\quad + 1 - p_i).
\end{aligned}$$

If $c\tilde{l}_i < 1$, we have $p_i = c\tilde{l}_i$ and

$$\begin{aligned}
&\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right] \\
&\leq c\tilde{l}_i\cdot(1 + l_i/4)(1 + 1/((1+\epsilon)c))^{-1} + (1 - c\tilde{l}_i)\cdot \\
&\quad (1 + l_i/4) \\
&= (1 + l_i/4)(cl_i(1 + 1/((1+\epsilon)c))^{-1} + 1 - cl_i) \\
&\leq (1 + l_i/4)(1 + cl_i(1 - 1/(4c) - 1)) \\
&= (1 + l_i/4)(1 - l_i/4) \leq 1.
\end{aligned}$$

Otherwise, $p_i = 1$ and we have:

$$\begin{aligned}
&\mathbf{E}_{i-1}\left[\exp(\tilde{l}_i/8 - \delta_i)\right] \\
&\leq (1 + l_i/4)(1 + \boldsymbol{A}_i^\top(\tilde{\boldsymbol{A}}_{i-1}^\top \tilde{\boldsymbol{A}}_{i-1} + \lambda\mathbf{I})^{-1}\boldsymbol{A}_i)^{-1} \\
&\leq (1 + l_i/4)(1 + l_i)^{-1} \leq 1.
\end{aligned}$$

We now analyze the expected product of $\exp(\tilde{l}_i/8 - \delta_i)$ over the first $k$ steps. For $k \geq 1$ we have

$$\mathbf{E}\left[\exp\left(\sum_{i=1}^{k}\tilde{l}_i/8 - \delta_i\right)\right] \leq \mathbf{E}\left[\exp\left(\sum_{i=1}^{k-1}\tilde{l}_i/8 - \delta_i\right)\right],$$

and so by induction on $k$

$$\mathbf{E}\left[\exp\left(\sum_{i=1}^{d}\tilde{l}_i/8 - \delta_i\right)\right] \leq 1.$$

Hence by Markov's inequality

$$\Pr\left(\sum_{i=1}^{d}\tilde{l}_i > 8n + 8\sum_{i=1}^{d}\delta_i\right) \leq e^{-n}.$$

Using Theorem 1, with high probability, we have

$$\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top \preceq (1+\epsilon)\boldsymbol{A}\boldsymbol{A}^\top,$$

implying that

$$\begin{aligned}
\det(\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top) &\leq (1+\epsilon)^n(\|\boldsymbol{A}\|_2^2)^n, \\
\log\det(\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top) &\leq n(1 + \log(\|\boldsymbol{A}\|_2^2)).
\end{aligned}$$

By the definition of $\delta_i$, it holds with high probability that

$$\begin{aligned}
\sum_{i=1}^{d}\delta_i &= \log\det(\tilde{\boldsymbol{A}}^\top\tilde{\boldsymbol{A}} + \lambda\mathbf{I}) - n \\
&\leq n(1 + \log(\|\boldsymbol{A}\|_2^2) - 1) \\
&= n(\log(\|\boldsymbol{A}\|_2^2)).
\end{aligned}$$

And with high probability,

$$\begin{aligned}
\sum_{i=1}^{d}\tilde{l}_i &\leq 8n + 8\sum_{i=1}^{d}\delta_i \\
&\leq 8n + 8n\log(\|\boldsymbol{A}\|_2^2) \\
&= \mathcal{O}\left(n\log(\|\boldsymbol{A}\|_2^2)\right) \\
&= \mathcal{O}\left(n\log(\|\boldsymbol{A}\|_2)\right).
\end{aligned}$$

The proof is complete. □

Thus Lemma 1 and 2 imply that Algorithm 1 selects $\mathcal{O}\left(\epsilon^{-2}n\log d\log(\|\boldsymbol{A}\|_2)\right)$ columns with high probability.

**Time Complexity.** The running time of Algorithm 1 is dominated by the online leverage score computation in Step 3, which is $\mathcal{O}\left(n^3\right)$. In the case that $\tilde{\boldsymbol{A}}_{i-1}$ is a Laplacian matrix, Step 3 can be implemented in $\mathcal{O}\left(d\log^2 n\right)$ time by a fast graph Laplacian solver with the Johnson-Lindenstrauss lemma, as stated in (Koutis et al., 2016).

**Memory Cost.** The memory cost for leverage score computation is significantly reduced from $\mathcal{O}\left(nd\right)$ to $\mathcal{O}\left(n^2\log n\right)$ (storage of $\tilde{\boldsymbol{A}}_i$). This follows from the analysis of Lemma 2 which states that when the algorithm terminates, only $\mathcal{O}\left(\epsilon^{-2}n\log n\log(\|\boldsymbol{A}\|_2)\right)$ features will be selected. Note that this paper considers the regime where $n \ll d$, such as the number of patients with rare diseases $n$ and the length of their gene expressions $d$, or the batch size in neural networks $n$ and the corresponding dimension of feature space $d$. Hence our online implementation is order of magnitude more efficient. It leads to practical values of our algorithm for learning tasks, such as clustering.

## 3.2. Application to $k$-Means Clustering

We explore the performance of matrix $\tilde{\boldsymbol{A}}$ returned by Algorithm 1 when it is used for $k$-means clustering. We first recall the $k$-means clustering problem.

Formally, $k$-means clustering seeks to partition the data matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ into $k$ clusters $\{C_1, \cdots, C_k\}$ to minimize the distance between data points and its closest center $\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k\}$ (Awasthi et al., 2010):

$$\min_{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k}\sum_{i=1}^{k}\sum_{j\in C_i}\left\|\boldsymbol{a}^j - \boldsymbol{\mu}_i\right\|_2^2 \tag{3.5}$$

where $\boldsymbol{\mu}_i$ be the center of data points in $C_i$. It is known that $k$-means clustering is an instance of low-rank approximation (Boutsidis et al., 2009). To see this, we construct an $n\times k$ matrix $\boldsymbol{X}$ as the cluster indicator matrix. Then for each solution $\{\boldsymbol{\mu}_i\}_{i=1}^k$ of (3.5), we will assign a cluster label, say $\boldsymbol{x}_j \in \{1, 2, \ldots, k\}$, to each sample $\boldsymbol{a}^j$. We set $\boldsymbol{X}_{ij} = 1/\sqrt{|C_j|}$ if $\boldsymbol{a}_i$ belongs to $C_j$, and 0 otherwise. In this way, the $i$th row of $\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{A}$ is actually the average of the points with label $i$, i.e., the center $\boldsymbol{\mu}_i$ of the $i$th class. Hence, from the discussion, we may rewrite (3.5) as follows:

$$\min_{\boldsymbol{X}}\sum_{i=1}^{k}\sum_{j\in C_i}\left\|\boldsymbol{a}^j - (\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{A})^i\right\|_2^2.$$

More compactly, we aim to solve

$$\min_{\boldsymbol{X}}\left\|\boldsymbol{A} - \boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{A}\right\|_F^2.$$

See (Ostrovsky et al., 2006) for a more detailed discussion.

Let the indicator matrix $\boldsymbol{X}_* \in \mathbb{R}^{n\times k}$ denote the optimal partition on $\boldsymbol{A}$, i.e.,

$$\boldsymbol{X}_* = \arg\min_{\boldsymbol{X}} \left\| \boldsymbol{A} - \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{A} \right\|_F^2 . \tag{3.6}$$

We first investigate how the cluster indicator matrix $\boldsymbol{X}$ over $\tilde{\boldsymbol{A}}$ is deviated from the optimum. The following lemma provides the bound of the $k$-means objective function value on $\tilde{\boldsymbol{A}}$.

**Lemma 3.** *Suppose that $\tilde{\boldsymbol{A}}$ is the matrix returned by Algorithm 1, then*

$$\begin{aligned}(1-\epsilon)\left\| \boldsymbol{A} - \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{A} \right\|_F^2 &\le \left\| \tilde{\boldsymbol{A}} - \boldsymbol{X}\boldsymbol{X}^\top \tilde{\boldsymbol{A}} \right\|_F^2 \\ &\le (1+\epsilon)\left\| \boldsymbol{A} - \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{A} \right\|_F^2 ,\end{aligned}$$

*when $\epsilon$ is the parameter of Algorithm 1.*

*Proof.* Using the notation $\boldsymbol{Y} = \boldsymbol{I} - \boldsymbol{X}\boldsymbol{X}^\top$, we can rewrite the objective function of $k$-means based on the data matrices $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ as

$$\begin{aligned}\left\| \boldsymbol{A} - \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{A} \right\|_F^2 &= \|\boldsymbol{Y}\boldsymbol{A}\|_F^2 = \mathrm{Tr}\left(\boldsymbol{Y}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{Y}\right), \\ \left\| \tilde{\boldsymbol{A}} - \boldsymbol{X}\boldsymbol{X}^\top \tilde{\boldsymbol{A}} \right\|_F^2 &= \left\|\boldsymbol{Y}\tilde{\boldsymbol{A}}\right\|_F^2 = \mathrm{Tr}\left(\boldsymbol{Y}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top\boldsymbol{Y}\right).\end{aligned}$$

Note that

$$\mathrm{Tr}\left(\boldsymbol{Y}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top\boldsymbol{Y}\right) = \mathrm{Tr}\left(\sum_{i=1}^n \boldsymbol{y}_i^\top \tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top \boldsymbol{y}_i\right),$$

where $\boldsymbol{y}_i$ is the $i$th column of $\boldsymbol{Y}$. Then by the spectral bound on $\boldsymbol{A}\boldsymbol{A}^\top$ in Theorem 1, we immediately get

$$\begin{aligned}(1-\epsilon)\mathrm{Tr}\left(\boldsymbol{Y}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{Y}\right) &\le \mathrm{Tr}\left(\boldsymbol{Y}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^\top\boldsymbol{Y}\right) \\ &\le (1+\epsilon)\mathrm{Tr}\left(\boldsymbol{Y}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{Y}\right).\end{aligned}$$

Plugging $\boldsymbol{Y} = \boldsymbol{I} - \boldsymbol{X}\boldsymbol{X}^\top$ into the above inequalities completes the proof. □

Now we show that $\tilde{\boldsymbol{A}}$ is also a good approximation to $\boldsymbol{A}$.

**Theorem 2.** *Suppose that $\tilde{\boldsymbol{A}}$ is returned by Algorithm 1. Let $\tilde{\boldsymbol{X}}_* = \arg\min \left\| \tilde{\boldsymbol{A}} - \boldsymbol{X}\boldsymbol{X}^\top \tilde{\boldsymbol{A}} \right\|_F^2$. Then given $\epsilon \in (0,1)$, we can get*

$$\left\| \boldsymbol{A} - \tilde{\boldsymbol{X}}_*\tilde{\boldsymbol{X}}_*^\top \boldsymbol{A} \right\|_F^2 \le \frac{1+\epsilon}{1-\epsilon}\cdot \left\| \boldsymbol{A} - \boldsymbol{X}_*\boldsymbol{X}_*^\top \boldsymbol{A} \right\|_F^2 .$$

*Proof.* Using Lemma 3, we have

$$\begin{aligned}(1-\epsilon)\left\| \boldsymbol{A} - \tilde{\boldsymbol{X}}_*\tilde{\boldsymbol{X}}_*^\top \boldsymbol{A} \right\|_F^2 &\le \left\| \tilde{\boldsymbol{A}} - \tilde{\boldsymbol{X}}_*\tilde{\boldsymbol{X}}_*^\top \tilde{\boldsymbol{A}} \right\|_F^2 \\ \left\| \tilde{\boldsymbol{A}} - \boldsymbol{X}_*\boldsymbol{X}_*^\top \tilde{\boldsymbol{A}} \right\|_F^2 &\le (1+\epsilon)\left\| \boldsymbol{A} - \boldsymbol{X}_*\boldsymbol{X}_*^\top \boldsymbol{A} \right\|_F^2\end{aligned}$$

On the other hand, by the optimality of $\tilde{\boldsymbol{X}}_*$ for $\tilde{\boldsymbol{A}}$, we have

$$\left\| \tilde{\boldsymbol{A}} - \tilde{\boldsymbol{X}}_*\tilde{\boldsymbol{X}}_*^\top \tilde{\boldsymbol{A}} \right\|_F^2 \le \left\| \tilde{\boldsymbol{A}} - \boldsymbol{X}_*\boldsymbol{X}_*^\top \tilde{\boldsymbol{A}} \right\|_F^2$$

Combining the above inequalities, we have

$$\left\| \boldsymbol{A} - \tilde{\boldsymbol{X}}_*\tilde{\boldsymbol{X}}_*^\top \boldsymbol{A} \right\|_F^2 \le \frac{1+\epsilon}{1-\epsilon}\cdot \left\| \boldsymbol{A} - \boldsymbol{X}_*\boldsymbol{X}_*^\top \boldsymbol{A} \right\|_F^2 .$$

The proof is complete. □

Theorem 2 implies that if $\tilde{\boldsymbol{X}}_*$ is an optimal solution for $\tilde{\boldsymbol{A}}$, then it also preserves an $(1+\epsilon)$-approximation for $\boldsymbol{A}$. We compare our algorithm with existing dimension reduction methods for $k$-means clustering as shown in Table 1.

## 4. Experiments

This section describes an empirical study of the efficacy and efficiency of our algorithm. We first elaborate the experimental settings.

**Data Sets.** We perform the experiments on 6 realistic data sets, including USPS[1], AR[2], COIL20[3], CIFAR-10[4], MNIST[5] and ORL[6]. The summary of them is shown in Table 2.

**Comparative Methods.** We compare our algorithm with state-of-the-art feature selection approaches, including supervised model, for instance, alpha-investing (Alpha) (Zhou et al., 2005), as well as unsupervised model, e.g., $\lambda$-ridge leverage score (LevS) (Alaoui & Mahoney, 2015) and Laplacian score (LapS) (He et al., 2005). We also compare to sparse random projection (SEC) (Liu et al., 2017) which is particularly designed for $k$-means clustering.

**Pipeline.** After running our method and the baselines above, we obtain a reduced set of features. Then we feed it to the standard $k$-means clustering that is available in Matlab. We also report the clustering result based on the original set of features, and we simply denote it by $k$-means.

**Results.** We report the clustering accuracy against the number of selected features in Figure 1. We can see that our algorithm achieves competitive performance with other batch methods. For example, our algorithm outperforms all the

[1]https://archive.ics.uci.edu/ml/datasets.html

[2]http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html

[3]http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[4]https://www.cs.toronto.edu/~kriz/cifar.html

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

[6]http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

*Table 1.* Comparison of dimension reduction methods for $k$-means clustering on data matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ with $n$ data points and $d$ features. $\delta \in (0,1)$ represents the confidence level. Note that we consider the high-dimensional regime where $n \ll d$.

| Methods | Dimension | Time | Approximation Ratio |
|---|---|---|---|
| Boutsidis et al. (2009) | $\mathcal{O}\left(\epsilon^{-2} k \log(k/\epsilon)\right)$ | $\mathcal{O}\left(\min(nd^2, n^2 d)\right)$ | $2+\epsilon$ |
| Boutsidis et al. (2015) | $\mathcal{O}\left(k/\epsilon^2\right)$ | $\mathcal{O}\left(nd\epsilon^{-2}k/\log(n)\right)$ | $2+\epsilon$ |
| Liu et al. (2017) | $\mathcal{O}\left(\max(\epsilon^{-2}(k+\log(1/\delta)), \frac{6}{\epsilon^2 \delta})\right)$ | $\mathcal{O}\left(\mathrm{nnz}(\boldsymbol{A})\right)$ | $1+\epsilon$ |
| Pourkamali-Anaraki & Becker (2017) | $\mathcal{O}\left(\log(n)/n\right)$ | $\mathcal{O}\left(nd\log(d) + d\log(n)\right)$ | N/A |
| **This Work** | $\mathcal{O}\left(\epsilon^{-2} n \log n \log(\\|\boldsymbol{A}\\|_2)\right)$ | $\mathcal{O}\left(n^3\right)$ | $1+\epsilon$ |

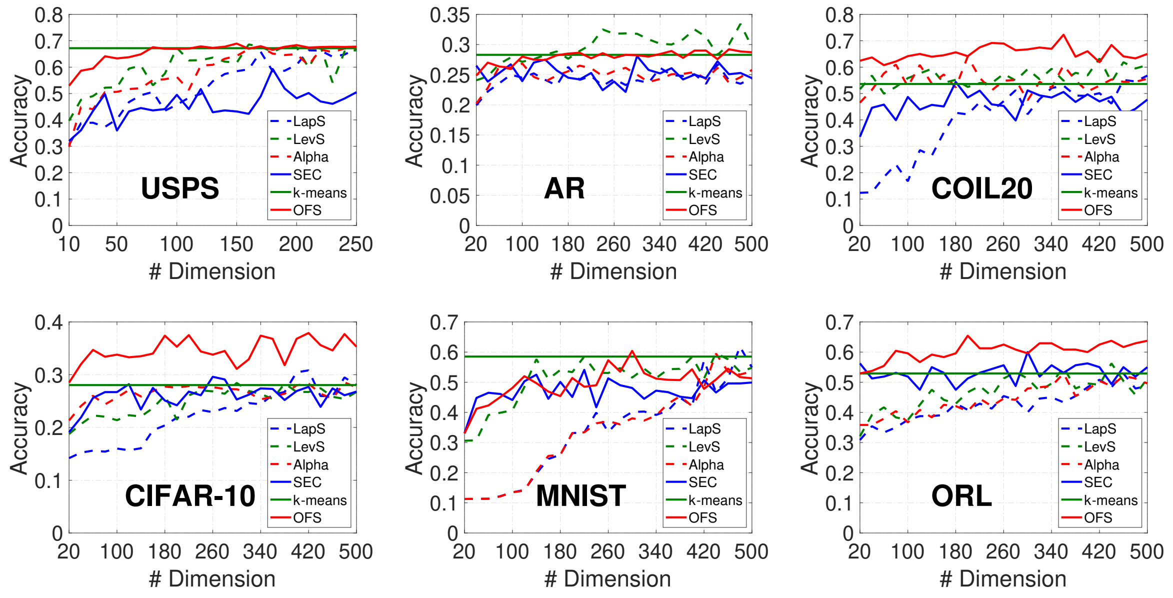

*Figure 1.* Clustering accuracy against the number of selected features.

*Table 2.* Summary of Data Sets.

| Data set | # Data Points | # Features |
|---|---|---|
| USPS | 9,298 | 256 |
| AR | 1,400 | 3,168 |
| COIL20 | 1,440 | 1,024 |
| CIFAR-10 | 60,000 | 512 |
| MNIST | 70,000 | 784 |
| ORL | 96,436 | 1,770 |

*Table 3.* CPU time of comparative algorithms (seconds in default).

| Data set | LapS | LevS | Alpha | SEC | OFS |
|---|---|---|---|---|---|
| USPS | 1.46 | 0.94 | 3.30 | 0.003 | 0.15 |
| AR | 0.70 | 7.76 | 26.99 | 0.005 | 1.74 |
| COIL20 | 0.23 | 0.64 | 4.79 | 0.002 | 0.58 |
| CIFAR-10 | 2 mins | 13.79 | 3 mins | 0.002 | 2.28 |
| MNIST | 19.38 | 10.23 | 9.62 | 0.003 | 1.32 |
| ORL | 0.24 | 0.45 | 0.42 | 0.003 | 0.05 |

baseline methods on COIL20, CIFAR-10 and ORL when the number of selected features varies from 10 to 500. The clustering performance on our selected subset even outperforms the one with all available features.

**Computational Efficiency.** We illustrate the running time in Table 3. In terms of efficiency, our algorithm outperforms most of the comparative methods. This is not surprising in that for batch methods, they often update the model with all the data while we process them one by one. For example, on the CIFAR-10 data set, Laplacian score requires 2 minutes for feature selection because the computation of the graph matrix based on global feature space is expensive. Our algorithm, in contrast, only requires a few seconds. The reason is that in each iteration, we operate with a skinny matrix $\tilde{\boldsymbol{A}}$ instead of the whole data matrix $\boldsymbol{A}$.

## 5. Conclusion

In this paper, we have proposed an online feature selection for $k$-means clustering. For features in a stream, we approximate its leverage score in an online manner and perform feature selection based on such an inexact score. We provide theoretical guarantee that our unsupervised approach produces an accurate estimation based on the original space. Moreover, in the high-dimensional regime the algorithm is computationally efficient and consumes little memory. Perhaps more importantly, our algorithm is capable of addressing streaming data which makes it a perfect fit for large-scale learning systems. In addition, we extend the analysis to the $k$-means clustering problem, and provably show that with the set of features reduced by our approach, we are still able to obtain a near-optimal solution to the original $k$-means problem. The extensive empirical study matches perfectly our analysis.

## Acknowledgements

Jing Wang, Jie Shen and Ping Li are supported by NSF-Bigdata-1419210 and NSF-III-1360971. Jing Wang is supported by grants from the Dalio Foundation. The work was done when Jing Wang was a postdoc at Rutgers University.

## A. Technical Lemmas

We provide a technical lemma which is due to (Tropp et al., 2011).

**Lemma 4.** *Let $\boldsymbol{Y}_0, \boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$ be a matrix martingale that are self-adjoint matrices with dimension d, and let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ be such that $\boldsymbol{X}_k = \boldsymbol{Y}_k - \boldsymbol{Y}_{k-1}$ for all $1 \leq k \leq n$. Assume*

$$\|\boldsymbol{X}_k\|_2 \leq R, \text{ almost surely for all } k.$$

*Define the predictable quadratic variation process*

$$\boldsymbol{W}_k := \sum_{j=1}^{k} \mathbf{E}_{j-1}\left[\boldsymbol{X}_j^2\right] \text{ for all } k,$$

*where $\mathbf{E}_{j-1}\left[\boldsymbol{X}_j^2\right]$ denotes the expectation of $\boldsymbol{X}_j^2$ conditioning on $\boldsymbol{X}_1, \cdots, \boldsymbol{X}_{j-1}$. Then, for all $\epsilon > 0$ and $\sigma^2 > 0$,*

$$\Pr\left(\|\boldsymbol{Y}_n\|_2 \geq \epsilon \text{ and } \|\boldsymbol{W}_n\|_2 \leq \sigma^2\right) \leq d \cdot \exp\left(\frac{-\epsilon^2/2}{\sigma^2 + R\epsilon/3}\right).$$

## References

Alaoui, A. and Mahoney, M. W. Fast randomized kernel ridge regression with statistical guarantees. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, pp. 775–783, 2015.

Awasthi, P., Blum, A., and Sheffet, O. Stability yields a PTAS for $k$-median and $k$-means clustering. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, pp. 309–318, 2010.

Boutsidis, C., Drineas, P., and Mahoney, M. W. Unsupervised feature selection for the $k$-means clustering problem. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, pp. 153–161, 2009.

Boutsidis, C., Zouzias, A., Mahoney, M. W., and Drineas, P. Randomized dimensionality reduction for $k$-means clustering. *IEEE Trans. on Information Theory*, 61(2): 1045–1062, 2015.

Cai, D., Zhang, C., and He, X. Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 333–342, 2010.

Chen, J., Stern, M., Wainwright, M. J., and Jordan, M. I. Kernel feature selection via conditional covariance minimization. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, pp. 6949–6958, 2017.

Cohen, M. B., Lee, Y. T., Musco, C., Musco, C., Peng, R., and Sidford, A. Uniform sampling for matrix approximation. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science*, pp. 181–190. ACM, 2015.

Cohen, M. B., Musco, C., and Pachocki, J. Online row sampling. In *Proceedings of the 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pp. 7:1–7:18, 2016.

Donoho, D. and Jin, J. Higher criticism thresholding: Optimal feature selection when useful features are rare and weak. *Proceedings of the National Academy of Sciences*, 105(39):14790–14795, 2008.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

Forman, G. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3(3):1289–1305, 2003.

Gittens, A. and Mahoney, M. W. Revisiting the Nyström method for improved large-scale machine learning. *Journal of Machine Learning Research*, 28(3):567–575, 2013.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(3):1157–1182, 2003.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

He, X., Cai, D., and Niyogi, P. Laplacian score for feature selection. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems*, pp. 507–514, 2005.

Kong, X. and Yu, P. S. Semi-supervised feature selection for graph classification. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 793–802, 2010.

Koutis, I., Levin, A., and Peng, R. Faster spectral sparsification and numerical algorithms for SDD matrices. *ACM Trans. Algorithms*, 12(2):17, 2016.

Li, M., Miller, G. L., and Peng, R. Iterative row sampling. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, pp. 127–136, 2013.

Liu, W., Shen, X., and Tsang, I. Sparse embedded $k$-means clustering. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, pp. 3321–3329, 2017.

Musco, C. and Musco, C. Recursive sampling for the Nyström method. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, pp. 3836–3848, 2017.

Ostrovsky, R., Rabani, Y., Schulman, L. J., and Swamy, C. The effectiveness of lloyd-type methods for the $k$-means problem. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 165–176, 2006.

Peng, H., Long, F., and Ding, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226–1238, 2005.

Perkins, S. and Theiler, J. Online feature selection using grafting. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 592–599, 2003.

Pourkamali-Anaraki, F. and Becker, S. Preconditioned data sparsification for big data with applications to PCA and $k$-means. *IEEE Trans. on Information Theory*, 63(5): 2954–2974, 2017.

Shen, J. and Li, P. On the iteration complexity of support recovery via hard thresholding pursuit. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3115–3124, 2017.

Spielman, D. A. and Srivastava, N. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6): 1913–1926, 2011.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

Tropp, J. A. et al. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16: 262–270, 2011.

Wang, J., Zhao, Z.-Q., Hu, X., Cheung, Y.-M., Wang, M., and Wu, X. Online group feature selection. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 1757–1763, 2013.

Wang, J., Wang, M., Li, P., Liu, L., Zhao, Z., Hu, X., and Wu, X. Online feature selection with group structure analysis. *IEEE Transactions on Knowledge and Data Engineering*, 27(11):3029–3041, 2015.

Wei, X. and Philip, S. Y. Unsupervised feature selection by preserving stochastic neighbors. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 995–1003, 2016.

Wu, X., Yu, K., Wang, H., and Ding, W. Online streaming feature selection. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 1159–1166, 2010.

Xu, Z., Huang, G., Weinberger, K. Q., and Zheng, A. X. Gradient boosted feature selection. In *Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 522–531, 2014.

Yang, W. and Xu, H. A unified robust regression model for lasso-like algorithms. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 585–593, 2013.

Zhang, T. On the consistency of feature selection using greedy least squares regression. *Journal of Machine Learning Research*, 10(3):555–568, 2009.

Zhang, Y., Ray, S., and Guo, E. W. On the consistency of feature selection with lasso for non-linear targets. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 183–191, 2016.

Zhao, Z. and Liu, H. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 1151–1157, 2007.

Zhou, J., Foster, D., Stine, R., and Ungar, L. Streaming feature selection using alpha-investing. In *Proceedings of the 11st ACM International Conference on Knowledge Discovery and Data Mining*, pp. 384–393, 2005.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 912–919, 2003.