How to Fake Multiply by a Gaussian Matrix

Michael Kapralov MICHAEL.KAPRALOV@EPFL.CH
EPFL, Lausanne, Switzerland

Vamsi K. Potluru VAMSI_POTLURU@CABLE.COMCAST.COM
Comcast Cable, Washington DC, USA 20005

David P. Woodruff DPWOODRU@US.IBM.COM
IBM Research, Almaden, San Jose, CA USA

Abstract

Have you ever wanted to multiply an $n \times d$ matrix X, with $n \gg d$, on the left by an $m \times n$ matrix $\tilde{G}$ of i.i.d. Gaussian random variables, but could not afford to do it because it was too slow? In this work we propose a new randomized $m \times n$ matrix T, for which one can compute $T \cdot X$ in only $O(\text{nnz}(X)) + \tilde{O}(m^{1.5} \cdot d^3)$ time, for which the total variation distance between the distributions $T \cdot X$ and $\tilde{G} \cdot X$ is as small as desired, i.e., less than any positive constant. Here $\text{nnz}(X)$ denotes the number of non-zero entries of X. Assuming $\text{nnz}(X) \gg m^{1.5} \cdot d^3$, this is a significant savings over the naïve $O(\text{nnz}(X)m)$ time to compute $\tilde{G} \cdot X$. Moreover, since the total variation distance is small, we can provably use $T \cdot X$ in place of $\tilde{G} \cdot X$ in any application and have the same guarantees as if we were using $\tilde{G} \cdot X$, up to a small positive constant in error probability. We apply this transform to nonnegative matrix factorization (NMF) and support vector machines (SVM).

1. Introduction

One approach to handle high dimensional data, often in the form of a matrix, is to first project the data to a much lower dimensional subspace. This is an example of sketching and the last decade has seen a systematic study of this approach. A linear sketch of a matrix replaces the original matrix by a smaller matrix which is often obtained by a random projection of the original matrix (see, e.g., (Woodruff, 2014) for a survey). Random projections have been successfully applied to speed up least squares regression and have been implemented with remarkable success (Avron et al., 2010). This is impressive considering the fact that these solvers have been highly optimized over the last few decades, exploiting both algorithmic improvements and machine dependent optimizations.

Many of these works rely on *fast projection matrices*, such as the Subsampled Randomized Hadamard Transform or the CountSketch, the latter being particularly well-suited for sparse data (see, e.g., (Woodruff, 2014) and references therein). However, there are certain applications for which multiplying by a Gaussian matrix is the only way that is known to reduce the dimensionality of the data. This arises mainly because the application requires rotational symmetry, which is often not preserved by other fast transforms, or because additional properties, such as spreading out a sparse vector to a vector with non-spiky elements, do not hold for transforms like CountSketch (some of these hold for the Fast Hadamard Transform, but the latter are not known to be able to exploit sparsity). We give two such applications below, one to nonnegative matrix factorization (NMF), and one to support vector machines (SVM).

1.1. Our Results

A New Randomized Transform. In this work we propose a new randomized transform T, which we call the CountGauss. It is simply a product of a CountSketch matrix and a Gaussian matrix. That is, given an $n \times d$ matrix X which we would like to multiply by an $m \times n$ matrix $\tilde{G}$ of Gaussians, we instead let $T = G \cdot S$, where S is a $B \times d$ CountSketch matrix where $B = \tilde{O}(d^2 m^{\cdot 5})$, and $\tilde{G}$ is an $m \times B$ matrix of i.i.d. Gaussians. Recall that a CountSketch matrix S satisfies that each column of S has only a single non-zero entry chosen in a uniformly random position. That non-zero is 1 with probability $1/2$, and -1 with probability $1/2$. The columns of S are independent of each other. Importantly, computing $S \cdot X$ can be done in $O(\text{nnz}(X))$ time, and this

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

significantly reduces the number of rows of X. Then computing $G \cdot (S \cdot X)$ can now be done in $\tilde{O}(m^{1.5}d^3)$ time. While such a composition of matrices has been used before in the context of subspace embeddings for regression, see, e.g., (Clarkson and Woodruff, 2013b), here we show a new property of this composition - the distribution of $G \cdot S \cdot X$ looks like the distribution of $\tilde{G} \cdot X$! Formally, the statistical distance between the two distributions is smaller than any positive constant.

Therefore, in any application which uses $\tilde{G} \cdot X$, if we replace $\tilde{G} \cdot X$ with $G \cdot S \cdot X$, then if p is the success probability of the old algorithm, then the success probability of the new algorithm is at least $p - \delta$, where $\delta > 0$ is an arbitrarily small constant.

We now give applications.

Non-negative Matrix Factorization. Learning low rank structures and representations is a fundamental problem in machine learning. With the rise of data-driven decision making, many businesses, government agencies, and scientific laboratories are collecting increasingly large amounts of data each day. For instance, the large Hadron Collider (LHC) experiments represent about 150 million sensors acquiring around 40 million samples per second. Even working with 0.001 percent of the sensor data, the data flow from all four LHC experiments is around 25 petabytes per day (Brumfiel, 2011). This means the traditional approach of storing the data, and then processing it later, may be infeasible. One approach would be to subsample the incoming streams. However, we may lose valuable information in the form of infrequent events.

We use our transform to solve the nonnegative matrix factorization (NMF) problem. Previous approaches (Damle and Sun, 2014; Benson et al., 2014; Tepper and Sapiro, 2015) have used random matrices for the projection. However, these approaches can be slow if the dimensionality of the data is high since they rely on multiplying by Gaussian matrices, e.g., for natural images or structural Magnetic Resonance Imaging brain scans. Recent work by Smola et al. (Le et al., 2013) have shown that sometimes dense random Gaussian matrices can be replaced by faster transforms, and moreover, each row of the transform is equally likely to be in any direction on the unit sphere. To show the correctness of the NMF algorithm, however, we need a much stronger property than this, namely that any small subset of rows of the transform has the property that its product with a fixed matrix X has low variation distance to the distribution of a product of a Gaussian matrix with X. These latter properties, of having a fast transform with *equal representation of directions on the sphere*, do not seem to have been exploited in the context of NMF. Our transformation, since it has low variation distance to multiplying by a Gaussian matrix, directly applies here and we can use existing analysis.

We note that the classical way of speeding up Gaussian transforms via the Fast Hadamard or Fast Fourier Transform (see, e.g., (Tropp, 2011)) do not work in this context, since they miss large sections of the sphere, and we provide a formal counterexample in the full version of the paper. Intuitively, while it is fine to miss directions along large sections of the sphere to approximate the norm of a vector, it is not fine to miss directions for NMF, where the corresponding polytope partitions the sphere into a small number of caps, and each cap should have a random direction chosen from it.

Support Vector Machines. We also apply random projections to the support vector machines (SVM) problem. Previously, the CountSketch (CW) (Clarkson and Woodruff, 2013a) projection and random Gaussian (RG) projection have been applied to the linear SVM problem. Despite Countsketch being much faster than the Gaussian projection, the overall running time of projection together with the SVM solver was similar for both projections (Paul et al., 2014), since the training of the projected data was faster when using Gaussian projections. Our projection combines the CW matrix with a smaller Gaussian matrix thereby getting the best of both worlds — similar projection time as CountSketch and similar Gaussian properties of RG that are useful for SVM.

Experiments. We empirically validate our results for both NMF and SVM applications. For NMF, we give an experimental evaluation by comparing with state-of-the-art algorithms such as SPA (Gillis et al., 2014), XRAY (Kumar et al., 2013), naïve random projections (Damle and Sun, 2014) , structured Gaussian random projections (Tepper and Sapiro, 2015), and Tall-Skinny QR factorization (Benson et al., 2014) for NMF problems with applications to breast cancer, flow cytometry, climate data and movie analysis. Also, we show experimental speedups using our projection when combined with linear SVM solvers for document classification problems (Paul et al., 2014).

2. A New Randomized Transform

A CountSketch matrix $S \in \mathbb{R}^{B \times n}$ is a matrix all of whose rows have exactly one nonzero in a uniformly random location, and the value of the nonzero element is independently chosen to be -1 or $+1$ with equal probability. We denote the number of rows in the CountSketch matrix by B.

We prove the next theorem in the full version of our paper, which gives the formal guarantees of our new transform.

Theorem 1. *There exists an absolute constant $C > 0$ such that for every $\delta \in (0, 1)$, every integer $m \geq 1$*

and every matrix $U \in \mathbb{R}^{n\times d}$ with orthonormal columns if $B \geq \frac{1}{\delta}C(\log n)^4 \cdot d^2 \cdot m^{1/2}$, $S \in \mathbb{R}^{B\times n}$ is a random CountSketch matrix, and $G \in \mathbb{R}^{m\times B}$ and $\tilde{G} \in \mathbb{R}^{m\times n}$ are matrices of i.i.d. unit variance Gaussians, then the total variation distance between the joint distribution GSU and $\tilde{G}U$ is less than δ.

We note that Theorem 1 applies to matrices U with orthonormal columns. This is sufficient for applying our transform to an arbitrary matrix X, since we can write $X = UR$, where the columns of U form an orthonormal basis for the range of X, and apply the theorem to U. Since $G \cdot S \cdot U$ is close the $\tilde{G} \cdot U$ in total variation distance, $G \cdot S \cdot UR = G \cdot S \cdot X$ is close to $\tilde{G} \cdot UR = \tilde{G} \cdot X$ in total variation distance as well. We note that the role of d and n in Theorem 1 is swapped in comparison to our notation for application to NMF below. The notation in Theorem 1 is more consistent with the numerical linear algebra literature, and we thus prefer to state the theorem in this form.

In the full version of our paper we also first present a weaker version of Theorem 1, which establishes exactly the same guarantees but instead with $B \geq \frac{C}{\delta^2} \cdot d^2 \cdot m$. This version has a much simpler proof.

We first present the intuition behind the simpler version of Theorem 1. Consider the distribution of the first row of the two matrices, namely $\tilde{G}U$ versus GSU. Both random variables are Gaussians in dimension d, but while the former is an ideal isotropic Gaussian, the latter, despite being Gaussian, has correlated entries. The correlations between the entries are due to the fact that the CountSketch matrix S is not a perfect isometry: the correlation is given exactly by $U^T S^T SU$, which is the identity in expectation, but not for most realizations of S. In order to show that these two distributions are close in total variation distance, it would suffice to argue that the covariance matrix $U^T S^T SU$ is sufficiently close to the identity. This is exactly how the proof of the simpler version of Theorem 1 proceeds, which fixes an S for which $U^T S^T SU$ is sufficiently close to the identity, using a so-called "approximate matrix product" theorem in the linear algebra community. After fixing such an S, one can use that the rows of $G \cdot S$ and the rows of $\tilde{G}$ are independent, and then bound the variation distance between individual rows of $G \cdot S$ and of $\tilde{G}$. For the latter, it is convenient to work with Kullback-Leibler divergence (KL divergence) which is additive over product spaces; here we bound the KL divergence betwee a standard multivariate Gaussian and one with covariance matrix $U^T S^T SU$.

While this works for the simpler version of Theorem 1, it can be seen that since we need to ensure that the **joint** distribution of $\tilde{G}U$ is close to the joint distribution of GSU for most choices of S, we would need to set B to be at least $\approx d^2 m$ as opposed to $d^2\sqrt{m}$, as in our bound in Theorem 1. The idea behind the stronger result is to crucially use the fact that the distribution of GSU is a mixture of Gaussians with varying covariance matrices. When $B \approx d^2\sqrt{m}$, one can see that most Gaussians GSU in the mixture are too far in distribution from $\tilde{G}U$. However, we show that these differences that exist for **most** choices of the CountSketch matrix S cancel out **on average**. While mixtures of Gaussians with varying means and fixed covariance structure have been analyzed in the literature, to the best of our knowledge our analysis is the first to handle nontrivial mixtures with changing covariance.

3. Preliminaries for the Applications

A few applications of our new randomized transform are NMF and SVM, which we now formally define.

3.1. Nonnegative Matrix Factorization

Given a nonnegative matrix X of size $d \times n$, we would like to approximate it as a product of nonnegative matrices as follows: $X \approx WH$, where W is of size $d \times k$ and H is $k \times n$. This problem was studied by Paatero and Tapper (Paatero and Tapper, 1994) under the name of positive matrix factorization and gained a wider popularity through the work of Lee and Seung (Lee and Seung, 2001). NMF arises in a wide range of problems and application domains such as curve resolution in chemometrics and document clustering; further references can be found in (Arora et al., 2012). Various extensions to the original model to incorporate domain knowledge such as sparsity, orthogonality (Ding et al., 2006), and under-approximation (Gillis and Glineur, 2010) have also been studied. Commonly used measures of approximation include the Frobenius norm, Itakuro-Saito (IS), and Bregman divergence with applications in image processing, speech and music analysis (Yılmaz et al., 2011) among other places. Typical algorithms use alternating minimization to solve the nonconvex objective function arising from NMF.

Until recently, the complexity of the NMF problem was unknown. Vavasis established that the NMF problem is NP-hard (Vavasis, 2009). However, if the data satisfies the separability condition, a condition introduced by Donoho and Stodden (Donoho and Stodden, 2003), then tractable algorithms exist and have been recently proposed by Arora et al. (Arora et al., 2012; Recht et al., 2012). Formally, a nonnegative matrix X is k-separable if it satisfies the following condition: $X = X_I H$, where I is an index set of size k corresponding to the columns of the data matrix X. Geometrically, this assumption implies that the columns of X lie in a cone generated by the k selected columns of X indexed by I. One can view these k selected columns as the extreme points of a polytope containing all other columns. In practice, k is much smaller than both d and n. We will assume k-separability.

Given X_I, one can solve for X by solving a nonnegative least squares problem (Damle and Sun, 2014), and therefore our focus is on finding X_I, or equivalently, the index set I of extreme points of the point cloud formed by the columns of X.

To understand the guarantees of our algorithm, we define a few geometric notions also used in (Damle and Sun, 2014), which we refer to for more background. The *normal cone* of a convex set C at a point x is the cone

$$N_C(x) = \{w \in \mathbb{R}^d \mid w^T(y - x) \leq 0 \text{ for any } y \in C\},$$

that is, it is the cone defined by the outward normals of supporting hyperplanes at the point x. One can define a measure $\omega(K)$ on any cone K, which for full-dimensional cones K satisfies $\omega(K) = \Pr[\theta \in K \cap S^{d-1}]$ where θ is a uniformly random point on the sphere S^{d-1} in d dimensions. This measure is known as the *solid angle* of K. For any convex polytope C, if P is the set of its extreme points, then $\sum_{p \in P} \omega(N_C(p)) = 1$, that is, the solid angles of the normal cones at the extreme points sum to 1. If we label the points $p_i \in P$, we will use the shorthand $\omega_i = \omega(N_C(p_i))$.

A key property we will use is that for a unit vector u and a convex set C, the maximum inner product of u with any point $p \in C$ is achieved by an extreme point p of C. Moreover, the maximum is achieved by the extreme point p precisely when $u \in N_C(p)$. This follows since the inner product with a fixed vector u is a linear function, which is maximized by an extreme point for any convex set. These conditions also hold if we replace maximum with minimum.

Our results, as in (Damle and Sun, 2014), depend on the condition number $\kappa = \frac{1}{k \log\left(\frac{1}{\max_i 1-2\omega_i}\right)}$. The larger κ is, the more pointed the polytope defined by the columns of X is, whereas if κ is small, the polytope has "fatter" vertices.

3.2. Support Vector Machines

Given a dataset of samples and labels $\{x_i, y_i\}_{i=1}^N$ where x_i corresponds to sample i and y_i the corresponding label belonging to one of two classes denoted by $\{-1, 1\}$, we would like to find a maximum-margin hyperplane that separates the two classes. The primal form for the linear SVM problem is as follows:

$$\min_w \frac{1}{2}\|w\|_2^2 + \frac{C}{N}\sum_{i=1}^{N} \max(0, 1 - y_i\langle w, x_i\rangle) \qquad (1)$$

where C is soft-margin parameter which allows for misclassfication errors in the dataset and w is the maximum-margin hyperplane that we are learning from the data. The dual form for the linear SVM problem is given as follows:

$$\min_{0 \leq \alpha \leq C} \frac{1}{2}\alpha^\top X^\top X\alpha - 1^\top\alpha \qquad (2)$$

Previously (Paul et al., 2014) have shown that the margin (hyperplane) and minimum enclosing ball of the original data is preserved after projection up to a multiplicative factor. However, in their original formulation it is possible to just replace all points with zero to achieve the same guarantee. We strengthen the theorems by requiring that the projected data upper bound the objective of the original data. The details are in the full version of our paper.

4. Application to NMF

We consider the separable NMF problem as defined in Section 3. We first review an algorithm proposed by Damle and Sun (Damle and Sun, 2014). Their algorithm involves the computation of $\tilde{G}X$ where $\tilde{G}$ is of dimensions $m \times d$ for a parameter m, and the element-wise entries are distributed as $N(0, 1)$. Notice that we need to first compute the $m \times d$ random matrix $\tilde{G}$ which is itself dense. We also need to compute the matrix product $\tilde{G}X$ with the input data. This is computationally expensive and is of the order $O(mnd)$ in practice. Fast matrix multiplication routines (Coppersmith and Winograd, 1990; Williams, 2012) can be used in theory, but the time will still be at least $k^{\omega-2}nd$, where $\omega \approx 2.376$ is the exponent of fast matrix multiplication. Instead, we propose to use our new transform to significantly speed up the computations for extracting the extreme points in the dataset. Note that both approaches are easily amenable to distributed-data settings by simply sharing the seed of the random number generator which allows identical matrix transformations on all the computational nodes. Our new algorithm is called **C**ount **G**auss NMF or **CountGauss** and is as follows: Instead of using Gaussian random matrices for the projection, we approximate them by the following projection matrix $T = G \cdot S$, where the matrices are defined in Algorithm 1.

Algorithm 1 **C**ount**G**auss NMF (**CG**)

Initialize the index sets I_{max}, I_{min} to empty.

1. Let $T = G \cdot S$ where G is an $m \times B$ matrix of i.i.d Gaussians, and S is a $B \times d$ CountSketch matrix. Here $B = Cn^2m^{\cdot 5}\log^4 n/\delta$.
2. Compute the product $Z = TX$.
3. Find the indices which give the maximum and minimum across each row of Z corresponding to I_{max}, I_{min}

Consider the convex polytope defined by the columns of X and their negations. As defined in Section 3, we assume k-separability, namely, that there are k columns of X, indexed by I, for which $X = X_I H$ for a nonnegative matrix

H. The columns of X_I are the extreme points of a convex polytope C. By definition of an extreme point of a convex polytope, the indices found in step 3 of Algorithm 1 belong to the index set I.

Damle and Sun show the following.

Theorem 2. *(Theorem 3.3 of (Damle and Sun, 2014)) Consider a modification to Algorithm 1 in which we replace T by an $m \times d$ matrix of i.i.d. $N(0,1)$ random variables, where $m = \kappa k \log(\frac{k}{\delta})$, where recall $\kappa = \frac{1}{k \max_i 1-2\omega_i}$ is the condition number. Then the probability that the output $I_{min} \cup I_{max}$ of Algorithm 1 contains the index set I of extreme points of X is at least $1 - \delta$.*

Using Theorem 1, we analyze the performance of Algorithm 1.

Theorem 3. *Let $\delta > 0$ be given. Suppose in Algorithm 1 we set the parameter $m = \kappa k \log(\frac{k}{\delta})$, where $\kappa = \frac{1}{k \max_i 1-2\omega_i}$ is the condition number, and choose $B \geq \frac{1}{\delta} C(\log d)^4 \cdot n^2 \cdot m^{1/2}$ for a sufficiently large constant $C > 1$ as per Theorem 1. Then the probability that the output $I_{min} \cup I_{max}$ of Algorithm 1 contains the index set I of extreme points of X is at least $1 - 2\delta$.*

Proof. Let I be the index set of extreme points of the polytope defined by the columns of X. By definition of an extreme point, in each iteration of step 4 of the algorithm, we add an index $i \in I$ to I_{max} and an index $j \in I$ to I_{min} (since we are taking the inner product with a linear function). Therefore, the behavior of Algorithm 1 is the same if we instead, in each invocation of step 2, compute the product $Z = TX_I$.

By our assumption on m, since X_I is a $d \times k$ matrix we may apply Theorem 1, with the role of n and d in that theorem swapped, to obtain that the variation distance of the distributions of TX_I and $\tilde{G}X_I$ is at most $1 - \delta$, where $\tilde{G}$ is a matrix of i.i.d. $N(0,1)$ random variables. Therefore, we can apply Theorem 2 to conclude by a union bound that the output of Algorithm 1 contains the set I with probability at least $1 - 2\delta$. □

We obtain the same guarantee as Theorem 2 with considerably faster computation time. Indeed, our matrix product TX can be computed in $O(\text{nnz}(X)) + \tilde{O}(m^{1.5}n^3)$ time using our transform T, as opposed to the $O(dnm)$ time needed in (Damle and Sun, 2014) to compute the product $\tilde{G}X$ for a matrix $\tilde{G}$ of i.i.d. Gaussians. This is significant when d is very large.

Distributed Environments: Our results naturally provide solutions to NMF in a distributed environment in which the columns of X are partitioned across multiple servers. Indeed, the servers can agree upon a short random seed of length $O(d)$ words to generate T. Each server can then compute its local sets I_{max}, I_{min}, and send them to a coordinator who can find the global maxima and minima.

5. Other Related work

Over the last couple of years, many approaches have been proposed to solve the separable-NMF problem.

XRAY Selects the anchors one at a time by expanding a cone until all columnds in the dataset are contained in it. At each step, XRAY finds the datapoint (column) which maximizes the inner product with the current residual matrix. It then computes the residual matrix corresponding to the new set of anchor points (Kumar et al., 2013).

SPA Successive projection algorithm (Araújo et al., 2001; Gillis et al., 2014) is a family of recursive algorithms where the projections are given by strongly convex functions.

TSQR Use tall and thin QR factorization when the number of rows/features is really large (Benson et al., 2014). This approach is especially attractive when the number of features is really large ($\gg 10^6$) and number of samples is small ($< 10^5$).

SC In (Tepper and Sapiro, 2015), an algorithm similar to the one proposed by Damle and Sun (Damle and Sun, 2014) is proposed. The difference is that instead of choosing a Gaussian or FastFood projection matrix, the projection is chosen to be a matrix which depends on X (data dependent projection), namely, one that is found via the subspace power iteration (see Figure 3 of (Tepper and Sapiro, 2015)). This approach is expensive in the case of distributed settings since the projection matrix depends on all the samples.

6. Experiments

We show experiments validating our projection operator countGauss (CG) for NMF problems on various synthetic and real-world datasets. Also, we apply CG on the SVM problem for the TechTC300 datasets. In all our experiments[1], we set $B = 5m$.

Synthetic datasets. Similar to Damle and Sun (Damle and Sun, 2014), we generate the data as follows: We set a grid of tuples (k, m) such that $m/k \approx \log k$. For each tuple, we generate 500 separable datasets, say X, such that they are of size 1000×500 and have nonnegative rank k. Set matrix U with i.i.d. samples from the uniform random distribution in $[0, 1]$ of size $d \times k$. Also, generate matrix V with the identity matrix for the top k indices and the rest with i.i.d samples from the uniform distribution. Normalize each row of the matrix V to unit norm and compute the matrix product $X = UV^\top$. From Figure 1, we see that the CountGauss algorithm also requires $O(k \log k)$ optimizations to find all k extreme points with high probability. We

[1] https://github.com/marinkaz/nimfa

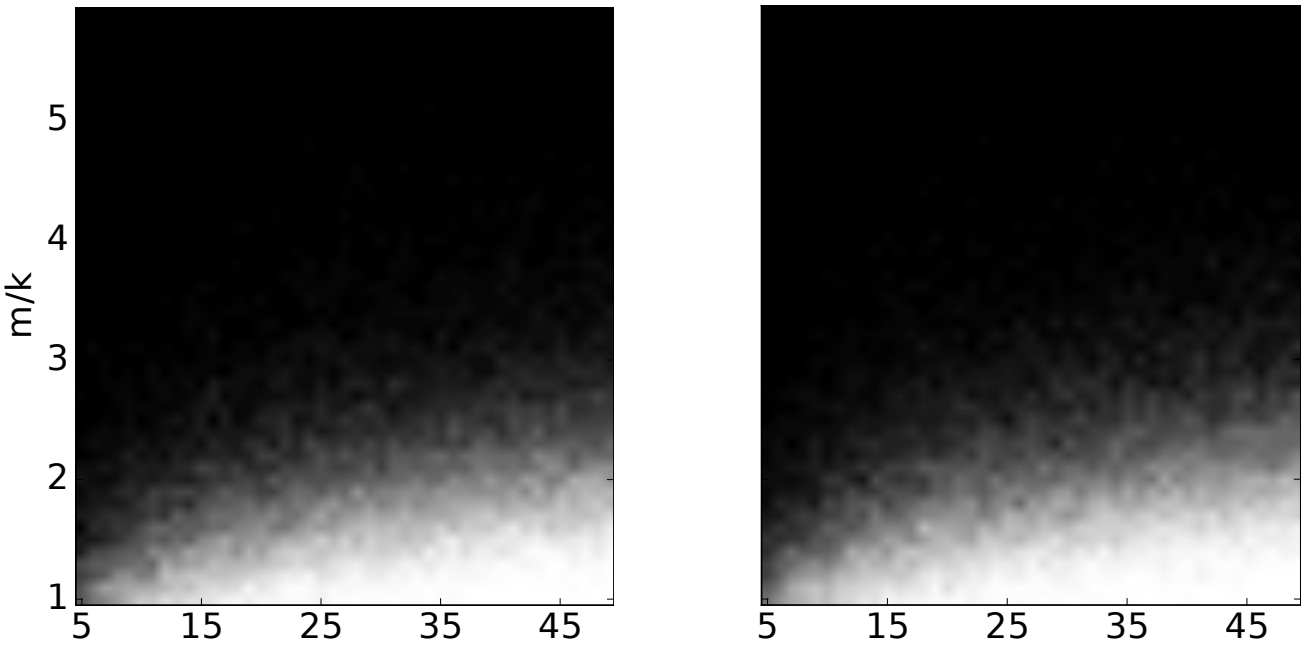


Figure 1: The fraction of trials in which the CG algorithm correctly extracted all "k' extreme points. For each value of k and m, we generated 500 matrices such that data matrix is of size 1000×500 and shown is how often we successfully recovered the original anchors (black indicates success). (Left) We contrast gaussian random projections (GP) with (right) our algorithm countGauss. Note that we recover the anchors with a similar success rate as GP.

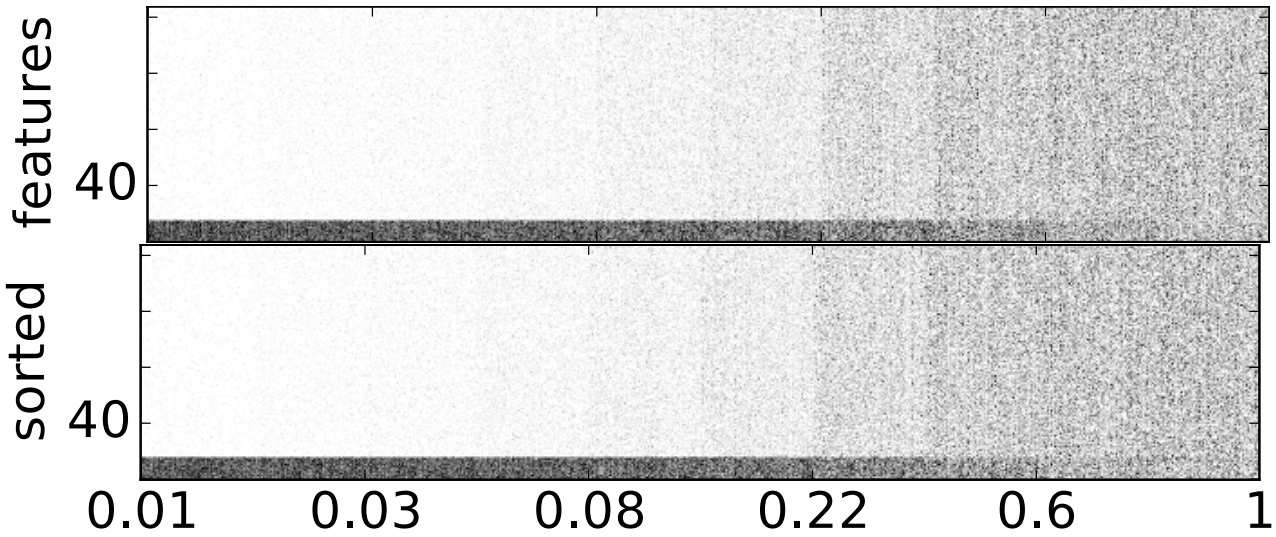


Figure 2: We show the scree plots at 20 noise levels and notice that there are sharp transitions at 20 corresponding to the rank of the data. (Top) Gaussian random projections and (bottom) our algorithm countGauss are applied to the dataset. For each noise value in $\{0.01, 0.02, 0.03, 0.05, 0.08, 0.12, 0.22, 0.36, 0.6, 1\}$, we generate 100 datasets. At higher noise levels, we note that both the algorithms GP and CG have most of the features active and there is no longer a sharp transition at 20.

also test the algorithm in the noisy case. For that, we generate U of size 1000×20 with uniform entries in $[0, 1]$ and and set the first 20 columns of data matrix X to U. The remaining 190 columns of X are set to the midpoints of the $k(k+1)/2$-dimensional faces of the polytope with extreme points chosen by the first 20 columns of X. Now, we add Gaussian noise to X with noise level σ which creates many spurious extreme points. The resulting scree plot is shown

Test		Proj		SVMf		Margin		Proj	Algo
mean	std	mean	std	mean	std	mean	std		
17.92	11.29	0.0000	0.0000	0.89	0.38	2.1057	3.9391	full	full
24.71	12.60	0.0086	0.0042	0.38	0.19	1.6792	3.5714	128	countSketch
25.27	13.08	0.0216	0.0047	**0.16**	**0.11**	1.6277	3.5634	128	countGauss
25.07	13.20	0.3676	0.1569	0.49	0.20	1.7143	3.7143	128	RG
17.92	11.29	0.0000	0.0000	0.89	0.38	2.1057	3.9391	full	full
22.56	12.42	0.0082	0.0036	0.54	0.21	1.8778	3.6709	256	countSketch
24.34	12.23	0.0565	0.0091	**0.21**	**0.07**	1.8722	3.7389	256	countGauss
23.66	12.86	0.8178	0.3286	0.98	0.35	1.8895	3.6747	256	RG
17.92	11.29	0.0000	0.0000	0.89	0.38	2.1057	3.9391	full	full
21.31	11.92	0.0075	0.0032	0.72	0.28	1.9914	3.7989	512	countSketch
22.11	12.89	0.1865	0.0228	**0.45**	**0.11**	1.9893	3.8453	512	countGauss
22.42	12.37	1.6057	0.6437	1.88	0.67	2.0148	3.9014	512	RG

Table 1: We applied CountGauss (CG), CountSketch (CW) and Random Gaussian (RG) on the TechTC300 dataset consisting of 295 pairs of data matrices and show the resulting mean and standard deviation for the resulting parameters such as projection time, SVMf time (projection + SVM training time), margin (gamma) and testing error. The results are shown over 10-fold cross validation with 4 repetitions and 3 runs over the random projection matrices. Note that the mean running times for our algorithm CG (highlighted) is faster than both CW and RG inspite of slower projection time than CW.

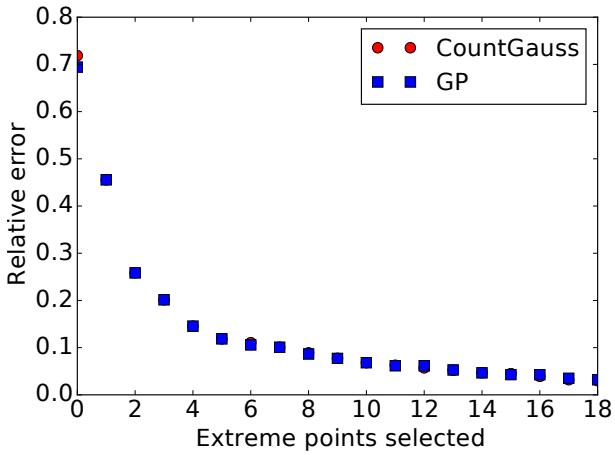


Figure 3: Relative reconstructive error as function of anchors selected by the two algorithms CountGauss and GP are shown. They are remarkably similar.

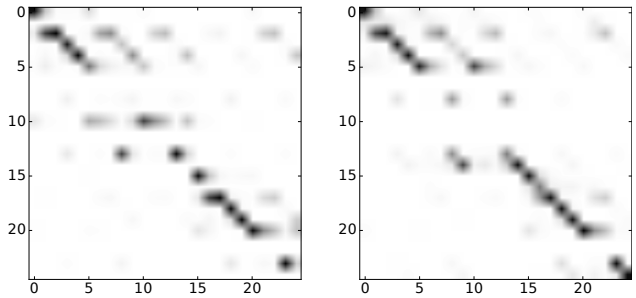


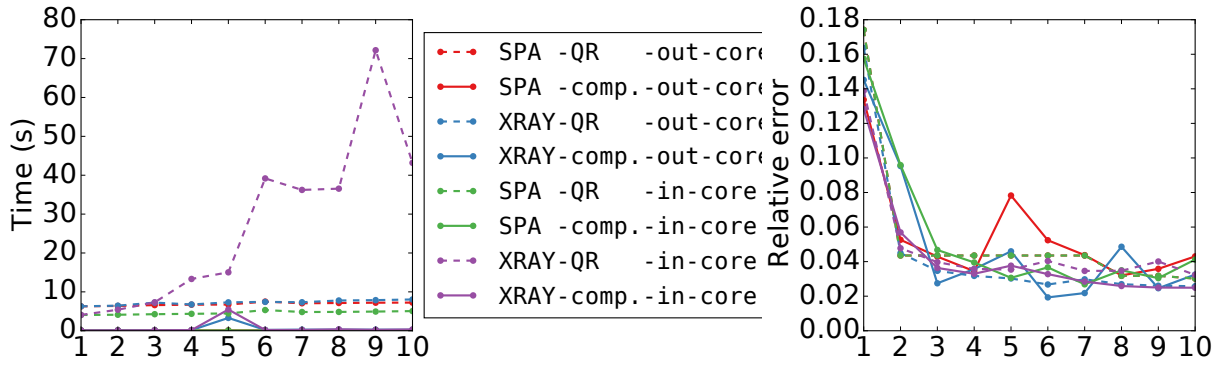
Figure 4: Coefficient matrices H are shown for the two algorithms GP and CountGauss for the flow cytometry data when k is set to 16. The coefficients tend to be clustered near the diagonal as has been previously observed.

in Figure 1.

Flow cytometry. The flow cytometry (FC) data represents abundances of fluorescent molecules labeling antibodies that bind to specific targets on the surface of blood cells. A more detailed description of the dataset can be found in (Benson et al., 2014). The measurements are represented as the data matrix A of size 40000×5. Since they study pairwise interactions in the data, the Kronecker product, $X = A \otimes A$ is formed which is of size $40000^2 \times 5^2$.

For this dataset, we exploit the data structure as follows. For some arbitrary input vector g, we know that $A \otimes Ag = A^\top GA$ where $g = \text{vec}(G)$. For each random projection, we can compute the matrix-matrix product AG very efficiently and in fact do not even need to generate the matrix G. For our algorithm, we do not need to explicitly compute the kronecker product and the complete NMF problem, including anchor selection and learning the weight coefficients, can be solved in a couple of seconds on an off-the-shelf desktop. As we can see from Figure 4 the results are pretty consistent from prior work (Benson et al., 2014). The weight matrix H still maintains a diagonal-like structure as previously observed.

Gene expression breast cancer dataset. We utilize the hereditary breast cancer dataset collected by Hedenfalk et al. (2001) which consists of the expression levels of 3226 genes on 22 samples from breast cancer patients. The patients consist of three groups: 7 patients with a BRCA1



(A) Running times using CountGauss (comp==CG) (B) Relative error using CG

Figure 5: Extracting columns with CG versus using QR factorization. Note that the QR-based methods are optimized for tall-and-skinny matrices and tend to do poorly for fat matrices. Note that CG (and SC) tends to perform well since it is based on random projections and is at least an order of magnitude faster than QR-based methods.

mutation, 8 samples with a BRCA2 mutation4, and 7 additional patients with sporadic cancers. It was analyzed using separable NMF in (Damle and Sun, 2014) and we similarly preprocess the dataset by exponentiating to make the log-expression levels nonnegative and normalize the columns. The size of the data matrix is 3222×22. The result of applying our algorithm CG and GP are shown in Figure 3. Notice that we get similar reconstruction error as GP while we vary the number of anchors.

Climate Dataset. We obtained a climate dataset which was analyzed in (Tepper and Sapiro, 2015). The data size is 10512×23742. First we present the running times and reconstruction error when using SC versus QR-based algorithms and then show the corresponding results when using CG algorithm in Figure 5. Note that CG (and SC) which is based on random projections is an order-of-magnitude faster compared to QR factorization methods.

SVM TechTC-300 Dataset. We obtained the TechTC-300 dataset which is a comprehensive directory of the web. There are 295-pairs of categories which provides a rich framework for running SVM experiments (Paul et al., 2014). Each data matrix has $10,000 - 40,000$ words and $150 - 280$ documents. LIBSVM was used with a linear kernel and soft-margin parameter C set to 500 for all experiments and we set the projections to $128, 256$, and 512. The results are summarized in the Table 1.

7. Discussion

We have presented an efficient way to multiply by a Gaussian matrix, without actually computing the dense matrix product. Theorem 1 provides our theoretical guarantees on this much faster transform, showing it has low variation distance to multiplication by a dense Gaussian matrix. We refer the reader to the full version of our paper for the formal proof of Theorem 1.

Our transform is useful in a surprising number of applications — here we apply our transform to NMF and SVM. The classical way of speeding up Gaussian transforms via the Fast Hadamard or FFT does not work in our setting since it misses large sections of the sphere.

Our experiments on synthetic and real-world datasets for NMF showed that the results obtained by our algorithm were on par with the state-of-the-art NMF algorithms such as SC, XRAY and SPA. In particular, for synthetic problems, we showed similar anchor recovery performance as random projection (GP) of Damle and Sun (Damle and Sun, 2014) both in the noiseless and noisy cases. Also, the performance was remarkably similar to GP when applied on the breast cancer dataset and also picked up activation patterns which might be of biological interest as previously noted in flow cytometry problems. Experiments on document classification tasks using the popular SVM formulation revealed that the new projection leads to faster SVM solutions than previous methods. Previously, it was shown that while CountSketch led to faster projection times it did not lead to overall faster training time and was in fact was found to be slower than random Gaussian projections (RG). Our new countGauss projection fixes this by sacrificing projection time compared to countSketch projection but leads to an overall faster SVM training time and thereby beats both random Gaussian and CountSketch-based SVM algorithms (Paul et al., 2014). We note that in practice for SVM, solution accuracy may be of critical importance rather than computation time and in these scenarios random projection based algorithms can be used to explore the optimal settings of the SVM parameters such as soft-margin. In our experiments (not shown) we noticed that these lead to faster training times while not sacrificing test accuracy.

Acknowledgements

Vamsi P. would like to acknowledge support from the following grant: NSF-IIP-1346452 and also RDI^2 at Rutgers University where this work was initiated. David W. would like to acknowledge the support from XDATA program of the Defense Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory contract FA8750-12-C-0323.

References

M. C. U. Araújo, T. C. B. Saldanha, R. K. H. Galvão, T. Yoneyama, H. C. Chame, and V. Visani. The successive projections algorithm for variable selection in spectroscopic multicomponent analysis. *Chemometrics and Intelligent Laboratory Systems*, 57(2):65–73, 2001.

S. Arora, R. Ge, R. Kannan, and A. Moitra. Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 145–162. ACM, 2012.

H. Avron, P. Maymounkov, and S. Toledo. Blendenpik: Supercharging lapack's least-squares solver. *SIAM J. Scientific Computing*, 32(3):1217–1236, 2010.

A. R. Benson, J. D. Lee, B. Rajwa, and D. F. Gleich. Scalable methods for nonnegative matrix factorizations of near-separable tall-and-skinny matrices. In *Advances in Neural Information Processing Systems*, pages 945–953, 2014.

G. Brumfiel. Down the petabyte highway. *Nature*, 469(20): 282–283, 2011.

K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 81–90, New York, NY, USA, 2013a. ACM. ISBN 978-1-4503-2029-0. doi: 10.1145/2488608.2488620. URL http://doi.acm.org/10.1145/2488608.2488620.

K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 81–90, 2013b.

D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

A. Damle and Y. Sun. Random projections for non-negative matrix factorization. *arXiv preprint arXiv:1405.4275*, 2014.

C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135. ACM, 2006.

D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in neural information processing systems*, page None, 2003.

N. Gillis and F. Glineur. Using underapproximations for sparse nonnegative matrix factorization. *Pattern recognition*, 43(4):1676–1687, 2010.

N. Gillis, S. Vavasis, et al. Fast and robust recursive algorithmsfor separable nonnegative matrix factorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(4):698–714, 2014.

A. Kumar, V. Sindhwani, and P. Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *Proceedings of The 30th International Conference on Machine Learning*, pages 231–239, 2013.

Q. Le, T. Sarlós, and A. Smola. Fastfood–approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

S. Paul, C. Boutsidis, M. Magdon-Ismail, and P. Drineas. Random projections for linear support vector machines. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(4):22, 2014.

B. Recht, C. Re, J. Tropp, and V. Bittorf. Factoring nonnegative matrices with linear programs. In *Advances in Neural Information Processing Systems*, pages 1214–1222, 2012.

M. Tepper and G. Sapiro. Compressed nonnegative matrix factorization is fast and accurate. *arXiv preprint arXiv:1505.04650*, 2015.

J. A. Tropp. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(1-2):115–126, 2011.

S. A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3): 1364–1377, 2009.

V. V. Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 887–898, 2012.

D. P. Woodruff. Sketching as a tool for numerical linear algebra. *arXiv preprint arXiv:1411.4357*, 2014.

K. Y. Yılmaz, A. T. Cemgil, and U. Simsekli. Generalised coupled tensor factorisation. In *Advances in Neural Information Processing Systems*, pages 2151–2159, 2011.