



Correlating Synchronous And Asynchronous Data Streams

Sudipto Guha
University of Pennsylvania
sudipto@cis.upenn.edu

D. Gunopulos
University of
California-Riverside
dg@cs.ucr.edu

Nick Koudas
AT&T Laboratories
koudas@research.att.com

ABSTRACT

In a variety of modern mining applications, data are commonly viewed as infinite time ordered data streams rather than finite data sets stored on disk. This view challenges fundamental assumptions commonly made in the context of several data mining algorithms.

In this paper, we study the problem of identifying correlations between multiple data streams. In particular, we propose algorithms capable of capturing correlations between multiple continuous data streams in a highly efficient and accurate manner. Our algorithms and techniques are applicable in the case of both synchronous and asynchronous data streaming environments. We capture correlations between multiple streams using the well known technique of Singular Value Decomposition (SVD). Correlations between data items, and the SVD technique in particular, have been repeatedly utilized in an off-line (non stream) data mining problems, for example forecasting, approximate query answering, and data reduction.

We propose a methodology based on a combination of dimensionality reduction and sampling to make the SVD technique suitable for a data stream context. Our techniques are approximate, trading accuracy with performance, and we analytically quantify this tradeoff. We present a thorough experimental evaluation, using both real and synthetic data sets, from a prototype implementation of our technique, investigating the impact of various parameters in the accuracy of the overall computation. Our results indicate, that correlations between multiple data streams can be identified very efficiently and accurately. The algorithms proposed herein, are presented as generic tools, with a multitude of applications on data stream mining problems.

Categories and Subject Descriptors

H.2.3 [Database Management]: Database Applications - Data Mining

Keywords

Singular Value Decomposition, Data Streams, Approximate Computation

1. INTRODUCTION

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. SIGKDD '03, August 24-27, 2003, Washington DC, USA. Copyright 2003, ACM 1-58113-737-0/03/0008...\$5.00

In many modern applications, data are commonly viewed as an infinite, possibly ordered data sequences rather than a finite data set stored on disk. Such a view, challenges fundamental assumptions related to the analysis and mining of such data, for example, the ability to examine each data element multiple times, through random or sequential access. In many traditional applications, such as networking and multimedia, as well as in new and emerging applications, like sensor networks and pervasive computing, this view of application data is prevalent. Commonly such (potentially) infinite ordered sequences of data, are referred to as *data streams*.

Networking infrastructure, such as routers, hubs and traffic aggregation stations, produce vast amounts of performance and fault related data in a streaming fashion. Such information is vital for network management operations and need to be collected and analyzed online. Network operators require precise characterizations of the temporal evolutions of such data and/or identification of abnormal events. Sensor networks are becoming increasingly commonplace. The vision of pervasive computing involves hundreds of autonomous devices collecting data (such as highway traffic, temperature etc) from dispersed geographic locations. Such data, is subsequently made available to inter operating applications which utilize them to make intelligent decisions.

Data elements in real data sets are rarely independent. Correlations commonly exist and are primarily due to the nature of the applications that generate the data. In settings involving multiple data streams, correlations between stream elements are encountered as well. Effectively quantifying correlations between multiple streams is of vast importance to a variety of applications.

In this paper, we study fast and efficient techniques to identify correlations between multiple data streams. We focus on a fundamental form of correlations between multiple streams, namely linear correlations and we adapt a technique widely utilized for identifying linear correlations. In particular we adapt the Singular Value Decomposition (SVD) [2], in a data stream context.

2. BACKGROUND AND DEFINITIONS

2.1 Data Stream Models

A data stream S is an ordered sequence of data points that can be read only once. Formally, a data stream is a sequence of data items $\dots x_i, \dots$ read in increasing order of the indices i . On seeing a new item x_i , two situations of interest arise: either we are interested in all N items seen or we are interested on a sliding window of the last n items, x_{i-n}, \dots, x_i . The former is defined as the standard data stream model and the latter as a *sliding window* data stream model. The central aspect of data stream computation is modeling in small space relevant to the parameter of interest N or n .

Data points in a single stream, have the form (i, Δ) representing a sequence of updates or modifications (increment or decrement) of a vector U . In the case of an update $U[i] = \Delta$.

Similarly, for modifications $U[i] = U[i] + \Delta$. Notice that an evolving time series can be represented by elements of updates (i, Δ) with the restriction that data arrives in increasing order of i , (indicating time of observation). Thus, for a time series model, Δ corresponds to the observed value at time i .

Let S_1, \dots, S_{m-1}, S_m be a collection of m data streams. In the applications we envision, $m \leq n$; that is, the number of streams is usually much smaller than the number of items or points of observation in each stream. We use the notation $A[i][j]$ to refer to the j -th point of the i -th stream. Thus, we treat the data streams as a matrix, A . Notice that our treatment of the streams as a matrix A is purely conceptual. Our techniques neither require nor materialize matrix A at any point. At each point in time, data elements (tuples) (i, t, Δ) appear, which denote that in the t^{th} observation of stream i , the entry $A[i][t]$ is either updated to Δ or modified (incremented or decremented) by Δ . In the sliding window model, at time τ we are interested in $A[i][t']$ for all $\tau - n \leq t' < \tau$; we refer to all other items as expired.

If there are no restrictions on the tuples (i, t, Δ) then the streams are considered *asynchronous*. For example, we can observe a sequence $\dots, (1, 3, 3), (2, 3, 1), (1, 1, 5), \dots$, for two streams which denotes that the streams are modified arbitrarily without any coordination between successive tuples. Assuming a collection of m streams, we will say that these streams are *synchronous* if at every time t , m values, each corresponding to one of the streams arrive. It is not necessary that the tuples be ordered according to the stream i , but it is required that the tuples be ordered in time. If a tuple (i, t, Δ) is not present at time t for stream i , the tuple $(i, t, 0)$ is assumed present, allowing streaming of 'sparse' streams.

Given this structure, observe that modifications are superfluous in synchronous streams since all modifications to the element $A[i][t]$ (t^{th} element of i^{th} stream) have to be grouped together. In a sense Δ values in the tuple (i, t, Δ) in synchronous streams, always expresses updates. Since we wish to present stream algorithms for both asynchronous and synchronous streams, we will proceed with the assumption of arbitrary arrivals of (i, t, Δ) (no restriction on t) assuming that Δ values express modifications. This, naturally expresses asynchronous as well as (suitably restricted requiring ordered t values and Δ values expressing updates) synchronous streams.

2.2 Correlations and SVD

The *Singular Value Decomposition* (SVD) is a very popular technique to identify correlations, with many applications in signal processing, visualization and databases. Informally the SVD of a collection of points (high dimensional vectors) identifies the 'best' subspace to project the point collection in a way that the relative point distances are preserved as well as possible under linear projection. Distances are quantified using the L_2 norm. More formally:

THEOREM 1 (SVD). *Let $A \in R^{m \times n}$ be an arbitrary m -by- n matrix with $m \geq n$. Then we can write $A = U\Sigma V^T$ where U is m -by- r and satisfies $U^T U = I$, V is m -by- r and satisfies $V^T V = I$ and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$, where $\sigma_1 \geq \dots \geq \sigma_r \geq 0$. The columns u_1, \dots, u_r of U are called left eigenvectors. The columns v_1, \dots, v_r of V are called right eigenvectors. The σ_i are called eigenvalues and r is the rank of matrix A , that is the number of linearly independent rows (if $m \leq n$, the SVD is defined by considering A^T).*

For each eigenvalue there is an associated eigenvector; commonly we refer to the largest eigenvalue as the *principal eigenvalue* and to the associated eigenvector as the *principal eigenvector*¹. This theorem has an intuitive geometric

¹Notice that if u is the principal eigenvector, $\|Au\| \geq \|Au'\| \forall u', \|u'\| = 1$.

interpretation. Given any m -by- n matrix A , think of it as a mapping of a vector $x \in R^n$ to a vector $y \in R^m$. Then we can choose one orthogonal coordinate system for R^n (where the unit axes are the columns of V) and another orthogonal coordinate system for R^m (where the unit axes are the columns of U) such that A is diagonal (Σ), i.e., maps a vector $x = \sum_{i=1}^r \mu_i v_i$ to a $y = Ax = \sum_{i=1}^r \sigma_i \mu_i u_i$.

According to theorem 1, $A = \sum_{i=1}^r \sigma_i u_i v_i^T$. Matrix A has small rank when data are correlated ($r \leq m$). Consequently, using $k \leq r$ eigenvectors (projecting to a subspace of dimension k) we have $A \approx \sum_{i=1}^k \sigma_i u_i v_i^T$. Such a projection introduces error which is quantified by $\|A - \sum_{i=1}^k \sigma_i u_i v_i^T\|$. The guarantee of SVD however, is that among all possible k dimensional projections, the one derived by SVD has the minimum error, i.e., minimizes $\|A - \sum_{i=1}^k \sigma_i u_i v_i^T\|$. The basis of the "best" k -dimensional subspace to project, consists of the k left eigenvectors of U . Essentially, this subspace identifies the *strongest linear* correlations in the underlying data set.

DEFINITION 1 (LINEAR CORRELATIONS). *Given a matrix A , let $U\Sigma V$ be its Singular Value Decomposition; we refer to the set of linear combinations of the k eigenvectors, corresponding to the k largest eigenvalues of A as the k strongest linear correlations in A .*

The relative magnitude of the eigenvalues determine the relative "strength" of correlations along the direction of the associated eigenvectors. This means that if one eigenvalue, σ is very large compared to the others, the eigenvector corresponding to σ signifies a stronger linear correlation towards the direction of the eigenvector in the subspace spanned by the k strongest linear correlations. We formalize this intuition by quantifying the relative magnitude of the eigenvalues with the following definition:

DEFINITION 2 (ϵ -SEPARATED EIGENVALUES). *Let A be a matrix of rank r and $\sigma_1, \dots, \sigma_r$ its eigenvalues. Assume, without loss of generality, that $|\sigma_1| \geq \dots \geq |\sigma_r|$. The ϵ -separating value for the collection of eigenvalues, is the smallest $\epsilon \geq 0$, such that $\forall i, 1 \leq i < r, |\sigma_i| \geq (1 + \epsilon)|\sigma_{i+1}|$. For this ϵ , we say that the eigenvalues are ϵ -separated.*

Notice that such an ϵ always exists; its magnitude however, specifies how significant are the eigenvectors in the linear combination. If ϵ is small, eigenvalues are close in magnitude and all the eigenvectors are significant. If ϵ is large, the linear correlations along the directions of the eigenvectors associated with the largest eigenvalues are more significant in the linear combination.

Given a matrix A m -by- n there exists a $O(m^2 n)$ algorithm to compute the SVD of A using the following celebrated theorem (see [2] for full details and a proof)

THEOREM 2. *Let $A = U\Sigma V^T$ be the SVD of the m -by- n matrix A , with eigenvalues σ_i and orthonormal eigenvectors u_i , where $m \geq n$. (There are analogous results for $m \leq n$.) The eigenvalues of the symmetric matrix AA^T are σ_i^2 . The left eigenvectors u_i are corresponding orthonormal eigenvectors of the eigenvalues σ_i^2 .*

The benefit of the above theorem appears in computation of SVD of sparse matrices. If the number of entries in a column is $r \ll m$ then the matrix AA^T can be computed in time $O(r^2 n)$ which is $O(r)$ times the number of nonzero entries in the matrix. The pseudo code is provided in Figure 1. The algorithm remains a good candidate for computing *incremental SVD* since the number of operations performed on an update is (on an average) the number of non-zero entries in a column.

```

Algorithm NaiveSVD( $A, M, U, \Sigma, V, T$ ) {
 $A \in R^{m \times n}, M = AA^T \in R^{m \times m}$ ,
 $U, V$  the set of left, right eigenvectors
 $\Sigma$  the eigenvalues,  $T = (i, t, \Delta)$  is current input
  for all nonzero entries in column  $t$ , i.e.  $\{j | A[j][t] \neq 0\}$  do {
     $M[i][j] += \Delta A[j][t]$  if  $j \neq i$ 
     $M[i][j] += 2\Delta A[j][t] + \Delta^2$  if  $j = i$ 
     $A[j][t] += \Delta$ 
    observe that the above for synchronous streams
    becomes  $A[j][t] = \Delta$  and  $M[i][i] = \Delta^2$ 
    under the assumption that  $A[j][t]$  is initially
    0 and changed only once.
  }
  SVD( $M, U, \Sigma, V$ ) /* our favorite SVD algorithm */
}

```

Figure 1: Algorithm NaiveSVD. Function SVD() can implement any SVD technique

2.3 Low Rank Approximations

The quadratic space requirement of $O(m^2)$ can be prohibitive and the approach is expensive even if we are interested in just the top eigenvector. The computation for non sparse matrices requires $O(m^2n)$ no matter if we are interested in just the topmost eigenvector. A step in this direction is the following column sampling result of [4, 3].

THEOREM 3. *Given a matrix A with columns C_i , if with probability² $\|C_i\|_F^2 / \|A\|_F^2$ we sample³ $O(k/\epsilon^2)$ columns then we can construct a matrix D of rank k such that for any matrix D^**

$$\|A - D\|_F^2 \leq \|A - D^*\|_F^2 + \epsilon \|A\|_F^2$$

[3] suggests alternate “test and sample” schemes for practical considerations, thus making the algorithm multi pass. A problem of the above result is that the approximation of the matrix need not be a good approximation of the eigenvalue which denotes the strength of the correlations. For example suppose we are interested in the topmost eigenvalue σ_1 . Following the results of [3] one can relate $\min_{D^*} \|A - D^*\|_F^2 = \sigma_1^2$. Thus, $\|A - D\|_F^2$ gives us an estimate of σ_1 . If $\|A\|_F$ is large, as is the case in non-sparse matrices, the above is a bad approximation since $\|A\|_F^2$ can be m times σ_1^2 . Thus, ϵ cannot be a constant to provide a good guarantee for the topmost eigenvalue. The result is useful in the context of approximating the entries of a matrix and as pointed out by the authors in [3], the approach is used if the matrix is sparse.

3. STREAM SVD

We will present an approximate technique to obtain the k largest eigenvalues and associated eigenvectors trading accuracy for computation speed. We will first present the case for the principal eigenvalue and the associated principal eigenvector, and then generalize to arbitrary k eigenvalues and eigenvectors. Due to space limitations, all the proofs of theorems and lemmas are omitted.

Given a matrix $A \in R^{m \times n}$ the set of all k correlations is defined as the set of linear combinations of the left eigenvectors corresponding to the k largest eigenvalues. Recall that u is a left eigenvector with eigenvalue σ if and only if $u^T A = \sigma u^T$. Theorem 1 asserts that we can find a set of orthonormal eigenvectors of any matrix A . The number of such vectors is the rank r of the matrix concerned. Before we proceed in the discussion let us assume that the

²The subscript indicates that the norm is Frobenius. $\|A\|_F^2$ is the sum of squares of the elements in the matrix A .

³If nice bounds on the ratios are known then sampling can be performed in one pass else in two.

eigenvectors of A are $\bar{u}_1, \bar{u}_2, \dots, \bar{u}_r$ with respective eigenvalues $\sigma_1, \sigma_2, \dots, \sigma_r$. Let us assume, without loss of generality that $|\sigma_1| \geq |\sigma_2| \geq \dots \geq |\sigma_r|$.

Our methodology will make use of the Johnson-Lindenstrauss Lemma (JL Lemma) [5] to reduce the dimension in a Euclidean space.

LEMMA 1 (JL LEMMA). *Given a set of N vectors V in space R^n , if we have a matrix $S \in R^{s \times n}$ where $s = O(\frac{1}{\epsilon^2} \log N)$ such that each element S_{ij} is drawn from a Gaussian distribution, appropriately scaled, for any vector $x \in V$, then $\|x\|_2 \leq \|Sx\|_2 \leq (1 + \epsilon)\|x\|_2$ holds true with vanishingly high probability, $1 - o(1/N)$.*

We discuss issues in computation and storage of maintaining AS^T in Section 3.1. For the present we investigate how matrix AS^T allows us to compute SVD.

Informally, the JL lemma states that if we distort vectors of dimensionality n with a matrix whose elements are suitably chosen from a Gaussian distribution we can preserve the relative distances between the vectors in the resulting space (of dimensionality s) upto $(1 + \epsilon)$ with arbitrarily high probability. Intuitively, suppose every vector is represented by a line segment starting from the origin. The length of the vector is the distance between the origin and the endpoint of the vector. The intuition behind the algorithm is that if we preserve distances between points (the origin and the endpoints of the vectors), then we preserve the length of the vectors.

3.0.0.1 The Single Eigenvalue case.

We make the simple observation that $\|x\|_2 = \|x^T\|_2$. So the JL lemma rewrites to,

$$\|x^T\|_2 \leq \|(Sx)^T\|_2 = \|x^T S^T\|_2 \leq (1 + \epsilon)\|x^T\|_2 \quad (1)$$

Both lemma 1 and theorem 2 are concerned with linear operations on the underlying vector space. It appears natural to first apply lemma 1 on A to reduce the dimensionality and then apply SVD on the “smaller” matrix obtained. This could be beneficial, because we will be running SVD on a much smaller matrix. Under such an approach, the relationship between the eigenvalues and eigenvectors of A before and after the application of lemma 1 needs to be established. Let us assume that the entries in matrix A have precision polynomially bounded in n . This gives rise to the following:

LEMMA 2. *Suppose \bar{u}_1 is the principal left eigenvector of A and u the principal left eigenvector of AS^T for a matrix S satisfying the JL Lemma with $s = O(\frac{1}{\epsilon^2} \log n)$. Then $\|\bar{u}_1^T A\|_2 \leq (1 + \epsilon)\|u^T A\|_2$*

Let σ_1 the principal eigenvalue of AS^T . From lemma 2 it is evident that $|\sigma_1| \leq (1 + \epsilon)|\sigma_1|$. Thus, the first eigenvalue is approximated within $(1 + \epsilon)$ factor in magnitude by application of lemma 1.

3.0.0.2 The Single Eigenvector case.

Lemma 2 shows that instead of computing the SVD of the matrix AA^T applying theorem 2, we can compute the SVD of AS^T to get a vector such that the columns of A have a large projection along it. The dimension of the matrix AA^T is $m \times m$ whereas the dimension of AS^T is $m \times \frac{1}{\epsilon^2} \log n$. For large m compared to $s = \frac{1}{\epsilon^2} \log n$, one has achieved a significant saving in computing the SVD. In particular the time to perform SVD has been reduced from $O(m^3)$ to $O(ms^2)$. Also we have saved the space and update time in the data stream context, from $O(m^2)$ to $O(ms)$.

Lemma 2 shows that the projections of a matrix are preserved under the application of lemma 1. We now show what is the quality of the approximation obtained to the actual

principal eigenvector. A measure of quality of approximation of the principal eigenvector, is the inner product with the actual principal eigenvector. Assuming all vectors are represented with unit length, a large value of the projection indicates a better approximation. Notice that such an approximation is meaningful only if the principal eigenvector is *unique*. Consider the case of a matrix A with $|\sigma_1| \approx |\sigma_2|$. Then any linear combination of \bar{u}_1 and \bar{u}_2 , say $u = a\bar{u}_1 + b\bar{u}_2$ (where $a^2 + b^2 = 1$ to preserve length of $\|u\|_2 = 1$) is a principal eigenvector, since there are a lot of vectors preserving the variation in the data, in this case. To see this, observe that in this case

$$\|u^T A\|_2^2 = u^T A A^T u = a^2 \sigma_1^2 + b^2 \sigma_2^2 \geq \min(\sigma_1^2, \sigma_2^2)$$

This is best illustrated if the data are uniformly distributed along a circle; any vector in the plane containing the circle is a good eigenvector. To clarify the situation, we assume that there is a significant linear trend in the data. This means that the eigenvalues are separated in magnitude. In case of the principal eigenvector this would imply $|\sigma_1| \gg |\sigma_2|$; we will address multiple eigenvectors in the subsequent subsections. In particular assume $|\sigma_1| = (1 + \delta\epsilon)|\sigma_2|$ for some $\delta > 4$.

For two vectors u, v , let $\langle u, v \rangle$ denote their inner product. If σ_1, σ_2 are the first and second eigenvalues and \bar{u}_1, \bar{u}_2 the associated eigenvectors, then

$$(1 + \epsilon)^{-2} \sigma_1^2 \leq \|u_1^T A\|_2^2 = \sum_i \langle u_1, \bar{u}_i \rangle^2 \sigma_i^2 \leq \langle u_1, \bar{u}_1 \rangle^2 \sigma_1^2 + (1 - \langle u_1, \bar{u}_1 \rangle^2) \sigma_2^2$$

since the coefficients $\langle u_1, \bar{u}_i \rangle$ represent the projection of u_1 to an orthogonal basis defined by $\{\bar{u}_i\}$, the sum of their squares evaluate to 1. Thus $\sum_{i \neq 1} \langle u_1, \bar{u}_i \rangle^2 = 1 - \langle u_1, \bar{u}_1 \rangle^2$. The above rewrites to

$$\langle u_1, \bar{u}_1 \rangle^2 \geq \frac{1}{(1 + \epsilon)^2} \frac{\sigma_1^2 - (1 + \epsilon)^2 \sigma_2^2}{\sigma_1^2 - \sigma_2^2} \geq \frac{1}{(1 + \epsilon)^2} - \frac{1}{\delta} \quad (2)$$

For a specific value of ϵ , equation 2 shows the quality of the approximation to \bar{u}_1 obtained. Notice that if $\delta \gg \epsilon$ (that is, the strength of linearity is greater than the precision lemma 1 guarantees) then $\langle u_1, \bar{u}_1 \rangle^2 \approx (1 + \epsilon)^{-2}$ which approaches 1. Thus, if the first two eigenvalues are $\delta\epsilon$ -separated, u_1 the approximated eigenvector and \bar{u}_1 the true eigenvector are very close. Effectively this establishes that if there is a significant linear trend in the data, performing SVD on matrix AS^T as opposed to matrix AA^T results in the same principal eigenvector. Smaller values of ϵ increase the time to compute the SVD of matrix AS^T , but yield a better approximation to the principal eigenvector and vice versa.

LEMMA 3. *If the data have a unique strong linear correlation, we can approximate the principal eigenvector.*

It is evident, that to guarantee a good approximation of the eigenvectors we have to compute at a greater precision than we need to identify the eigenvalues. That is ϵ , the precision set by lemma 1 has to be significantly smaller than the separating value of the eigenvalues.

Reasoning in a similar fashion a related approximation bound can be obtained for multiple eigenvalues and eigenvectors of the original matrix. We omit the details due to space limitations.

3.1 Discussion

The analysis of the previous section established that it is possible to compute eigenvalues and eigenvectors of a matrix A (of size $m \times n$) up to desired accuracy, by computing the SVD decomposition (using any applicable technique) of a much smaller matrix AS^T (of size $m \times s$). This could have significant performance benefits, independently of the

specific technique used to compute the SVD, since the procedure would operate on a much smaller matrix.

Matrix S is populated initially from a suitably scaled Gaussian distribution in accordance to lemma 1. The full matrix S is not realized, instead it is stored as a collection of s hash functions $h[j]$ such that $S[j][t] = h[j](t)$. This is one of the central techniques in streaming computation and [1] phrase the inner product $M[i][j] = \sum_t A[i][t]S[j][t]$ as *sketches* of the data.

Thus, as new stream elements arrive, matrix AS^T can be updated in a very efficient fashion. Let us first assume that we are in the standard stream model. For synchronous streams a single tuple (i, t, Δ) arrives for element $A[i][t]$ and the correct value $A[i][t]S[j][t]$ gets added to $M[i][j]$. For the asynchronous case the value $A[i][t]$ accepts (possibly multiple) modifications. But the contribution to $M[i][j]$ over all the modifications is again $A[i][t]S[j][t]$ which is the correct value. The entire procedure is presented in Figure 2. Notice that updates/modifications are provided in an incremental fashion to matrix AS^T , and that matrix A is not explicitly materialized.

The problem with the computation of Figure 2 is that although the SVD computation performed is expected to be faster (because it operates on a smaller matrix), one still has to perform the computation each time matrix AS^T changes. This is required to assure that the eigenvector and eigenvalues maintained stay within desired accuracy levels.

```

Algorithm MapSVD((i, t, Δ), M, U, Σ, V, P) {
  M = AST ∈ Rm × s, U ∈ Rm × s, Σ ∈ Rs × s
  V ∈ Rs × s, P ∈ Rm
  S ∈ Rs × n in accordance to lemma 1, it is a product of
  suitable hash functions, only the functions are stored
  T = (i, t, Δ) is current input (representing A[i][t])
  for (j = 0; j < s; j++) {
    M[i][j] = M[i][j] + ΔS[j][t]
    /* For synchronous streams M[i][j]+ = A[i][t]S[j][t],
       resulting in computation of AST. For asynchronous
       streams the same result is arrived at since A[j][t]
       is the sum of the modification */
  }
  SVD(M, U, Σ, V) /* favorite SVD algorithm */
}

```

Figure 2: Algorithm *MapSVD*. For the standard stream model

3.2 Recomputations and Sampling

We will develop a sampling strategy that will select stream tuples and periodically apply SVD while, at the same time, is able to preserve the quality of the underlying eigenvectors and eigenvalues obtained.

Suppose the stream has not changed significantly from a certain time when we computed the SVD for it. Then, the matrix corresponding to the stream has also not changed significantly. Suppose we recomputed the SVD last when the matrix corresponding to the stream was A_1 . Suppose the stream currently corresponds to matrix A . These matrixes are used conceptually only; in practice we never store them. Suppose the two matrixes agree almost everywhere, and thus their eigenvectors/values agree as well. This is captured by the following lemma:

LEMMA 4. *If $\|y^T A_1 S^T\|_2 = \sigma$ and $\|y\|_2 = 1$ then $\|y^T A\|_2 \geq \sigma/(1 + \epsilon) - \|A_1 - A\|_F$. $\|A_1 - A\|_F^2$ is the square of the Frobenius norm of $A_1 - A$ and is equal to the sum of squares of all elements.*

This means that if y was an eigenvector with a large projection in A_1 and $\|A_1 - A\|_F$ is small compared to σ , then y still has a large projection. In other words it is an approximate eigenvector. We first show that

LEMMA 5. Suppose we computed SVD for the stream which corresponds to the matrix A_1 at time t_1 and did not recompute SVD since. Suppose after that we saw tuples (i, t, Δ) and currently the matrix corresponding to the stream is A . Suppose further that no tuple expired⁴, if

$$\sum_{(i,t,\Delta) \text{ seen since } t_1} |\Delta| = D$$

then $\|A_1 - A\|_F \leq D$.

If we do not recompute the SVD and $\|A_1 - A\|_F$ is small compared to σ the eigenvectors of A_1 are still reasonable for our current stream matrix A . Suppose we are interested in preserving the principal eigenvector (for other eigenvectors the discussion is similar). Since we are interested in $1 \pm \epsilon$ approximation, we will have to ensure that $D \sim \epsilon\sigma_1$. An excellent way to achieve this is by randomly recomputing the SVD depending on the magnitude $|\Delta|$ seen. If $|\Delta|$ is large compared to $\epsilon\sigma_1$ we should choose to recompute, otherwise we would not. Thus the recomputation should be done with probability $|\Delta|/(\epsilon\sigma_1)$.

The ϵ factor in the probability ensures that after we have seen enough new information which satisfies $\sum |\Delta| \geq \epsilon\sigma_1$ we would have very likely (probability $1 - \frac{1}{e} \sim 0.63$) have recomputed the SVD. If we did not, then by the time $\sum |\Delta| \geq 2\epsilon\sigma_1$, we would have recomputed the SVD with probability $1 - \frac{1}{e^2} \sim 0.86$. The probability of not having computed the SVD for long, decreases exponentially. The Expected value is $1.4\epsilon\sigma_1$. Thus from Lemma 4 and Lemma 5 the principal eigenvector of $A_1 S^T$ would have a projection on A which is at least $1 - O(\epsilon)$ with some high probability.

4. THE STREAMSVD ALGORITHM

The sampling procedure introduced leads to an effective way to save on the number of times the SVD is computed. Instead of computing the SVD of matrix AS^T every time an item arrives, in accordance to lemma 4, we can compute it less often and still get a good approximation to eigenvectors and eigenvalues of matrix A .

Combining the results of Section 3 we can now realize efficient algorithms for maintaining the SVD decomposition on the various stream models, namely the standard and the sliding window stream model. The algorithm is provided in Figure 4 for the case of the sliding window model. The algorithm for the standard model is the same, there is no expiry and that condition is never used.

The algorithm starts from *MapSVD* and probabilistically recomputes the SVD depending on the magnitude $|\Delta|$ of the value seen compared to the eigenvalue σ_1 (assuming that we are interested in the topmost eigenvalue; if interested in all the k -th largest eigenvalues, we substitute σ_1 with σ_k). For the case of the synchronous model, the sampling procedure breaks the stream into several sub-matrices (Figure 3), $B_1 \dots B_c$ depending on when we sample. The sub-matrix B_1 starts at time t_1 when we sampled in the probabilistic step in STREAMSVD and ended when we sampled the next time (at t_2). We store the products of the sub matrices $B_u S^T$ in the blocks M^u in the algorithm STREAMSVD. For the standard streaming model it is easy to see that $\sum_u M^u$ is the entire inner product, namely matrix M . For sliding window streams if $t_1 < \tau - n \leq t_2$ then the block B_1 is partially relevant – some of its entries have expired. Now the sum of the $|\Delta|$ for the entries in each sub matrix B^u is $\Theta(\epsilon\sigma_1)$ as follows from the discussion in the previous section, since we did not recompute the SVD in the middle. If we computed the SVD last when the matrix was A_1 using a certain number of blocks and none of the blocks expired (otherwise we would recompute SVD) – the two matrices A_1 and A agree everywhere except in the current block. Now the $\sum |\Delta|$ of

⁴Always true in standard streaming model

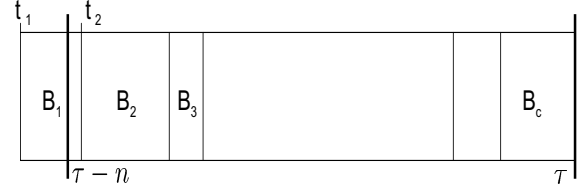


Figure 3: The structure of blocks created by StreamSVD for the case of the synchronous sliding window stream model

each block is $1.4\epsilon\sigma_1$. By Lemma 5 we have a $(1 \pm 1.4\epsilon)$ approximation. Therefore the eigenvalue is preserved.

LEMMA 6. The maximum number of blocks created in case of synchronous sliding window streams is at most $O(m/\epsilon)$.

The above follows from the fact that we have an estimate of the Frobenius norm of the blocks related to σ_1 and likewise the Frobenius norm A is related to σ_1 . The proof is completed by relating the norms of the blocks to norms of A .

The case of asynchronous streams is more involved. Since the data do not arrive in order, the pieces of matrix whose inner product is in the different blocks overlap. The eigenvalue is still preserved up to $1 \pm 1.4\epsilon$. A lemma analogous to lemma 6 can be proved under certain restrictions. We omit details due to lack of space.

```

Algorithm STREAMSVD( $(i, t, \Delta), M, U, \Sigma, V, P$ ) {
   $M = AS^T \in R^{m \times s}, U \in R^{m \times s}, \Sigma \in R^{s \times s}$ 
   $V \in R^{s \times s}, P \in R^m, S \in R^{s \times n}$  as in lemma 1
   $\sigma_1$  largest eigenvalue of  $M$  computed in a previous
  invocation of STREAMSVD, Current time is  $\tau$ 
  The inner product  $\sum_t A[i][t]S[j][t]$  is maintained through at
  most  $c$  blocks where  $\sum_u M^u[i][j] = \sum_t A[i][t]S[j][t]$ 
  Block  $M^c$  is Current, On arrival of  $(i, t, \Delta)$ , with  $t \geq \tau - n$  {
    If  $((\text{stamp of } M^1 \text{ is } \tau - n) \text{ or } (\text{with probability } O(\frac{|\Delta|}{\epsilon\sigma_1})))$  {
      Block  $M^c$  is closed with stamp  $\tau$ 
      If  $(\text{stamp of } M^1 \text{ is } \tau - n)$  { /*  $M^1$  expires */
        for  $(u=1; u < c; u++)$ 
           $M^u \leftarrow M^{u+1}$ 
         $c \leftarrow c - 1$ 
      }
      Start a new block  $M^{c+1}$  and set it Current
      Recompute the SVD( $M, U, \Sigma, V$ ).
      /* use favorite algorithm */
    }
  }
  for  $(j = 0; j < s; j++)$ 
    Current Block $[i][j] += \Delta S[j][t]$ 
}

```

Figure 4: Algorithm STREAMSVD for the sliding window model

Algorithm *StreamSVD* is given in Figure 4 for the case of the sliding window stream model. A similar algorithm can be designed for the case of the standard stream model.

Independently, this sampling step could be applied to algorithm *NaiveSVD* surpassing the dimensionality reduction step. This would provide an $(1 - \epsilon)$ approximation to the eigenvalues, for some $\epsilon \geq 0$. Following reasoning related to that in Section 3 the eigenvectors are preserved well also. Indeed we explore this option for algorithm *NaiveSVD* in section 5.

5. EXPERIMENTAL EVALUATION

In this section we present a performance analysis of the algorithms and techniques discussed thus far. We seek to

quantify the benefits both in terms of accuracy and performance of the proposed techniques. We present the data sets we experimented on, as well as the metrics used to quantify accuracy.

Description of Data Sets: Correlation affects the sampling component of our algorithms and thus is vital for the performance of our schemes. In addition to real data sets, we used synthetic data sets, in which we had the freedom to vary the degree of the underlying correlations and gain additional intuition about the performance of our proposal. We describe the data sets below:

- **Gaussian:** The values of each data stream are chosen independently from a Gaussian distribution $N(50,50)$ (mean 50 and variance 50). We expect no correlations between the streams.
- **Linear:** The values between the streams are linearly correlated.
- **Linear-S:** Starting from data set Linear we distort each data stream value by adding noise. In particular we add a sample from $N(2,2)$.
- **Linear-M:** Similar to data set Linear-S but we add samples from $N(10,10)$.
- **Real:** Real data representing the number of packets through various interfaces of several network cards of an operational router.

5.1 Performance Issues

In the set of experiments we report, we evaluate the performance of algorithm STREAMSVD compared with that of NaiveSVD. We report on the average time spent per stream tuple during the execution of the algorithms. This time consists of the time to update matrix M (AA^T in the case of NaiveSVD and AS^T in case of STREAMSVD) as well as the time to perform SVD on M , if required, amortized over a large number of stream tuples (100K). In these experiments algorithm *NaiveSVD* employs sampling of stream tuples, as proposed in section 3, boosting its performance. The performance gain is arising out of the fact that we require $O(m)$ time as opposed to $O(m^2)$ required by NaiveSVD to update the necessary matrices and not from sampling.

As far as performance is concerned two parameters are of interest; the number of streams involved m , as well as the value of s that affects the quality of the approximation.

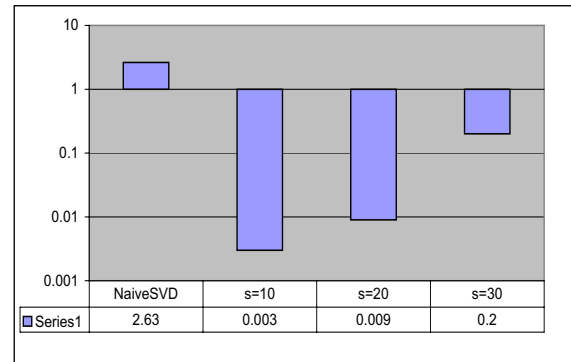
Varying s : The results are presented in Figure 5, to summarize:

- Figure 5(a) presents the time per stream tuple for data set Gaussian, as s increases, for $m = 100$ streams. Since there is no correlation between the streams, both algorithms compute the SVD for each new tuple arriving in the stream.
- Figure 5(b), presents the result of the same experiment for data sets Linear-M and Real. In this case, sampling is applied by both algorithms. The savings in response time per stream tuple achieved by STREAMSVD, are profound.

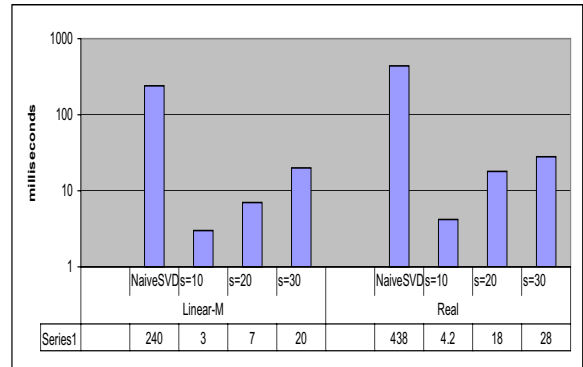
The graphs for varying m are omitted due to lack of space. We note however that the performance benefits of STREAMSVD increase with increasing m .

5.2 Additional Experiments

We have conducted experiments evaluating the accuracy of the approximation on eigenvalues and eigenvectors. The overall conclusion from these experiments is that the proposed techniques can attain very good accuracy for this task. We refer to the full version of this paper for further details.



(a) Data set Gaussian



(b) Data sets Linear-M, Real

Figure 5: Average time spend per stream tuple as the value of s increases, for various data sets, $m = 100$

6. CONCLUSIONS

We considered the problem of identifying correlations between multiple data streams using Singular Value Decomposition. We have proposed an algorithm to maintain the SVD of multiple data streams and identify correlations between the streams. Due to space limitations, this paper aims to convey the essence of our technique. We refer to the full version of the paper (available by contacting the authors) for a complete discussion of the techniques and the stream models our technique can adapt, complete set of results, experiments (evaluation of accuracy of our technique in various settings), and related work.

7. REFERENCES

- [1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Proceedings of the Symposium on Theory of Computing*, pages 20–29, 1996.
- [2] J. Demmel. *Applied Numerical Linear Algebra*. Society of Industrial and Applied mathematics, 1997.
- [3] P. Drineas, R. Kannan, A. Frieze, and V. Vinay. Clustering in large graphs and matrices. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1999.
- [4] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 1998.
- [5] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mapping into Hilbert Space. *Contemporary Mathematics, Vol 26*, pages 189–206, May 1984.