Data Min Knowl Disc (2018) 32:453–482
https://doi.org/10.1007/s10618-017-0542-x

An improvement of the parameterized frequent directions algorithm

Deena P. Francis[1] · **Kumudha Raimond**[1]

Received: 17 August 2016 / Accepted: 31 August 2017 / Published online: 16 September 2017

Abstract Matrix sketching is a technique used to create summaries of large matrices. Frequent directions (FD) and its parameterized variant, α-FD are deterministic sketching techniques that have theoretical guarantees and also work well in practice. An algorithm called the iterative singular value decomposition (iSVD) has been shown to have better performance than FD and α-FD in several datasets, despite the lack of theoretical guarantees. However, in datasets with major and sudden drift, iSVD performs poorly when compared to the other algorithms. The α-FD algorithm has better error guarantees and empirical performance when compared to FD. However, it has two limitations: the restriction on the effective values of its parameter α due to its dependence on sketch size and its constant factor reduction from selected squared singular values, both of which result in reduced empirical performance. In this paper, we present a modified parameterized FD algorithm, β-FD in order to overcome the limitations of α-FD, while maintaining similar error guarantees to that of α-FD. Empirical results on datasets with sudden and major drift and those with gradual and minor or no drift indicate that there is a trade-off between the errors in both kinds of data for different parameter values, and for $\beta \approx 28$, our algorithm has overall better error performance than α-FD.

Keywords Matrix sketching · Streaming data · Frequent directions · β-FD · Parameterized FD

Responsible editor: Johannes Fürnkranz.

✉ Deena P. Francis
deena.francis@gmail.com

Kumudha Raimond
kraimond@karunya.edu

[1] Department of Computer Sciences Technology, Karunya University, Coimbatore, India

1 Introduction

Matrices with large number of examples and/or large number of attributes pose a significant challenge in large-scale machine learning applications like recommender systems, gene expression data analysis etc. Storing as well as manipulating such large data has become difficult, particularly in the streaming data setting. Creating a sketch of the data matrix is a useful step towards reducing the amount of storage as well as reducing the computational costs of subsequent operations performed on it. Computing a low rank approximation of the matrix is one such approach of matrix sketching. This can be done by performing the singular value decomposition (SVD) of the matrix. But for an $n \times d$ matrix, computing its SVD requires time $O(\min\{nd^2, n^2d\})$ which becomes a problem in situations where n and/or d is large. In order to overcome this problem, other low rank approximation schemes have been proposed in the past (Drineas et al. 2006; Har-Peled 2014). Given a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and a rank k, such methods construct a matrix $\mathbf{X}_k$ with rank k such that $\mathbf{X}_k$ is a good approximation to $\mathbf{X}$. A sketch of a matrix is a summarized version of it that has certain theoretical guarantees. Randomized matrix sketching techniques have been developed by Woodruff (2014) and Mahoney (2011), a column sampling technique was proposed by Boutsidis et al. (2009), and random projection based approaches were proposed by Sarlós (2006), Achlioptas and McSherry (2007), Clarkson and Woodruff (2013) and Nelson and Nguyên (2013). A deterministic matrix sketching algorithm, FD was originally proposed by Liberty (2013) which was shown to have better empirical performance than previous methods. It was later improved upon by Ghashami et al. (2014) and Desai et al. (2016).

This work focuses on the deterministic matrix sketching algorithm, FD and its parameterized variant. The FD (Liberty 2013) algorithm creates the sketch of a matrix in the streaming data model (rows arrive in a streaming manner). It has a number of advantages over previously proposed methods for matrix sketching. It has good provable guarantees (Liberty 2013; Ghashami and Phillips 2014) and also has better empirical performance than other techniques (Ghashami et al. 2014; Desai et al. 2016). A heuristic approach called iSVD performs better than FD in practice for datasets with gradual and minor or no drift. However, it fails to produce a good enough sketch when the input data contains sudden and major drift.

Later on, a parameterized form of the FD algorithm called α-FD was proposed by Ghashami et al. (2014). The aim of this algorithm was to match the empirical performance of iSVD on datasets with gradual and minor or no drift, and that of FD on datasets with sudden and major drift, while maintaining the theoretical guarantees of FD. It has improved empirical performance as well as better theoretical guarantees when compared to FD. Its performance in datasets with gradual and minor or no drift is slightly worse than that of iSVD. But it obtained significantly better results than iSVD on datasets with major and sudden concept drift (Ghashami et al. 2014). However, it has two main drawbacks. The first is the restriction on the *effective* values of its parameter α due to its dependence on sketch size and its constant factor reduction from selected squared singular values, both of which result in reduced empirical performance.

In this work an improvement of the α-FD algorithm is proposed. It overcomes the drawbacks of α-FD, while maintaining error guarantees similar to that of α-FD. It has better empirical performance than α-FD on several real-world as well as synthetic datasets, and also performs significantly better than iSVD on datasets with major and sudden drift.

1.1 Data and concept drift

An important characteristic of data that has significant impact on the performance of learning algorithms is drift. Concept drift (Schlimmer and Granger 1986; Widmer and Kubat 1996) is a phenomenon in which the underlying distribution of the data keeps changing over time. It can be categorized into types based on factors such as subject, frequency, transition, re-occurrence and magnitude (Webb et al. 2016). This work considers only unsupervised data, so we consider only drifts that happen due to change in the data distribution. This can be categorized as covariate drift according to Webb et al. (2016). A covariate drift occurs when the distribution of the data changes over time and throughout the work, whenever we use the term concept drift, we mean the covariate drift of the data. The duration of the occurrence of drift also introduces two categories of drift, namely, sudden/abrupt and gradual (Tsymbal 2004). The magnitude of the drift also introduces categories such as major and minor drifts. Data streams in which the distribution changes suddenly and dramatically are adversarial to the iSVD algorithm (Ghashami et al. 2014).

First we introduce a quantifiable notion of the magnitude of drift in data streams. A difference function D (Webb et al. 2016) can be used to measure the difference between two distributions of data streams that arrive at two different time instances.

$$\text{Drift Magnitude} = D(t, t + p) \tag{1}$$

This function is domain dependent, but some of the choices are Kullback–Leibler (KL) divergence (Kullback and Leibler 1951), Hellinger distance (Hoens et al. 2011), etc. The drift magnitude measures the difference between the data distribution at two time instances t and $t + p$, $p > 0$. If the drift magnitude between the data streams observed at two different times is greater than a threshold, κ, it can be concluded that a drift has occurred.

Let S_x be the starting time of a stable concept x and E_y be the ending time of another stable concept y. The data stream has minor drift if the criterion in Eq. 2 is satisfied and it has major drift if Eq. 3 is satisfied. Here κ is a positive real number that acts as a threshold.

$$D(E_x, S_y) < \kappa \tag{2}$$

$$D(E_x, S_y) \geq \kappa \tag{3}$$

Sudden drift is said to occur when Eq. 4 is satisfied. Here η is a small natural number that is an upper bound on the duration during which sudden drift occurs. A gradual drift occurs when for μ, the maximum difference between the two concepts during

time period m, the Eq. 5 is satisfied. Here m is a natural number, typically much bigger than η. Whenever the drift magnitude rises slowly up to a threshold μ, gradual drift is said to occur.

$$S_y - E_x \leq \eta \tag{4}$$

$$\forall_{t \in [E_x, S_y - m]} D(t, t + m) \leq \mu \tag{5}$$

In this work, data with sudden, gradual, minor, major and no drift are considered.

1.2 Overview of main contributions and scope

While α-FD comes with theoretical guarantees and clearly outperforms iSVD in datasets with sudden and major drift, it has certain limitations which restrict its empirical performance. The drawbacks of the previous method is overcome by the proposed algorithm, which we call β-FD. The key contributions are summarized as follows.

- We propose β-FD algorithm that has a parameter β and a special function aimed at reducing the effect of noisy components in the data.
- We provide theoretical guarantees with regard to performance.
- We also provide extensive experimental results demonstrating an improved error performance when compared to the previous method.
- We demonstrate the ability to improve the performance of the proposed algorithm to handle the two different kinds of data considered in this work, namely data with (1) sudden and major drift and (2) gradual and minor or no drift, by fine tuning its parameter β.

The β-FD algorithm creates the sketch of a matrix, where the rows arrive as streams. The accuracy of its sketch remains unaffected and its theoretical guarantees hold regardless of whether the data is mean-centered or not. The requirement of mean-centering comes into play only when a machine learning algorithm like Principal Component Analysis (PCA) is to be applied on the data, as it is the case for a dataset called Birds (Sect. 5.2).

The number of rows and columns of the input matrix can be arbitrarily large. The experiments (Sect. 5) carried out in this work consist of two parts, namely the sketching part and the performance evaluation part. While the former part involves creating a sketch of the input data as it arrives in a stream, the latter part involves computing the error between the original data matrix and the computed sketch. Since the error measurement, which involves computing the full SVD of the whole data matrix, is costly ($O(n^2 d)$ in the worst case, n is the number of examples and d is the number of attributes), we have limited the size of the datasets considered in the experiments. This however does not mean that the proposed algorithm is limited to handling such small datasets. On the contrary, the proposed algorithm can handle datasets of any size (rows or columns). The running time of the proposed algorithm is only $O(ndl)$, where l is the sketch size. Some of the datasets are also sparse in nature. The proposed algorithm can also handle data with concept drift. There are many kinds of concept drift present in modern data, but in our work we have considered only data with

sudden, gradual, major, minor or no drift. The detailed definitions of these types of drift are given in Sect. 1.1. In the case of data with major drift there exists a simple baseline which discards the current sketch of the data and starts anew as soon as a major drift is encountered. We also provide a comparison of β-FD with this simple baseline (Sect. 5.5).

The paper is organized as follows: Sect. 2 describes the basic formulas and notations used in this work, the related previous works, their limitations and a discussion on how to overcome them. Section 3 describes the proposed β-FD algorithm. The theoretical guarantees of the algorithm are given in Sect. 4. Experimental results are given in Sect. 5. Section 6 provides the conclusion.

2 Preliminaries

In this section the notations used throughout the work, the related works, their limitations and the ways of overcoming those limitations are discussed.

2.1 Notations used

Let $\mathbf{A}$ be a matrix with size $n \times d$, where n is the number of rows and d is the number of columns. The entries of the matrix $\mathbf{A}$ arrive as a stream of its rows, $\mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_n, \quad \mathbf{a}_i \in \mathbb{R}^d$. Throughout the work, array and matrix indices start from 1. The Frobenius norm of $\mathbf{A}$ is defined as $\|\mathbf{A}\|_F = \sum_{i=1}^{n} |\mathbf{a}_i|^2$. The spectral norm of $\mathbf{A}$ is $\|\mathbf{A}\|_2 = max_{\{\mathbf{x}:\|\mathbf{x}\|=1\}} \|\mathbf{A}\mathbf{x}\|$. The SVD of $\mathbf{A}$ results in three matrices, $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$ such that $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices of sizes $n \times n$ and $d \times n$ respectively. $\mathbf{\Sigma}$ is a diagonal matrix of size $n \times n$ that contains the singular values $\sigma_1, \sigma_2, \ldots \sigma_n$ of the matrix $\mathbf{A}$ such that $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_n$, where $\forall i, \sigma_i \geq 0$. The covariance error between the input matrix $\mathbf{A}$ and its sketch $\mathbf{B}$ is computed as $\|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2$. Another error measure used is projection error which is computed as $\|\mathbf{A} - \mathbf{A}\mathbf{B}_k^{\dagger}\mathbf{B}_k\|_F^2$ where $\mathbf{B}^{\dagger}$ is the pseudo-inverse of $\mathbf{B}$ and $\mathbf{B}_k$ is the k-rank approximation of $\mathbf{B}$ for some k. Both these error measures have been used in Ghashami et al. (2014).

2.2 Related works

The sketch of a matrix has been produced using methods like random projection (Sarlós 2006; Achlioptas and McSherry 2007; Clarkson and Woodruff 2013; Nelson and Nguyên 2013), column/row sampling (Boutsidis et al. 2009) and deterministic techniques (Liberty 2013; Ghashami et al. 2014; Desai et al. 2016). We focus on deterministic sketching algorithm, FD and its parameterized variant, α-FD.

2.2.1 Frequent directions

FD algorithm (Liberty 2013) is a deterministic matrix sketching algorithm that takes a row of an input matrix at a time and produces a sketch matrix. For any given matrix

$\mathbf{A} \in \mathbb{R}^{n \times d}$, a sketch matrix $\mathbf{B} \in \mathbb{R}^{l \times d}$ is constructed such that the following bound is satisfied.

$$\|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2 \leq \frac{2}{l}\|\mathbf{A}\|_F^2 \tag{6}$$

This bound was further improved by Ghashami et al. (2016) to provide the following error bounds,

$$\|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2 \leq \|\mathbf{A} - \mathbf{A_k}\|_F^2/(l-k) \tag{7}$$

$$\|\mathbf{A} - \pi_{\mathbf{B}_k}(\mathbf{A})\|_F^2 \leq \left(1 + \frac{k}{l-k}\right)\|\mathbf{A} - \mathbf{A}_k\|_F^2 \tag{8}$$

where $\mathbf{A}_k$ is the best k-rank approximation of $\mathbf{A}$, l is the number of rows in the sketch, and $\pi_{\mathbf{B}_k}(\mathbf{A}) = \mathbf{A}\mathbf{B}_k^{\dagger}\mathbf{B}_k$. This algorithm extended the idea of frequent item approximation problem to the problem of matrix sketching. The steps involved are explained in Algorithm 1. For the FD algorithm, in the *reduceRank* procedure (Eq. 9), $\alpha = 1$. Here $\mathbf{B}_i$ denotes the value of $\mathbf{B}$ at the end of the ith iteration, $\mathbf{C}_i$ is not actually computed in the algorithm, it is only needed for the proof. The algorithm maintains an $l \times d$ sketch matrix $\mathbf{B}$ which is first populated with l rows of $\mathbf{A}$. Then SVD of this matrix $\mathbf{B}$ is computed, $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. This SVD computation is inexpensive since the number of rows l in $\mathbf{B}$ is very small compared to n. Then a singular value shrinking step is done where $\mathbf{\Sigma}' \leftarrow \sqrt{\mathbf{\Sigma}^2 - \delta\mathbb{I}_{l\times l}}$ where $\mathbf{\Sigma}'$ is the rank reduced version of $\mathbf{\Sigma}$ and $\delta = \mathbf{\Sigma}_{l,l}^2$. The value of $\mathbf{B}$ is updated with the result of $\mathbf{\Sigma}'\mathbf{V}^T$. The running time of this algorithm is $O(ndl^2)$. A fast variant of FD improves this to $O(ndl)$ for $\delta = \mathbf{\Sigma}_{l/2,l/2}^2$ and setting $\mathbf{\Sigma}' \leftarrow \max\{\sqrt{\mathbf{\Sigma}^2 - \delta\mathbb{I}_{l\times l}}, 0\}$.

Algorithm 1 Generic-FD

```
Input: l, A ∈ R^{n×d}, α ∈ [0, 1]
B_0 ← 0^{l×d}
for i ∈ 1 ... n do
    Insert a_i into zero valued row of B_{i−1} and store into B_i
    if B_i has zero valued rows then
        continue
    end if
    [U, Σ, V] = SVD(B_i)
    C_i = ΣV^T        // Used only in the proof, not computed.
    Σ' = reduceRank(Σ, α)
    B_i = Σ'V^T
end for
return B_i
```

2.2.2 *Parameterized FD algorithm*

The FD algorithm was improved by Ghashami et al. (2014) using a parameter $\alpha \in [0, 1]$. The algorithm basically uses the same steps as that of Liberty (2013), but makes

an important improvement wherein the singular value matrix $\mathbf{\Sigma}$ undergoes a selective α dependent reduction step. The parameter α essentially determines the index of the singular values from which the reduction happens. This is done as follows. $\sigma'_1 \leftarrow \sigma_1, \sigma'_2 \leftarrow \sigma_2, \ldots, \sigma'_{(1-\alpha)l} \leftarrow \sigma_{(1-\alpha)l}, \sigma'_{(1-\alpha)l+1} \leftarrow \sqrt{\sigma^2_{(1-\alpha)l+1} - \delta}, \sigma'_{(1-\alpha)l+2} \leftarrow \sqrt{\sigma^2_{(1-\alpha)l+2} - \delta}, \ldots, \sigma'_l \leftarrow \sqrt{\sigma^2_l - \delta}$ where $\delta = \sigma_l$. Note that the singular values between 1 and $(1-\alpha)l$ remain unchanged, whereas the rest of the singular values are reduced by the constant factor δ. The *reduceRank*$(\mathbf{\Sigma}, \alpha)$ procedure returns the matrix $\mathbf{\Sigma}'$, given by Eq. 9.

$$\mathbf{\Sigma}' = \text{diag}(\mathbf{\Sigma}_{1,1}, \ldots, \mathbf{\Sigma}_{l(1-\alpha),l(1-\alpha)}, \sqrt{\mathbf{\Sigma}^2_{l(1-\alpha)+1,l(1-\alpha)+1} - \delta}, \ldots, \sqrt{\mathbf{\Sigma}^2_{l,l} - \delta}) \quad (9)$$

where $\delta = \mathbf{\Sigma}^2_{l,l}$ and diag() denotes the diagonalization operation. We refer to this parameterized algorithm, that combines *reduceRank* with the *Generic-FD* algorithm as α-FD. The steps are the same as Generic-FD (Algorithm 1), but using $0 < \alpha < 1$. The error guarantees associated with this algorithm are described below.

$$\|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2^2 \leq \|\mathbf{A} - \mathbf{A}_k\|_F^2/(\alpha l - k) \quad (10)$$

$$\|\mathbf{A} - \pi_{\mathbf{B}_k}(\mathbf{A})\|_F^2 \leq \frac{\alpha l}{\alpha l - k}\|\mathbf{A} - \mathbf{A}_k\|_F^2 \quad (11)$$

The time complexity of this algorithm is the same as that of FD.

2.2.3 Fast parameterized FD algorithm

The fast parameterized FD algorithm was proposed by Desai et al. (2016). We refer to this algorithm as Fast-α-FD. It was suggested that setting δ to be $\mathbf{\Sigma}^2_{t,t}$ where $t = (l - l\alpha/2)$ will cause the α-FD algorithm to run faster. Similar to fast FD, the reduction step is $\mathbf{\Sigma}' \leftarrow \max\{\sqrt{\mathbf{\Sigma}^2 - \delta\mathbb{I}_{l\times l}}, 0\}$. This algorithm has the same bounds as α-FD and the time complexity of this algorithm is $O(ndl/\alpha)$.

2.2.4 Comparison of FD and its parameterized variants

FD algorithm's *reduceRank* step involves making the lth singular value to be zero, and subsequently reducing a constant value from the squares of the rest of the singular values. Since the singular values are sorted in the non-increasing order, and the last value corresponds to the least important vector's magnitude, the effect of noisy components (associated with the least singular values) is reduced.

In α-FD, the parameter α determines the index of singular values from which the reduction step must take place. This algorithm is more effective in removing noise when compared to FD, because it subtracts square of the lth singular value from the chosen index onward, and not just from the first singular value. This in effect reduces the noise in the data. Instead of considering just the last singular value as that belonging to noise, the last $l\alpha$ $(= l - \{l(1-\alpha)+1\}+1)$ singular values are considered

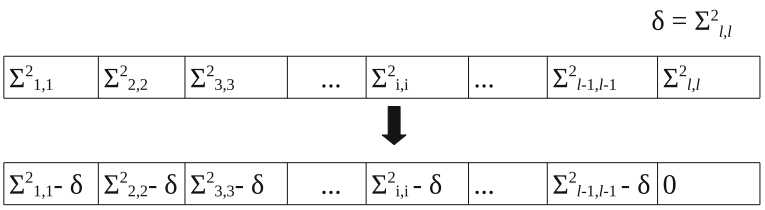

Fig. 1 *reduceRank()* in FD algorithm

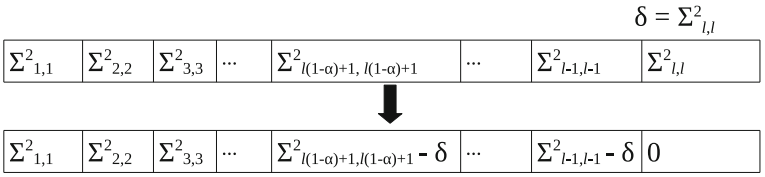

Fig. 2 *reduceRank()* in α-FD algorithm

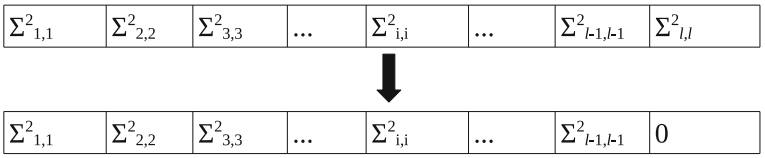

Fig. 3 *reduceRank()* in iSVD algorithm

to contain noise. Figures 1 and 2 shows the *reduceRank* procedure of FD and α-FD respectively. In Figs. 1 and 2, only the reduction from squared singular values are shown for clarity, whereas in the actual algorithm, during the reduction, square root of the reduced squared singular values are computed. In Fast-α-FD, the steps involved are the same as parameterized FD, but the number of SVD computations is reduced, and hence only the running time is improved, without actually providing any error performance improvement.

2.2.5 Iterative SVD

iSVD is an algorithm that performs SVD computation for data streams. The work of Brand (2002) computes the left singular vectors incrementally from the rows of a large matrix. Similarly, many other works (Hall et al. 1998; Levey and Lindenbaum 2000), have performed SVD incrementally. The steps are the same as in Generic-FD (Algorithm 1), but with $\alpha = 0$. See Fig. 3 to see the *reduceRank* procedure in the case of iSVD. Here, the last singular value is set to zero and the rest of them remain unchanged. This algorithm does not have any theoretical guarantees, but it has been shown to perform well on many real-world datasets with gradual or no drift (Ghashami et al. 2014). The work of Ghashami et al. (2014) introduced a dataset with sudden and major concept drift that is adversarial to iSVD. In this dataset, iSVD was found to perform poorly when compared to α-FD algorithm.

2.3 Limitations of previous works

The α-FD algorithm improves the FD algorithm by providing better error bounds as well as better empirical performance. However, it has two main limitations, both of which result in reduced empirical performance.

- It uses a constant factor (δ) reduction from selected squared singular values $(\sigma^2_{l(1-\alpha)+1}, \ldots, \sigma^2_l)$.
- The restriction on the effective values of α due to its dependence on sketch size.

From the selected singular value onward, the *rankReduce* step subtracts a constant value. This method is unfair because it treats all the singular values with equal weightage, whereas in reality, they are arranged in the non-increasing order of their importance and hence should be treated differently. An associated sub-problem is that the reduction happens only from a selected index onward. This is a restriction on the flexibility of the algorithm, because we ideally want to include all singular values in the *rankReduce* step as in Liberty (2013).

The parameter α essentially chooses the index of singular values from which the *reduceRank* step is applied. This index is calculated as $l(1-\alpha)+1$. Although the range of values of α is [0, 1], its *effective* values are limited to the range $\{\frac{1}{l}, \frac{2}{l}, \ldots, \frac{l-1}{l}, \frac{l}{l}\}$. This is because, the index of the selected singular value $l(1-\alpha)+1$ can only be a positive integer and in order to satisfy this requirement, α can only take certain value of the finite set $\{\frac{1}{l}, \ldots, \frac{l}{l}\}$. For example, when $\alpha = \frac{1}{l}$, the selected index is $l(\frac{l-1}{l}) + 1 = l$, and when $\alpha = \frac{2}{l}$, the index becomes $l-1$, and so on. This shows that the values of α are dependent on the sketch, which is alright with respect to α-FD algorithm. But if were to change the algorithm in order to include all singular values in the *rankReduce* step and use a variable reduction as mentioned above, then this constraint on the value of α would be restrictive.

The iSVD algorithm outperforms α-FD on most datasets, but fails in the presence of sudden and major concept drift (Ghashami et al. 2014). The limitations of the algorithm are:

- Its poor performance in datasets with sudden and major drift (Ghashami et al. 2014).
- Lack of any theoretical guarantees.

Refer Sect. 5.3.2 for experimental results that show iSVD failing in the presence of data with sudden concept drift.

2.4 Overcoming the limitations of previous methods

Some of the ways of overcoming the limitations of the previous methods, namely, that of α-FD and iSVD are discussed in this section.

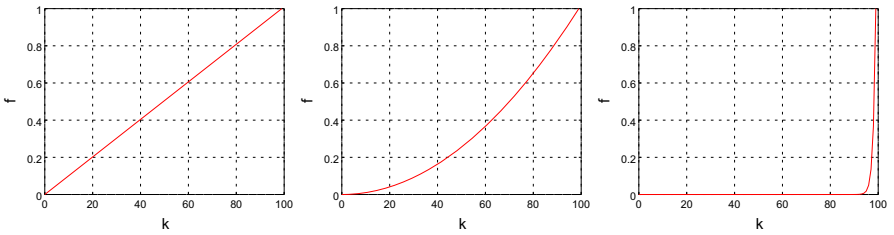

Fig. 4 Various choices of function f which obey properties 1, 2 and 3. **a** Linear function, **b** quadratic function, **c** exponential function

2.4.1 α-FD

First we deal with the problem of constant factor reduction in the *reduceRank* procedure of α-FD. Reducing the effect of noise by subtracting a certain quantity from the least few singular values (squared) is a sound idea. But instead of subtracting a constant value from the selected squared singular values, we subtract a variable quantity from each. This is because the singular values are sorted in the non-increasing order of their importance as discussed in Sect. 2.3. This variable quantity, should intuitively take monotonically increasing values, so that more is subtracted from the lth (least of the singular values), and less is a subtracted from $(l-1)$th value, and so on. Moreover, this variable can be set to the value of a function f of the sketch size, l, and the current singular value index k. In order for this idea to work correctly, we require the following properties to be satisfied by f.

1. $f(k,l)$ is a strictly increasing function of k, for a given l.
2. $f(0,l) = 0$
3. $f(l-1,l) = 1$

Using these three properties, one can devise a number of functions. Here, the various choices and their problems are enumerated for clarity.

Linear function The simplest choice for f is a linear function of k, such that the properties 2 and 3 are also satisfied. One such possible formulation is as follows.

$$f(k,l) = \frac{k}{l-1} \tag{12}$$

This function is a strictly increasing function with bounds as seen in Fig. 4a. The graph has been constructed by setting $l = 100$. It however, does not have the functionality we desire. Instead of slowly increasing, this function rapidly increases thereby causing more reduction than needed in some large singular values.

Quadratic function Another choice is a quadratic function of k and l.

$$f(k,l) = \frac{k^2}{(l-1)^2} \tag{13}$$

The graph of this function is shown in Fig. 4b. Again a rapid increase in the function values is seen, which can over-reduce some singular values. Ideally we want minimal (almost zero) reduction for largest singular values, but this quadratic function reduces even those singular values by a non-trivial amount. If this form of function is used, then the possibility of over-reducing (removing the actual data part of some singular values) increases.

Exponential function The next logical choice for f is an exponential function that obey properties 2 and 3. Such a function is given below.

$$f(k,l) = \frac{e^k - 1}{e^{l-1} - 1} \tag{14}$$

The graph of this function is shown in Fig. 4c. This function satisfies the three properties described above, and also has the desirable quality of being slowly increasing. However, if this function were used on all kinds of datasets, the results may not be optimal across all datasets. For instance, there may be datasets that require more reduction on smaller singular values, and on some datasets, a smaller reduction suffices. There is also another problem with using this function. Although f is bounded, the numerator and denominator by themselves are not bounded. In fact, both the numerator and denominator explode as k and l increases. For instance, if $k = 40$ and $l = 100$, the numerator takes a very large number and the denominator also takes a really large value, so the function f is potentially numerical unstable.

A tuneable exponential function It has been shown that an exponential function is a good choice for f. But using the same exponential function for all kinds of datasets may not yield optimal results. It also suffers from the problem of potential numerical instability of f. The ways of overcoming the two disadvantages of the exponential function (Eq. 14) are discussed here.

The first limitation can be overcome by introducing a tuning parameter $\beta \in (0, \infty)$ into f (Eq. 15) in order to make it generalize to a family of functions rather than to a single function. The second limitation can be overcome by making both the numerator and denominator of the function bounded. In the Eq. 15, the numerator and denominator are at most 1.

$$f(k,l) = \frac{e^{-\frac{(l-1-k)\beta}{l-1}} - e^{-\beta}}{1 - e^{-\beta}} \tag{15}$$

In Eq. 15, the parameter β takes on an infinite range of values. This ability to choose β from a continuous range provides fine-grained control over the performance of the algorithm. Depending on the value of β, the function takes on different forms. The detailed behavior of this function is discussed in Sect. 3 and its graph is shown in Fig. 5. The function (15) can be simplified as follows.

$$f(k,l) = \frac{e^{\frac{k\beta}{l-1}} - 1}{e^{\beta} - 1} \tag{16}$$

Next we deal with the problem of the restricted effective values of α due to its dependence on sketch size. A simple solution would be to make the parameter of the algorithm independent of the sketch size. In the proposed algorithm, we call this parameter β. In solution to the previous problem, we had introduced a tuneable exponential function with a parameter β. It can take any values in the range $(0, \infty)$, and thus offers a more flexible fine-tuning than α-FD.

2.4.2 iSVD

The main limitation of iSVD is its zeroing-out step, which simply discards the least important singular value. Whenever a sudden and major change occurs the new concept is treated as noise, and it is subsequently discarded. FD and its variants overcome this by replacing the zeroing-out step with a more smooth *rankReduce* step. This step involves reducing the singular values by an amount determined by the algorithm. So it would be beneficial to adopt a similar technique in order to deal with sudden and major concept drift. FD and its variants also have theoretical guarantees with regard to their performance.

3 Proposed algorithm

In this section, the proposed technique, its intuition, design requirements, and description are discussed.

3.1 Intuition

The highest singular values are generally associated with the signal, and the lowest valued among them are associated with noise. In α-FD's reduction step, the same value of δ is subtracted from all of the selected squared singular values. It is desirable to reduce the magnitude of noisy or least singular values. So the reduction step must happen in such a way that the highest singular values are reduced by a very small amount, and the lowest reduced by a larger amount. In order to do so, we introduce a scaling function $attenuate(\beta, k, l)$, which provides the desired scaling factor of δ for a set of l singular values each indexed by k based on a parameter $\beta \in (0, \infty)$. This function has the form of Eq. 16. Refer Sect. 2.4 for detailed discussion on the possible choices for this function and reasoning behind why the current choice was made. The function has the following properties: $\forall \beta \in (0, \infty)$

1. $attenuate(\beta, k, l)$ is a strictly increasing function of k, for a given β and l.
2. $attenuate(\beta, 0, l) = 0$
3. $attenuate(\beta, l-1, l) = 1$

Due to these properties, the highest singular value, σ_1 $(k = 0)$ is not reduced at all and the lowest singular value, σ_l $(k = l-1)$ is reduced to 0. The other singular values $\forall k \in (0, l-1)$ are reduced in such a way that σ_k is reduced by a strictly lower amount than σ_{k+1}.

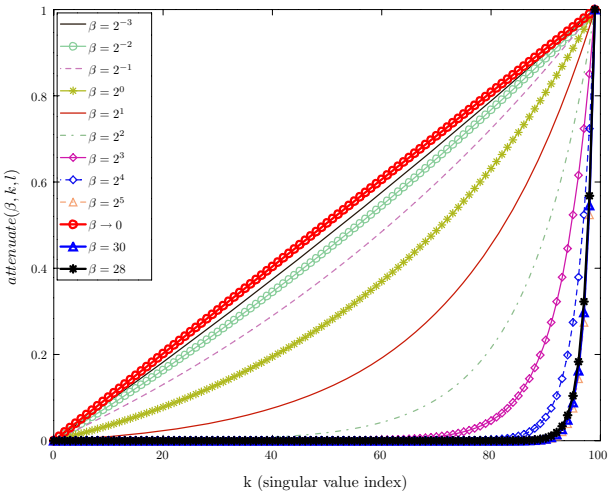

Fig. 5 Plot of $attenuate(\beta, k, l)$ for different values of β and $l = 100$

In Fig. 5 it can be seen that as $\beta \to 0$ *attenuate* behaves like a linear function whereas, as β assumes higher values the function is nearly zero for lower k (corresponding to higher singular values) and larger values for higher values of k. For $\beta = 30$, the function has nearly zero values for k till ~ 80, implying nearly zero reduction for first 80 singular values, which is the same as in the case of $\alpha = 0.2$-FD. The difference is that for $k > 80$ instead of a constant reduction (as in the case of α-FD), the reduction increases exponentially with k till $k = l - 1$ where reduction is maximum (=1) and hence the lth singular value is zeroed out.

Theoretically, β can assume infinitely many values, whereas in α-FD, *effective* values of the parameter α are limited to the countable set $\{\frac{1}{l}, \frac{2}{l}, \ldots, \frac{l-1}{l}, \frac{l}{l}\}$. In datasets with gradual and minor or no drift the smaller the value of the parameter, the lower the empirical errors observed, whereas in datasets with major and sudden drift (adversarial dataset and ConnectUS) empirical error reduces as α is increased. Setting α to $\approx$0.2 results in a good trade-off between errors in these two types of datasets (Ghashami et al. 2014). Since all $\beta \in (0, \infty)$ have a unique effect on the β-FD algorithm subject to the machine precision, we can better "fine tune" the trade-off between these errors by varying β.

3.2 β-FD

We introduce a variant of the parameterized FD algorithm called β-FD. In this algorithm, the *reduceRank* step has been modified to include a scaling function *attenuate()*. The β-FD algorithm uses the same basic steps as Generic-FD (Algorithm 1), but with an important difference. In step 10 of the algorithm, the *reduceRank* procedure is replaced with *modifiedReduceRank* that returns $\mathbf{\Sigma}'$, given by Eq. 17.

$$\mathbf{\Sigma}' = \text{diag}\left(\left[\sqrt{\mathbf{\Sigma}_{1,1}^2 - attenuate(\beta, 0, l)\delta}, \ldots \sqrt{\mathbf{\Sigma}_{l,l}^2 - attenuate(\beta, l-1, l)\delta}\right]\right) \tag{17}$$

where $l = \text{size}(\mathbf{\Sigma})$ and $\delta = \mathbf{\Sigma}_{l,l}^2$. The β-FD algorithm uses steps similar to α-FD algorithm, with the exception that the *rankReduce* step has been modified. In each iteration, the rank of B is reduced by subtracting the value of δ scaled by the function *attenuate()* (given in Eq. 18). There is another important difference between our algorithm and that of α-FD, which is that the reduction step is done for all singular values instead of the last few. The *attenuate* function returns a quantity γ given as follows.

$$\gamma = \left(\frac{e^{\frac{k\beta}{l-1}} - 1}{e^{\beta} - 1}\right) \tag{18}$$

Figure 5 shows the behavior of this function for different values of β. The running time of the algorithm is the same as that of α-FD which is $O(ndl^2)$. This could be improved to $O(ndl)$ as discussed later.

3.3 Fast β-FD algorithm

A fast version of the β-FD algorithm is introduced by incorporating the suggestion of Desai et al. (2016) (Sect. 3.1.2 in the referred work) in the proposed algorithm. Setting $\alpha = 0.2$, we get,

$$\begin{aligned} l_d &= (l - l\alpha/2) \\ &= 0.9l \end{aligned}$$

Here we set $\delta = \mathbf{\Sigma}_{l_d,l_d}^2$ and $l_d = 0.9l$ to make our algorithm comparable to Fast-$\alpha = 0.2$-FD in effective sketch size. As a result, after the reduction step is done, at least $l/10$ singular values are zeroed out and the SVD step is done only for every $l/10$ input rows. Thus the algorithm takes $O(nd + (n/(l/10))dl^2) = O(ndl)$ time. The *reduceRank* procedure for fast β-FD returns $\mathbf{\Sigma}'$, which is computed as follows.

$$\begin{aligned} \mathbf{\Sigma}' = \text{diag}\Big(\Big[&\sqrt{\mathbf{\Sigma}_{1,1}^2 - attenuate(\beta, 0, l_d)\delta}, \sqrt{\mathbf{\Sigma}_{2,2}^2 - attenuate(\beta, 1, l_d)\delta}, \ldots, \\ &\sqrt{\mathbf{\Sigma}_{l_d,l_d}^2 - attenuate(\beta, l_d - 1, l_d)\delta}, \text{zeros}(1, l - l_d)\Big]\Big). \end{aligned} \tag{19}$$

4 Theoretical guarantees

In this section we show that β-FD algorithm admits theoretical error guarantees similar to α-FD. For any unit vector $\mathbf{x} \in \mathbb{R}^d$,

Claim 1. $\|\mathbf{Ax}\|^2 - \|\mathbf{Bx}\|^2 \geq 0$
Claim 2. $\|\mathbf{Ax}\|^2 - \|\mathbf{Bx}\|^2 \leq \Delta$ where $\Delta > 0$
Claim 3. $\|\mathbf{Ax}\|_F^2 - \|\mathbf{Bx}\|_F^2 \geq g(\beta, l)\Delta$ where $\Delta > 0$ and $g(\beta, l)$ is a positive valued function of β and l.

In order to prove the claims, we first prove the following lemma.

Lemma 1 *For any unit vector* $\mathbf{x}$ *and* $\beta \in (0, \infty)$, $0 \leq \|\mathbf{C}_i\boldsymbol{x}\|^2 - \|\mathbf{B}_i\mathbf{x}\|^2 \leq \delta_i$

Proof

$$\|\mathbf{C}_i\mathbf{x}\|^2 - \|\mathbf{B}_i\mathbf{x}\|^2 = \sum_{j=1}^{l} \boldsymbol{\Sigma}_{j,j}^2 \langle \mathbf{v}_j, \mathbf{x}\rangle^2 - \boldsymbol{\Sigma}_{j,j}^{'2} \langle \mathbf{v}_j, \mathbf{x}\rangle^2 \tag{20}$$

$$= \sum_{j=1}^{l} \left(\boldsymbol{\Sigma}_{j,j}^2 - \boldsymbol{\Sigma}_{j,j}^{'2}\right) \langle \mathbf{v}_j, \mathbf{x}\rangle^2 \tag{21}$$

$$= \delta_i \sum_{j=1}^{l} attenuate(\beta, j-1, l) \langle \mathbf{v}_j, \mathbf{x}\rangle^2 \tag{22}$$

$$\leq \delta_i \sum_{j=1}^{l} attenuate(\beta, l-1, l) \langle \mathbf{v}_j, \mathbf{x}\rangle^2 \tag{23}$$

$$= \delta_i \sum_{j=1}^{l} 1 \langle \mathbf{v}_j, \mathbf{x}\rangle^2 \leq \delta_i \|\mathbf{x}\|^2 = \delta_i \tag{24}$$

Equations (23) and (24) follow from the properties 1 and 3 of attenuate function (see Sect. 3.1) respectively. Left side of the inequality of Lemma 1 is also true since

$$\delta_i \sum_{j=1}^{l} attenuate(\beta, j-1, l) \langle \mathbf{v}_j, \mathbf{x}\rangle^2 \geq 0$$

□

Using Lemma 1 with the fact $\|\mathbf{a}_i\mathbf{x}\|^2 = \|\mathbf{C}_i\mathbf{x}\|^2 - \|\mathbf{B}_{i-1}\mathbf{x}\|^2$, and summing up for all steps of our algorithm, it can be inferred that (see Lemma 2.3 of Ghashami and Phillips (2014))

$$0 \leq \|\mathbf{Ax}\|^2 - \|\mathbf{Bx}\|^2 \leq \sum_{i=1}^{n} \delta_i = \Delta$$

Thus Claims 1 and 2 are proved for our algorithm for any $\beta \in (0, \infty)$. Next we prove Claim 3.

Proof of Claim 3,

$$\begin{aligned}\|\mathbf{C}_i\|_F^2 &= \sum_{j=1}^{l} \mathbf{\Sigma}_{j,j}^2 = \sum_{j=1}^{l} \left(\mathbf{\Sigma}_{j,j}'^2 + \delta_i attenuate(\beta, j-1, l) \right) \\ &= \|\mathbf{B}_i\|_F^2 + \delta_i \sum_{k=0}^{l-1} attenuate(\beta, k, l) \\ &= \|\mathbf{B}_i\|_F^2 + \delta_i g(\beta, l) \quad \text{where} \quad g(\beta, l) = \sum_{k=0}^{l-1} attenuate(\beta, k, l)\end{aligned}$$

Now we use the fact that $\|\mathbf{a}_i\|^2 = \|\mathbf{C}_i\|_F^2 - \|\mathbf{B}_{i-1}\|_F^2$, we get

$$\begin{aligned}\|\mathbf{a}_i\|^2 &= \left(\|\mathbf{B}_i\|_F^2 + \delta_i g(\beta, l) \right) - \|\mathbf{B}_{i-1}\|_F^2 \\ \|\mathbf{A}\|_F^2 &= \sum_{i=1}^{n} \|\mathbf{a}_i\|^2 = \sum_{i=1}^{n} \|\mathbf{B}_i\|_F^2 - \|\mathbf{B}_{i-1}\|_F^2 + \delta_i g(\beta, l) = \|\mathbf{B}\|_F^2 + g(\beta, l)\Delta\end{aligned}$$

This implies Claim 3. □

Closed form expression of $g(\beta, l)$

$$\begin{aligned}g(\beta, l) &= \sum_{k=0}^{l-1} attenuate(\beta, k, l) \\ &= \sum_{k=0}^{l-1} \left(\frac{e^{\frac{k\beta}{l-1}} - 1}{e^{\beta} - 1} \right) \\ &= \frac{\sum_{k=0}^{l-1} \left(e^{\frac{k\beta}{l-1}} \right) - l}{e^{\beta} - 1} \\ &= \frac{\sum_{k=0}^{l-1} \left(\left(e^{\frac{\beta}{l-1}} \right)^k \right) - l}{e^{\beta} - 1} \\ &= \frac{\left(\frac{e^{\frac{\beta l}{l-1}} - 1}{e^{\frac{\beta}{l-1}} - 1} \right) - l}{e^{\beta} - 1} \\ &= \frac{(e^{\frac{\beta l}{l-1}} - 1 - l e^{\frac{\beta}{l-1}} + l)}{(e^{\frac{\beta}{l-1}} - 1)(e^{\beta} - 1)} \\ &= \frac{e^{\frac{\beta}{l-1}}(e^{\beta} - l) + l - 1}{(e^{\frac{\beta}{l-1}} - 1)(e^{\beta} - 1)}\end{aligned}$$

The closed form of $g(\beta, l)$ is given by $\frac{e^{\frac{\beta}{l-1}}(e^{\beta}-l)+l-1}{(e^{\frac{\beta}{l-1}}-1)(e^{\beta}-1)}$. The function $g(\beta, l)$ is a strictly decreasing function of β, for a constant l.

Let $\phi = e^{\frac{\beta}{l-1}}$, then

$$g(\beta, l) = \frac{\phi(e^{\beta} - l) + l - 1}{(\phi - 1)(e^{\beta} - 1)}$$

Theorem 1 *β-FD algorithm produces a sketch $\mathbf{B} \in \mathbb{R}^{l \times d}$ given an input matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ such that for all $k < g(\beta, l)$*

$$\begin{aligned} 0 \leq \|\mathbf{Ax}\|^2 - \|\mathbf{Bx}\|^2 &\leq \frac{\|\mathbf{A} - \mathbf{A}_k\|_F^2}{g(\beta, l) - k} \\ &= \frac{(\phi - 1)(e^{\beta} - 1)}{\phi(e^{\beta} - l) + l - 1 - k(\phi - 1)(e^{\beta} - 1)} \|\mathbf{A} - \mathbf{A}_k\|_F^2 \end{aligned}$$

and projection error admits the following bound

$$\begin{aligned} \|\mathbf{A} - \pi_{\mathbf{B}_k}(\mathbf{A})\|_F^2 &\leq \frac{g(\beta, l)}{g(\beta, l) - k} \|\mathbf{A} - \mathbf{A}_k\|_F^2 \\ &= \frac{\phi(e^{\beta} - l) + l - 1}{\phi(e^{\beta} - l) + l - 1 - k(\phi - 1)(e^{\beta} - 1)} \|\mathbf{A} - \mathbf{A}_k\|_F^2 \end{aligned}$$

The Proof of Theorem 1 can be arrived at by invoking Claim 3 in the proof of Lemmas 5 and 6 of Ghashami et al. (2014).

5 Experiments

The experimental settings and results are discussed in this section. We perform a comparison between Fast-α-FD algorithm, iSVD and Fast-β-FD algorithm. Here we used the value of 0.2 for α since it was the value of α for which better errors were obtained on the two different kinds of data when compared to FD and iSVD (Ghashami et al. 2014). To see the comparison of different values of α refer Ghashami et al. (2014).

5.1 Performance metrics

In order to evaluate the performance of the proposed algorithm, we use the following error metrics.

– *Scaled covariance error:* The scaled covariance error between the input matrix A and its sketch B is computed as

$$\text{Scaled covariance error} = \|\mathbf{A}^T\mathbf{A} - \mathbf{B}^T\mathbf{B}\|_2 / \|\mathbf{A}\|_F^2 \tag{25}$$

Table 1 Datasets and their characteristics

Dataset	# examples	# attributes	Type of drift	nnz (%)
Birds	11,789	312	None	100
Adversarial	10,000	500	Major and sudden	100
Random Noisy	10,000	500	None	100
Spam	9324	499	Minor and gradual	0.327
ConnectUS	394,792	512	Major and sudden	0.5578
MNIST	10,000	789	None	87.117
PeMS SF	63,360	963	None	100
HAR	10,299	561	None	100
Usenet	5995	500	Minor and gradual	0.2296

– *Scaled projection error:* The scaled projection error is calculated as

$$\text{Scaled projection error} = \|\mathbf{A} - \mathbf{A}\mathbf{B}_k^{\dagger}\mathbf{B}_k\|_F^2 / \|\mathbf{A} - \mathbf{A}_k\|_F^2. \tag{26}$$

5.2 Experimental setup

Datasets used: Synthetic as well as real world datasets were used for evaluating the performance of the three algorithms (iSVD, α-FD and β-FD). There are two different kinds of data considered in this work. The first kind of data has sudden and major concept drift (Adversarial, ConnectUS), and the second kind of data has gradual and minor or no drift (Birds, Random Noisy, Spam, MNIST, PeMS SF, HAR and Usenet). The datasets and their characteristics are shown in Table 1. Here $nnz(\mathbf{A})$ represents the number of non-zero entries in the matrix $\mathbf{A}$. The types of drift in the datasets have been identified experimentally with a simple windowing scheme and KL divergence measure (see Sect. 5.5), and from the literature.

– Birds (Wah et al. 2011): It contains the images of birds with each column representing some attribute. The size of the dataset is $11,789 \times 312$. This dataset needs to be centered around the mean because PCA is a typical operation applied on this dataset (Ghashami et al. 2014).
– Adversarial data: This data was generated using the method of Ghashami et al. (2014). There is a sudden drift in its stream which causes iSVD to perform poorly. The method of generation of this dataset is as follows. The projection of two random vectors $\mathbf{x}$ and $\mathbf{y}$ on two orthogonal subspaces $\mathbf{M}$ and $\mathbf{N}$ is computed. Here, $\mathbf{M} \in \mathbb{R}^{m_1}$, $\mathbf{N} \in \mathbb{R}^{m_2}$ and we set $m_1 = 400$ and $m_2 = 4$. The dataset is constructed as the concatenation of the normalized projection vectors, such that projection of $\mathbf{x}$ on $\mathbf{M}$ appears before the projection of $\mathbf{y}$ on $\mathbf{N}$. This data has a major and sudden drift due to this construction procedure, which becomes evident in Sect. 5.5. The size of this dataset is $10,000 \times 500$.
– Random noisy: This data was generated by a mechanism similar to Ghashami et al. (2014). The $n \times d$ matrix $\mathbf{A}$ ($n = 10,000, d = 500$) was constructed in

the following manner. **A** is composed of both signal and noise parts. The noise part consists of values taken i.i.d from the Gaussian distribution with mean 0 and standard deviation 1. It also has a signal to noise ratio controlling parameter ς set to 10. The signal part contains a k dimensional signal such that $k < d$. In the experiments we set $k = 30, 50$. When $k = 30$, the dataset is referred to as Rand30, and when $k = 50$, the dataset is referred to as Rand50.

- Spam: The data can be downloaded from http://mlkd.csd.auth.gr/concept-drift.html. The Spam dataset consists of spam messages and it exhibits minor and gradual drift (Katakis et al. 2010). From the data that was downloaded, the first 499 attributes were used and thus the data used in the experiments has a size of 9324×499.
- ConnectUS (Buss 2016): It is a large sparse dataset consisting of $394,792 \times 512$ values. It contains the web page preferences of users. This dataset exhibits sudden and major drift. In Sect. 5.5, the evidence for the sudden as well as major drift in the data is provided.
- MNIST (Lecun and Cortes 2009): This dataset consists of 60,000 images with 784 attributes. In the experiments 10,000 random images from this dataset were used.
- PeMS SF (Cuturi 2011): This dataset contains the occupancy rates of the car lanes of the San Francisco Bay area freeways from the Caltrans Performance Measurement System (PeMS). Data was collected every 10 min from 963 sensors for 440 days. The number of measurements taken in a day is 144. The number of columns in the dataset is 963 and the number of rows is $(144 \times 440) = 63,360$. The dataset obtained from the UCI repository contains data that has been permuted using the file provided along with the dataset. In order to obtain the dataset in the calender order, we used the inverse of the permutation.
- Human Activity Recognition (HAR) using smartphones dataset (Anguita et al. 2013): This dataset consists of the recordings of the activities of 30 subjects. The size of this dataset is $10,299 \times 561$.
- Usenet: This is an email dataset that exhibits minor and gradual concept drift (Katakis et al. 2008). It can be downloaded from http://mlkd.csd.auth.gr/concept_drift.html. In the experiments, 5995 examples and the first 500 attributes were used.

System specifications The experiments were carried out in a Fedora 24 system with Intel Core i7 CPU (3.5 GHz), and 16 GB of RAM. Coding was done in C++.

5.3 Results and discussion

This section consists of plots of covariance and projection errors obtained for various sketch sizes. Such plots have been made in order to understand the relation between the sketch size of the matrix sketch **B** and the error obtained. According to the theory, one can expect the error to decrease as the sketch size increases. The results for the two kinds of data considered in this work are provided here.

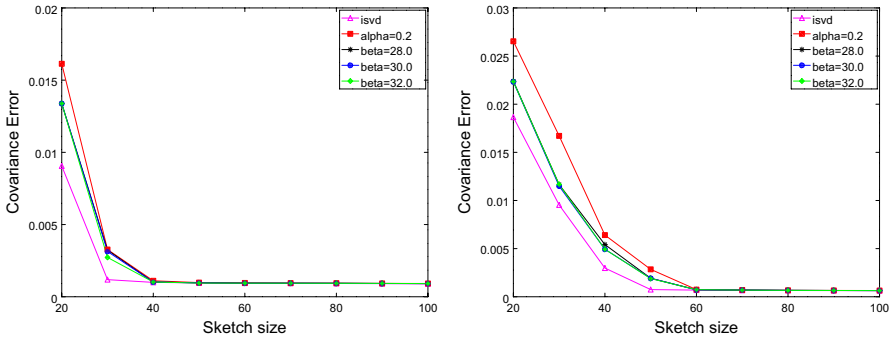

Fig. 6 Comparison of covariance error for Rand30 (left), Rand50 (right)

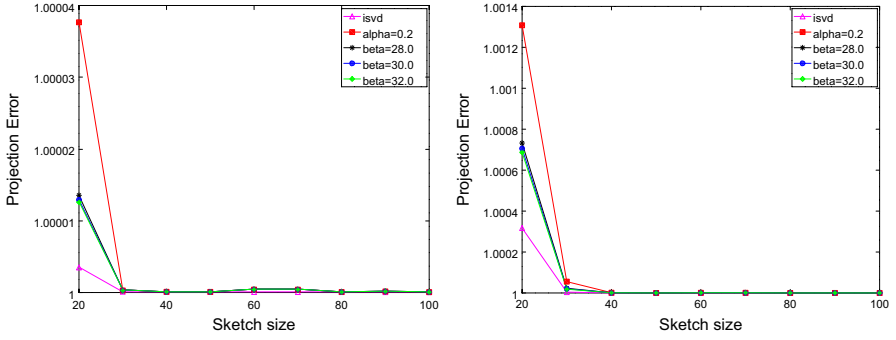

Fig. 7 Comparison of projection error for Rand30 (left), Rand50 (right)

5.3.1 Results for data with minor and gradual or no drift

Plots of covariance errors obtained for Rand30 and Rand50 datasets are given in Fig. 6. For the Rand30 dataset error obtained by β-FD is smaller than that of α-FD for sketch size less than 40. It can also be observed that on Rand50 dataset, β-FD outperforms 0.2α-FD on all small values of l ($l < 60$). For instance, when $l = 40$, α-FD has error 0.0064, whereas β-FD has error 0.00542. Projection error comparison on Rand30 and Rand50 datasets (Fig. 7) indicate that the proposed algorithm performs better than α-FD for sketch size less than 40, after which all the three algorithms have almost similar performance. In Fig. 8, on MNIST dataset, β-FD has low values of covariance error for all l. iSVD still has the lowest error when compared to the other algorithms. The experiments confirm the theory discussed in Sect. 3: as β increases, the error decreases. In the case of projection error for MNIST, for $l \leq 40$, β-FD has lower error values than α-FD.

It can be seen that in Fig. 9, β-FD algorithm performs much better than 0.2α-FD for all values of l, on the Birds dataset. For the Spam dataset, the β-FD algorithm for $\beta \approx 28$, outperforms 0.2α-FD for almost all values of l. Projection error comparison is shown in Fig. 10. In the Birds dataset, for $l = 20$, 0.2α-FD algorithm has a larger error when compared to both the other algorithms. Although for $l \geq 30$, β-FD has

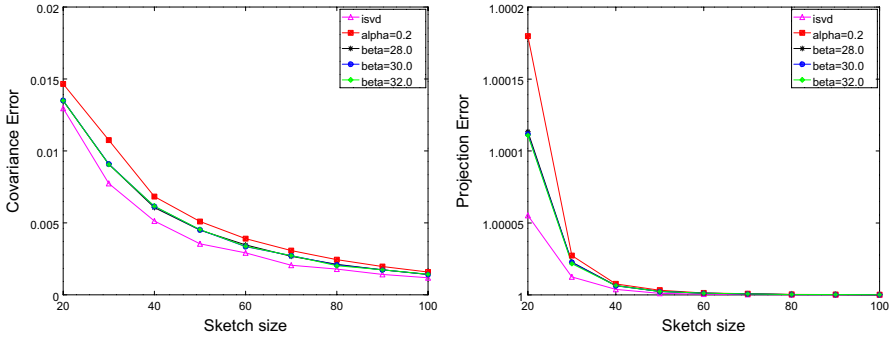

Fig. 8 Comparison of covariance (left) and projection (right) errors for MNIST

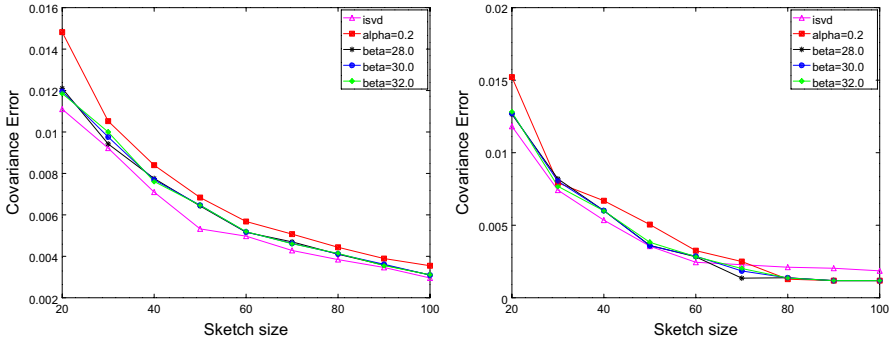

Fig. 9 Comparison of covariance error for Birds (left), Spam (right)

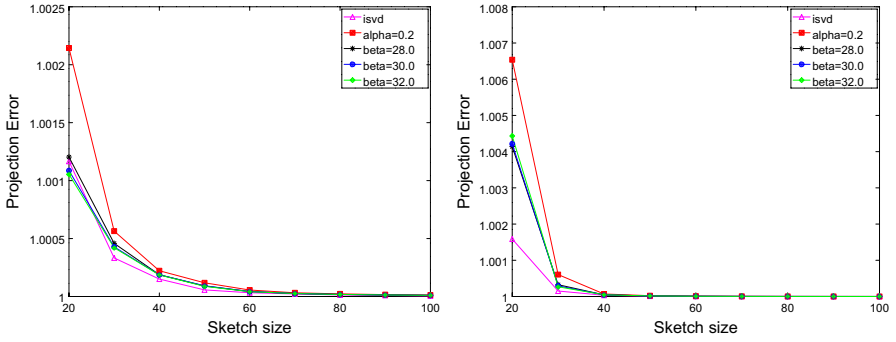

Fig. 10 Comparison of projection error for Birds (left), Spam (right)

higher error values than iSVD, for $l = 20$, β-FD has lower error than iSVD. In Fig. 11, the covariance error obtained by β-FD on PeMS indicates that for small values of l, β-FD ($\beta = 32$) has better performance than 0.2α-FD. For HAR dataset, the covariance error obtained for β-FD is smaller than that of α-FD for almost all values of l except for $l = 40, 50$, where both the algorithms have similar performance. In Fig. 12, the projection error obtained for β-FD is better than that of α-FD for both PeMS and

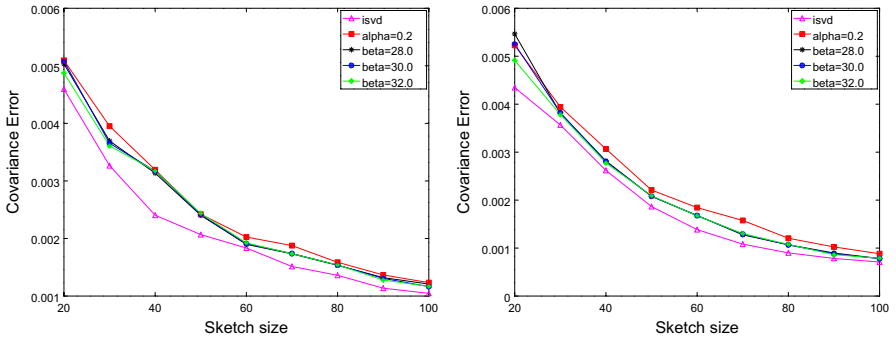

Fig. 11 Comparison of covariance error for PeMS (left), HAR (right)

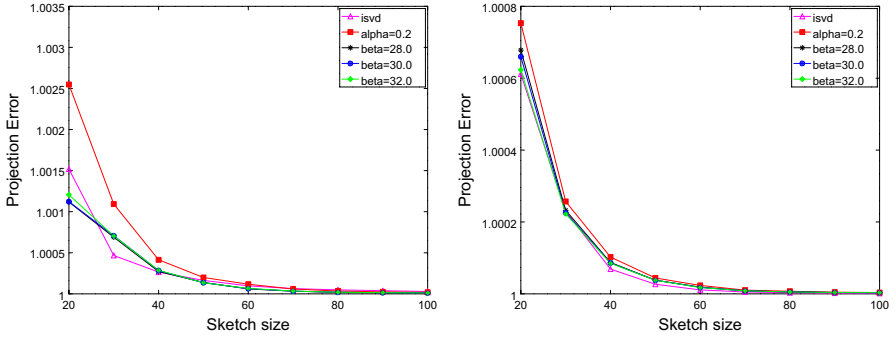

Fig. 12 Comparison of projection error for PeMS (left), HAR (right)

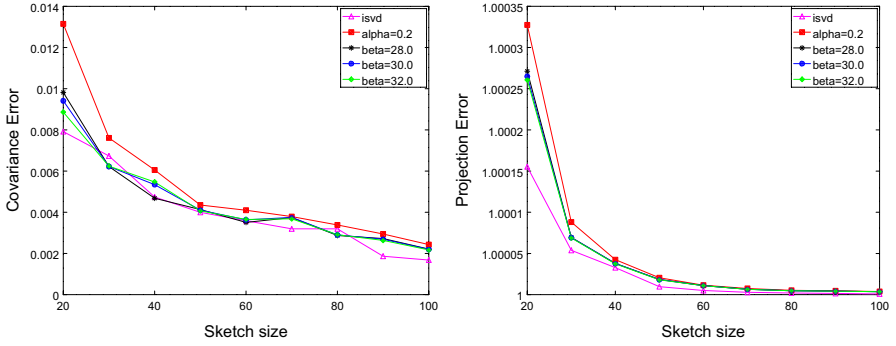

Fig. 13 Comparison of covariance (left) and projection (right) errors for Usenet

HAR datasets, and in the PeMS dataset, the error obtained by β-FD is lower than that obtained by iSVD for $l = 20$. In the case of Usenet dataset, the covariance error of β-FD is smaller than that of α-FD, particularly for $l \leq 60$. For $l \approx 30$, β-FD performs better than iSVD. The proposed algorithm has lower projection error for small sketch sizes ($l \leq 60$) when compared to α-FD as shown in Fig. 13.

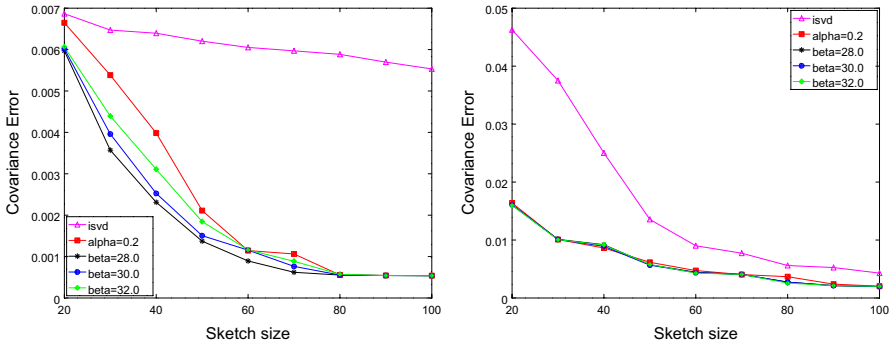

Fig. 14 Comparison of covariance error for adversarial (left), ConnectUS (right)

In the experiments, different values of β were tried and the β values for which the error was minimum was found to be $\{28, 30, 32\}$. By further investigation of the results it can be concluded that for all datasets with gradual and minor or no drift, the higher the value of β, the lower the error. In this case, the lowest error is obtained for $\beta = 32$.

5.3.2 Results for data with major and sudden drift

Adversarial and ConnectUS datasets: Recall the procedure for constructing the adversarial data where **m** and **n** are the two orthogonal subspaces onto which the two vectors **X** and **Y** are projected. The adversarial data is then constructed as the concatenation of those projections after they have been normalized. Due to the nature of the adversarial data, iSVD cannot adapt to the sudden and major drift in the data. The iSVD algorithm first encounters the data from **m**, and then those of **n**. In the course of the algorithm the rows from **n** will always be removed because such rows will always be associated with the smallest singular values, and hence iSVD will always perform poorly on such data with sudden and major drift. In the case of both α-FD and β-FD, the new rows will not be removed, but rather, it adjusts to this sudden concept drift by the *reduceRank* procedure.

In Fig. 14 it can be observed that iSVD performs much worse than both α-FD and β-FD for all values of l in both the datasets. In the adversarial dataset, for all values of l, β-FD algorithm has lower error than α-FD for l up to 80, after which both of them have similar error values. On the ConnectUS dataset, β-FD matches the performance of α-FD, whereas iSVD has larger error values for all values of l. In both these datasets, lowest values of error were obtained for smaller values of β. In particular, for $\beta = 28$, the lowest error was obtained.

It has been observed empirically, after trying different values of β, that for datasets with sudden and major concept drift, the lower the value of β, the lower the error obtained, and for data with gradual and minor or no drift, the higher the value of β, the lower the error obtained. A trade-off between the errors on the two different kinds of data was found to exist, and for $\beta \approx 28$, our algorithm performs better than α-FD.

5.4 Comparison of β-FD with α-FD

It has already been shown that α-FD empirically outperforms FD on all datasets and iSVD on datasets with sudden and major concept drift (Ghashami et al. 2014). In the previous section we further showed that β-FD outperforms α-FD on almost all datasets. In order to clearly understand the differences in error obtained between β-FD and α-FD, plots showing the percentage reduction in errors obtained for β-FD when compared to α-FD for all the datasets have been shown in Fig. 15. Here, $\beta = 28$ and $\alpha = 0.2$. The percentage of error reduction is calculated as follows.

$$Error_{pct} = \frac{E_\alpha - E_\beta}{E_\alpha} * 100 \tag{27}$$

where E_α is the covariance error obtained for α-FD, and E_β is the covariance error obtained for β-FD, for a particular value of β ($\beta = 28$).

In Fig. 15a, there is greater than 15% reduction in error for β-FD in the case of Rand30 dataset. For Rand50 dataset (Fig. 15b), the percentage of error reduction is 15 for sketch size 20, and it becomes more than 30% for $l = 30$. In both the random noisy datasets, the percentage of error reduction is minimal for $l > 60$. In the case of MNIST, Spam, PeMS, and HAR datasets, the error reduction increases as sketch size grows. In the case of Birds dataset, there is 20% error reduction for $l = 20$. In the adversarial data we see a significant ($\approx$40%) error reduction for sketch size 40 and in the ConnectUS dataset, the percentage error reduction is notable only for sketch size greater than 40. In summary, there is a good percentage error reduction for β-FD ($\beta = 28$) for data with sudden and major drift. On data with gradual and minor or no drift, the percentage of error reduction is sufficiently big for small sketch sizes. This means that, even for small sketch sizes, the proposed algorithm is able to compute a more accurate sketch than α-FD.

5.5 Comparison of β-FD with a simple baseline

In the case of data with major drift, there is a simple baseline algorithm which discards the current sketch and recomputes a new sketch whenever major drift is encountered. The intuition behind constructing such a baseline is that if the drift magnitude (between the old and new concepts) is too large, then it is better to discard the old concept and start afresh with the new concept. In this section we compare β-FD ($\beta = 28$) with a simple *baseline* for datasets with major drift. In order to detect major drift, we employ Eq. 3, with KL divergence as D and since the data arrives in a streaming manner, the indices of the data stream can be thought of as the time, t.

Using a basic windowing scheme it is possible to measure the KL divergence between the distribution of the historical data and the current data. The historical window includes a minimum number of data points that have arrived before time t whose probability distribution is then compared against the distribution of the current window, which includes the set of points that arrived after time t and within time $t + p$ for some fixed p. Measuring of drift happens at checkpoints and both the windows

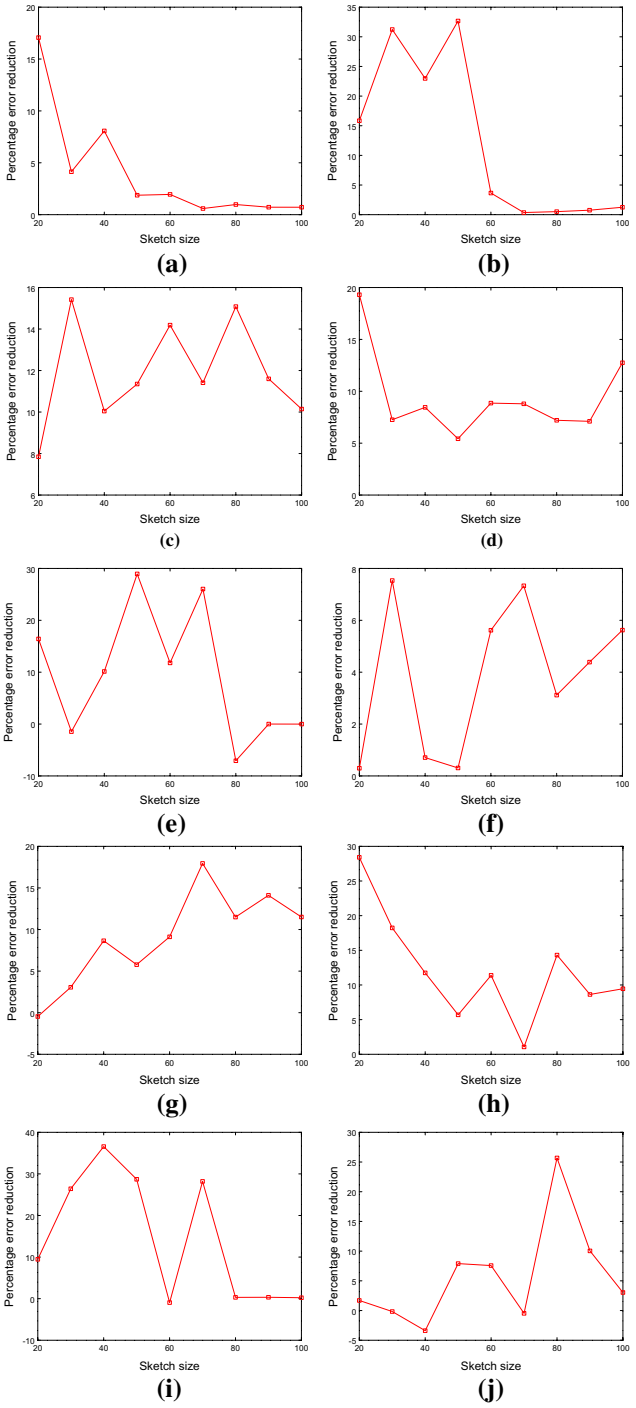

Fig. 15 Percentage reduction of relative covariance error of β-FD over α-FD for all datasets. **a** Rand30, **b** Rand50, **c** MNIST, **d** Birds, **e** Spam, **f** PeMS, **g** HAR, **h** Usenet, **i** Adversarial, **j** ConnectUS

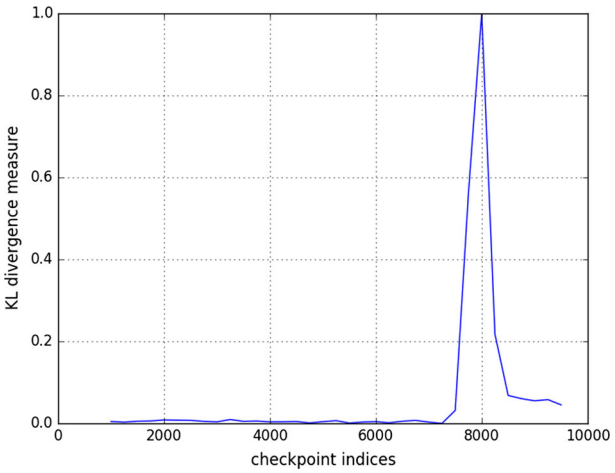

Fig. 16 Variation of KL measure for the adversarial data

advance forward with time and KL divergence of the distributions of the points in both the windows is computed. Whenever the drift magnitude crosses a threshold κ, a major drift is said to have occurred. The advantage of using this windowing scheme is that it can easily be integrated into any streaming algorithm like the simple *baseline*, and as and when a major drift occurs it can restart.

A simple baseline algorithm: This algorithm basically incrementally computes the SVD of the data and hence its sketch, and whenever a major drift occurs (corresponds to KL measure exceeding threshold κ), the current SVD is scrapped and the SVD of the new data (new concept) is computed. This is equivalent to discarding the current sketch and starting anew whenever a major drift occurs.

Figure 16 shows the KL divergence across the data stream for the Adversarial data. The KL measure at the index 7500 is 0.0, and it becomes 1 at index 8000, which is a sudden and a major drift. Here the drift magnitude is large at index 7500, and this rise of magnitude from 0 to 1 happens near instantaneously.

We compare our algorithm with the baseline. During the course of the simple baseline algorithm, we detect major drift (Eq. 3) using KL divergence, setting $\kappa = 0.6$. Figure 17 shows the comparison of β-FD algorithm with the baseline for various values of l for the Adversarial dataset. It can be seen that the baseline algorithm performs poorly in terms of the error obtained, when compared to β-FD.

The drift in the ConnectUS dataset is also sudden and major, as shown in Fig. 18. There are two regimes in the plot, the first being the region of major drift (drift magnitude >0.6) and the second being the region of minor drift. The major drift occurs suddenly (in the matter of a few points). The plot of error for various values of l for the baseline and β-FD is shown in Fig. 19. The error incurred by baseline algorithm is larger than β-FD. Note that the magnitude of the baseline error in this case is much smaller than that of the error obtained in case of Adversarial data, which can be accounted for the fact that the restart in the case of ConnectUS dataset happens early on as opposed to the late restart in the case of Adversarial dataset. This indicates

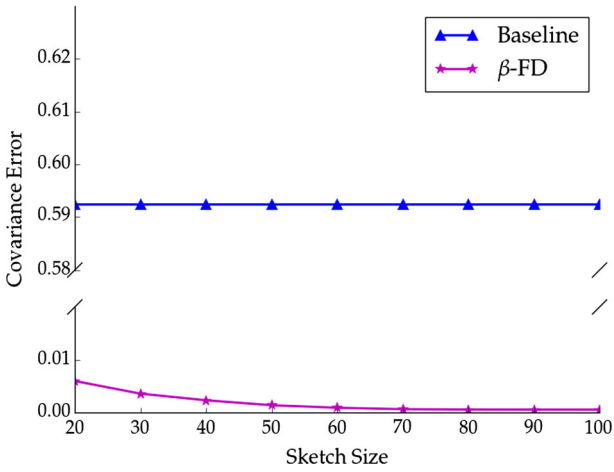

Fig. 17 Comparison of β-FD with the baseline algorithm for the adversarial data

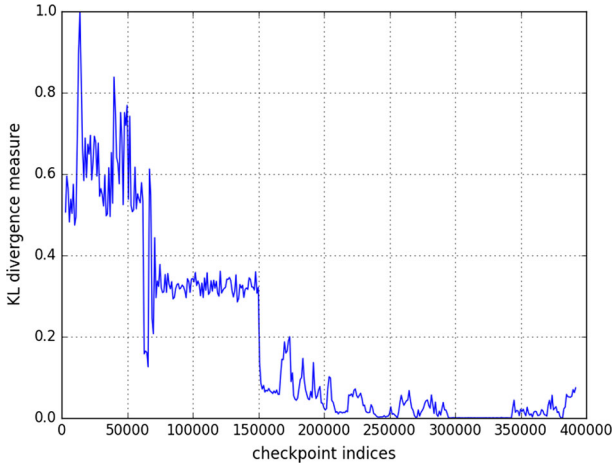

Fig. 18 Variation of KL scores for the ConnectUS dataset

that the simple baseline in fact removes more data than necessary to create a meaningful sketch.

5.6 Inferences

Some of the inferences that can be drawn from the experiments are as follows.

– The error obtained for datasets with sudden and major, gradual and minor or no concept drift by β-FD is lesser than that obtained for α-FD. These results act as evidence for the fact that the proposed solutions (as discussed in Sects. 2.4.1 and 2.4.2) actually work in practice. This shows the effectiveness of the proposed method in overcoming the limitations of the previous method (Sect. 2.4).

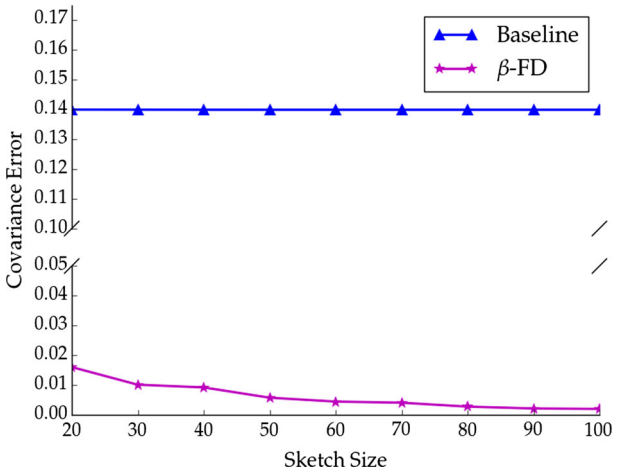

Fig. 19 Comparison of β-FD with the baseline algorithm for ConnectUS dataset

- Tuning the parameter β effectively improves the performance of the sketching algorithm as discussed in Sects. 2.4.1 and 3, thereby overcoming the limitations of α-FD. Particularly, for $\beta = 28$, our algorithm performs better than α-FD on all datasets.
- The proposed method performs significantly better than iSVD on datasets with sudden and major concept drift (Fig. 14). This shows that the strategies of smooth reduction is more effective in handling sudden and major concept drift as opposed to simply zeroing-out the least important singular value (Sect. 2.4.2). These types of datasets are handled more effectively by the proposed method when compared to the previous method, α-FD. In particular, the error obtained for the Adversarial data for β-FD is much lesser than that obtained by α-FD (Fig. 14).
- β-FD performs much better than a simple baseline algorithm in the case of data with major drift (Figs. 17, 19). The results indicate that the proposed algorithm is flexible enough to handle datasets with minor as well as major drift equally well.

6 Conclusion

Many matrix sketching techniques have been proposed in the past, one of which is a deterministic method called FD. It has good error guarantees, and also has a parameterized variant called α-FD. The iSVD algorithm has been found to have better performance in many real world datasets with gradual and minor or no drift when compared to FD and α-FD, but, it has two disadvantages. The first is the lack of any theoretical error guarantees and the second is its poor performance in the datasets with sudden and major concept drift. The α-FD algorithm has good error guarantees and it has better empirical performance than iSVD on data with sudden and major drift. However, α-FD has two limitations, namely, the restriction on the effective values of its parameter α and its constant factor reduction, both of which result in reduced empirical performance. A modification of α-FD called the β-FD algorithm is

proposed. The limitations of the previous method are overcome by using a parameter β and a scaling function that ensures variable reduction. The proposed algorithm has error guarantees similar to that of α-FD. Empirical results indicate that there exists a trade-off between the error values on data with sudden and major drift and data having gradual and minor or no drift for different parameter values. On many real-world as well as synthetic datasets, for $\beta \approx 28$, the proposed algorithm was found to perform better than α-FD. In the case of data with major drift we compared β-FD with a simple baseline algorithm. It was found that our algorithm gave significantly better results than the simple baseline which indicates that the proposed algorithm is flexible in handling the two categories of data considered in this work.

Acknowledgements The authors would like to thank the financial support offered by the Visvesvaraya Ph.D. Scheme for Electronics and Information Technology, Ministry of Electronics and Information Technology (MeitY), Govt. of India.

References

Achlioptas D, McSherry F (2007) Fast computation of low-rank matrix approximations. J. ACM (JACM) 54(2):9

Anguita D, Ghio A, Oneto L, Parra X, Reyes-Ortiz JL (2013) A public domain dataset for human activity recognition using smartphones. In: ESANN

Boutsidis C, Mahoney MW, Drineas P (2009) An improved approximation algorithm for the column subset selection problem. In: Proceedings of the twentieth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, pp 968–977

Brand M (2002) Incremental singular value decomposition of uncertain data with missing values. In: European conference on computer vision. Springer, Berlin, pp 707–720

Buss S (2016) Connectus data set Florida sparse matrix collection. http://www.cise.ufl.edu/research/sparse/matrices/Buss/connectus.html

Clarkson KL, Woodruff DP (2013) Low rank approximation and regression in input sparsity time. In: Proceedings of the forty-fifth annual ACM symposium on theory of computing. ACM, New York, pp 81–90

Cuturi M (2011) Fast global alignment kernels. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 929–936

Desai A, Ghashami M, Phillips JM (2016) Improved practical matrix sketching with guarantees. IEEE Trans Knowl Data Eng 28(7):1678–1690

Drineas P, Kannan R, Mahoney MW (2006) Fast monte carlo algorithms for matrices II: computing a low-rank approximation to a matrix. SIAM J Comput 36(1):158–183

Ghashami M, Phillips JM (2014) Relative errors for deterministic low-rank matrix approximations. In: Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, pp 707–717

Ghashami M, Desai A, Phillips JM (2014) Improved practical matrix sketching with guarantees. In: European symposium on algorithms. Springer, Berlin, pp 467–479

Ghashami M, Liberty E, Phillips JM, Woodruff DP (2016) Frequent directions: simple and deterministic matrix sketching. SIAM J Comput 45(5):1762–1792

Hall PM, Marshall AD, Martin RR (1998) Incremental eigenanalysis for classification. In: BMVC, vol 98. Citeseer, pp 286–295

Har-Peled S (2014) Low rank matrix approximation in linear time. arXiv preprint arXiv:1410.8802

Hoens TR, Chawla NV, Polikar R (2011) Heuristic updatable weighted random subspaces for non-stationary environments. In: 2011 IEEE 11th international conference on data mining (ICDM). IEEE, Washington, pp 241–250

Katakis I, Tsoumakas G, Vlahavas IP (2008) An ensemble of classifiers for coping with recurring contexts in data streams. In: ECAI, pp 763–764

Katakis I, Tsoumakas G, Vlahavas I (2010) Tracking recurring contexts using ensemble classifiers: an application to email filtering. Knowl Inf Syst 22(3):371–391
Kullback S, Leibler RA (1951) On information and sufficiency. Ann Math Stat 22(1):79–86
Lecun Y, Cortes C (2009) The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/
Levey A, Lindenbaum M (2000) Sequential Karhunen–Loeve basis extraction and its application to images. IEEE Trans Image Process 9(8):1371–1374
Liberty E (2013) Simple and deterministic matrix sketching. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, London, pp 581–588
Mahoney MW (2011) Randomized algorithms for matrices and data. Found Trends Mach Learn 3(2):123–224
Nelson J, Nguyên HL (2013) Osnap: faster numerical linear algebra algorithms via sparser subspace embeddings. In: 2013 IEEE 54th annual symposium on Foundations of Computer Science (FOCS). IEEE, Washington, pp 117–126
Sarlós T (2006) Improved approximation algorithms for large matrices via random projections. In: 2006 47th annual IEEE symposium on foundations of computer science (FOCS'06). IEEE, Washington, pp 143–152
Schlimmer JC, Granger RH (1986) Incremental learning from noisy data. Mach Learn 1(3):317–354
Tsymbal A (2004) The problem of concept drift: definitions and related work. Computer Science Department, Trinity College Dublin, Dublin 106(2)
Wah C, Branson S, Welinder P, Perona P, Belongie S (2011) The Caltech-UCSD birds-200-2011 dataset. Tech. rep, California Institute of Technology
Webb GI, Hyde R, Cao H, Nguyen HL, Petitjean F (2016) Characterizing concept drift. Data Min Knowl Discov 4(30):964–994
Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23(1):69–101
Woodruff DP et al (2014) Sketching as a tool for numerical linear algebra. Found Trends Theor Comput Sci 10(1–2):1–157