Random Matrices in Data Analysis

Dimitris Achlioptas

Microsoft Research, Redmond, WA 98052, U.S.A.
optas@microsoft.com

Abstract. We show how carefully crafted random matrices can achieve distance-preserving dimensionality reduction, accelerate spectral computations, and reduce the sample complexity of certain kernel methods.

1 Introduction

Given a collection of n data points (vectors) in high-dimensional Euclidean space it is natural to ask whether they can be projected into a lower dimensional Euclidean space without suffering great distortion. Two particularly interesting classes of projections are: i) projections that tend to preserve the interpoint distances, and ii) projections that maximize the average projected vector length.

In the last few years, distance-preserving projections have had great impact in theoretical computer science where they have been useful in a variety of algorithmic settings, such as approximate nearest neighbor search, clustering, learning mixtures of distributions, and computing statistics of streamed data.

The general idea is that by providing a low dimensional representation of the data, distance-preserving embeddings dramatically speed up algorithms whose run-time depends exponentially in the dimension of the working space. At the same time, the provided guarantee regarding pairwise distances often allows one to show that the solution found by working in the low dimensional space is a good approximation to the solution in the original space.

Perhaps the most commonly used projections aim at maximizing the average projected vector length, thus retaining most of the variance in the data. This involves representing the data as a matrix A, diagonalizing $A = UDV$, and projecting A onto subspaces spanned by the vectors in U or V corresponding to the largest entries in D. Variants of this idea are known as Karhunen-Loève transform, Principal Component Analysis, Singular Value Decomposition and others.

In this paper we examine different applications of random matrices to both kinds of projections, all stemming from variations of the following basic fact: if R is an $n \times n$ random matrix whose entries are i.i.d. Normal random variables, $N(0,1)$, then the matrix $\frac{1}{\sqrt{n}} R$ is very close to being orthonormal.

2 Euclidean distance preservation

A classic result of Johnson and Lindenstrauss [7] asserts that any set of n points in $\mathbb{R}^d$ can be embedded into $\mathbb{R}^k$, with $k = O(\log n)$, so that all pairwise distances are maintained within an arbitrarily small factor. More precisely,

Lemma 1 ([7]). *Given $0 < \epsilon \leq 1$ and an integer n, let k be a positive integer such that $k \geq k_0 = (12/\epsilon^2) \log n$. For every set P of n points in $\mathbb{R}^d$ there exists $f : \mathbb{R}^d \to \mathbb{R}^k$ such that for all $u, v \in P$*

$$(1 - \epsilon)||u - v||^2 \leq ||f(u) - f(v)||^2 \leq (1 + \epsilon)||u - v||^2 \ .$$

Perhaps, a naive attempt to construct an embedding as above would be to pick a random set of k coordinates from the original space. Unfortunately, two points can be very far apart while differing only along one original dimension, dooming this approach. On the other hand, if (somehow) for all pairs of points, all coordinates contributed "roughly equally" to their distance, such a sampling scheme would be very natural. This consideration motives the following idea: first apply a random *rotation* to the n points, and then pick the first k coordinates as the new coordinates. The random rotation can be viewed as a form of insurance against axis alignment, analogous to applying a random permutation before running Quicksort.

Of course, applying a random rotation and then taking the first k coordinates is equivalent to projecting the n points on a uniformly random k-dimensional subspace. Indeed, this is exactly how the original proof of Lemma 1 by Johnson and Lindenstrauss proceeds: to implement the embedding, multiply the $n \times d$ data matrix A with a random $d \times k$ orthonormal matrix. Dasgupta and Gupta [5] and, independently, Indyk and Motwani [6] more recently gave a simpler proof of Lemma 1 by taking the following more relaxed approach towards orthonormality.

The key idea is to consider what happens if we multiply A with a random $d \times k$ matrix R whose entries are independent Normal random variables with mean 0 and variance 1, i.e., $N(0, 1)$. It turns out that while we do not explicitly enforce either orthogonality or normality in R, its columns will come very close to having both of these properties. This is because, as d increases: (i) the length of each column-vector concentrates around its expectation as the sum of d independent random variables; (ii) by the spherical symmetry of the Gaussian distribution, each column-vector points in a uniformly random direction in $\mathbb{R}^d$, making the $k \leq d$ independent column-vectors nearly orthogonal with high probability.

More generally, let R be a random matrix whose entries are independent random variables with $\mathbf{E}(r_{ij}) = 0$ and $\mathrm{Var}(r_{ij}) = 1$. If $f : \mathbb{R}^d \to \mathbb{R}^k$ is given by

$$f(x) = \frac{1}{\sqrt{k}}\, x\, R \ ,$$

it is easy to check that for any vector $x \in \mathbb{R}^d$ we have $\mathbf{E}\,(||f(x)||) = ||x||$. Effectively, the squared inner product of x with each column of R acts as an independent estimate of $\|x\|^2$, making $||f(x)||^2$ the consensus estimate (sum) of the k estimators. Seen from this angle, requiring the k vectors to be orthonormal simply maximizes the mutual information of the k estimators. For good dimensionality reduction, we also need to minimize the variance of the estimators.

In [1], it was shown that taking $r_{ij} = \pm 1$ with equal probability, in fact, slightly reduces the number of required dimensions k (as the variance of each column-estimator is slightly smaller). At the same time, and more importantly, this choice of r_{ij} makes f a lot easier to work with in practice.

3 Computing Low Rank Approximations

Given n points in $\mathbb{R}^d$ represented as an $n \times d$ matrix A, one of the most common tasks in data analysis is to find the "top k" singular vectors of A and then project A onto the subspace they span. Such low rank approximations are used widely in areas such as computer vision, information retrieval, and machine learning to extract correlations and remove noise from matrix-structured data.

Recall that the top singular vector of a matrix A is the maximizer of $\|Ax\|_2$ over all unit vectors x. This maximum is known as the L_2 norm of A and the maximizer captures the dominant linear trend in A. Remarkably, this maximizer can be discovered by starting with a random unit vector $x \in \mathbb{R}^d$ and repeating the following "voting process" until it reaches a fixpoint, i.e., until x stops rotating:

- Have each of the n rows in A vote on candidate x, i.e., compute $y = Ax \in \mathbb{R}^n$.
- Compose a new candidate by combining the rows of A, weighing each row by its enthusiasm for x, i.e., update $x \leftarrow \dfrac{A^T y}{\|A^T y\|} \in \mathbb{R}^d$.

The above idea extends to $k > 1$. To find the k-dimensional invariant subspace of A, one starts with a random subspace, i.e., a random $d \times k$ orthonormal matrix, and repeatedly multiplies by $A^T A$ (orthonormalizing after each multiplication). Computing the singular row-vectors of A, i.e., the eigenvectors of $B = A^T A$, is often referred to as Principal Component Analysis (PCA). The following process achieves the exact same goal, by extracting the dominant trends in A sequentially, in order of strength: let A_0 be the all zeros matrix; for $i = 1, \ldots, k$:

- Find the top singular vector, x_i, of $A - A_{i-1}$, via the voting process above.
- Let $A_i = A_{i-1} + Ax_i x_i^T$, i.e., A_i is the optimal rank i approximation to A.

To get an idea of how low rank approximations can remove noise, let G be an $n \times d$ random matrix whose entries are i.i.d. $N(0, \sigma^2)$ random variables. We saw earlier that each column of G points in an independent, uniformly random direction in $\mathbb{R}^n$. As a result, when n is large, with high probability the $d \leq n$ columns of G are nearly orthogonal and there is *no* low-dimensional subspace that simultaneously accommodates many of them. This means that when we compute a low rank approximation of $A + G$, as long as σ is "not too large" (in a sense we will make precise), the columns of G will exert little influence as they do not strongly favor any particular low-dimensional subspace. Assuming that A contains strong linear trends, it is its columns that will command and receive accommodation.

To make this intuition more precise, we first state a general bound on the impact that a matrix N can have on the optimal rank k approximation of a matrix A, denoted by A_k, as a function of $\|N_k\|$. Recall that $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$.

Lemma 2. *For any matrices A and N, if $\widehat{A} = A + N$ then*

$$\|A - \widehat{A}_k\|_2 \leq \|A - A_k\|_2 + 2\|N_k\|_2 \quad \textit{and}$$

$$\|A - \widehat{A}_k\|_F \leq \|A - A_k\|_F + \|N_k\|_F + 2\sqrt{\|N_k\|_F \|A_k\|_F} \ .$$

Notice that all error terms above scale with $\|N_k\|$. As a result, whenever N is poorly approximated in k dimensions, i.e., $\|N_k\|$ is small, the error caused by adding N to a matrix A is also small.

Let us consider the norms of our Gaussian perturbation matrix.

Fact 1 *Let G be a random $n \times d$ matrix, where $d \leq n$, whose entries are i.i.d. random variables $N(0, \sigma^2)$. For any $\epsilon > 0$, with probability $1 - 1/\mathrm{poly}(n, \epsilon)$,*

$$\|G\|_2 = \ \|G_k\|_2 \ < \ (2+\epsilon)\,\sigma\sqrt{n} \ \ \textit{and} \ \ \|G_k\|_F \ < \ (2+\epsilon)\,\sigma\sqrt{kn} \ .$$

Remarkably, the upper bound above for $\|G\|_2$ is within a factor of 2 of the *lower bound* $\sigma\sqrt{n}$ on the L_2 norm of *any* $n \times d$ matrix with mean squared entry σ^2. In other words, a random Gaussian matrix is nearly as unstructured as possible, resembling white noise in the flatness of its spectrum. On the other hand, $\|A\|_2$ can be as large as $\sigma\sqrt{dn}$ for an $n \times d$ matrix A with mean squared entry σ^2.

This capacity of spectral techniques to remove Gaussian noise is by now very well-understood. We will see that the above geometric explanation of this fact can actually accommodate much more general noise models, e.g. N_{ij} that are not identically distributed and, in fact, whose distribution depends on A_{ij}. In the next section, this generality will enable the notion of "computation-friendly noise", i.e., noise that enhances (rather than hinders) spectral computations.

Fact 1 also suggests a criterion for choosing a good value of k when seeking low rank approximations of a $n \times d$ data matrix A:

$$\|A - A_k\|_2 \sim \sigma\sqrt{n}, \text{ where } \sigma^2 \text{ is the mean squared entry in } A - A_k.$$

In words: we should stop when, after projecting onto the top k singular vectors, we are left with a matrix, $A - A_k$, whose strongest linear trend is comparable to that of a random matrix of similar scale.

3.1 Co-opting the Noise Process

Computing optimal low rank approximations of large matrices often runs against practical computational limits since the algorithms for this task generally require superlinear time and a large working set. On the other hand, in many applications it is perfectly acceptable just to find a rank k matrix C satisfying

$$\|A - C\| \leq \|A - A_k\| + \delta \ ,$$

where A_k is the optimal rank k approximation of the input matrix A, and δ captures an appropriate notion of "error tolerance" for the domain at hand.

In [2], it was shown that with the aid of randomization one can exploit such an "error allotment" to aid spectral computations. The main idea is as follows.

Imagine, first, that we squander the error allotment by obliviously adding to A a Gaussian matrix G, as in the previous section. While this is not likely to yield a computationally advantageous matrix, we saw that at least it is rather harmless. The first step in using noise to aid computation is realizing that G is innocuous due precisely to the following three properties of its entries:

independence, zero mean, small variance.

The fact that the G_{ij} are Gaussian is *not* essential: a fundamental result of Füredi and Komlós [4] shows that Fact 1 generalizes to random matrices where the entries can have different, in fact arbitrary, distributions as long as all N_{ij} are zero-mean, independent, and their variance is bounded by σ^2.

To exploit this fact for computational gain, given a matrix A, we will create a distribution of noise matrices N that *depends on* A, yet is such that the random variables N_{ij} still enjoy independence, zero mean, and small variance. In particular, we will be able to choose N so that $\widehat{A} = A + N$ has computationally useful properties, such as sparsity, yet N is sufficiently random for $\|N_k\|$ to be small with high probability.

Example: Set $N_{ij} = \pm A_{ij}$ with equal probability, independently for all i, j.

In this example, the random variables N_{ij} are independent, $\mathbf{E}[N_{ij}] = 0$ for all i, j, and the standard deviation of N_{ij} equals A_{ij}. On the other hand, the matrix $\widehat{A} = A + N$ will have about half as many non-zero entries as A, i.e., it will be about twice as sparse. Therefore, while $\|A\|_2$ can be proportional to $\sqrt{dn}$, the error term $\|N\|_2$, i.e., the price for the sparsification, is only proportional to $\sqrt{n}$.

The rather innocent example above can be greatly generalized. To simplify exposition, in the following, we assume that $A_{ij} \in [-1, +1]$.

- **Quantization:** For all i, j, independently, set $\widehat{A}_{ij}$ to $+1$ with probability $(1 + A_{ij})/2$, and to -1 with probability $(1 - A_{ij})/2$. Clearly, for all i, j, we have $\mathbf{E}[N_{ij}] = \mathbf{E}[\widehat{A}_{ij} - A_{ij}] = 0$, while $\mathrm{Var}(N_{ij}) \leq N_{ij}^2 \leq 4$.
- **Uniform sampling:** For any desired fraction $p \in (0, 1]$, set $\widehat{A}_{ij} = A_{ij}/p$ with probability p, and 0 otherwise. Now, $\mathrm{Var}(N_{ij}) = A_{ij}^2(1 - p)/p \leq 1/p$, so that the error grows only as $1/\sqrt{p}$ as we retain a p-fraction of all entries.
- **Weighted sampling:** For all i, j, independently, set $\widehat{A}_{ij} = A_{ij}/p_{ij}$ with probability p_{ij}, and 0 otherwise, where $p_{ij} = pA_{ij}^2$. This way we retain even fewer small entries, while maintaining $\mathrm{Var}(N_{ij}) = 1/p - A_{ij}^2 \leq 1/p$.

Reducing the number of non-zero entries and their representation length causes standard eigenvalue algorithms to work faster. Moreover, the reduced memory footprint of the matrix $\widehat{A}$ enables the handling of larger data sets. At a high level, we perform data reduction by randomly perturbing each data vector so as to simplify its representation, i.e., sparsify and quantize. The point is that the perturbation vectors we use, by virtue of their independence, do not fit in a small subspace, acting effectively as "white noise" that is largely filtered out.

4 Kernel Principal Component Analysis

Given a collection $\mathcal{X}$ of training data $x_1, \ldots, x_n \in \mathbb{R}^d$, techniques such as linear SVMs and PCA extract features from $\mathcal{X}$ by computing linear functions of $\mathcal{X}$. However, often the structure present in the training data is not a linear function of the data representation. Worse, many data sets do not readily support linear operations such as addition and scalar multiplication (text, for example).

In a "kernel method" the idea is to map $\mathcal{X}$ into a space $\mathcal{H}$ equipped with inner product. The dimension of $\mathcal{H}$ can be very large, even infinite, and therefore it may not be practical (or possible) to work with the mapped data explicitly by applying $\Phi : \mathcal{X} \to \mathcal{H}$. Nevertheless, in many interesting cases it is possible to efficiently evaluate the dot products $\langle \Phi(x_i), \Phi(x_j) \rangle$ via a positive definite kernel k for Φ, i.e., a function k so that $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$. Algorithms whose operations can be expressed in terms of inner products can thus operate on $\Phi(\mathcal{X})$ implicitly, given only the *Gram* matrix

$$K_{ij} := k(x_i, x_j) \ .$$

Given n training data points, the Kernel PCA (KPCA) method [8] begins by forming the Gram matrix K above and computing the ℓ largest eigenvalues, $\lambda_1, \ldots, \lambda_\ell$, and corresponding eigenvectors, $e_1, \ldots, e_\ell$ of K, for some appropriate choice of $\ell \leq n$. Then, given an input point x, the method computes the value of the ℓ nonlinear feature extractors, corresponding to the inner product of the vector $k(x) = (k(x, x_1), k(x, x_2), \ldots, k(x, x_n))$ with each of the eigenvectors. These feature-values can be used for clustering, classification etc.

While Kernel PCA is very powerful the matrix K, in general, is dense making the input size scale as n^2, where n is the number of training points. As kernel functions become increasingly more sophisticated, e.g. invoking dynamic programming to evaluate the similarity $k(x_i, x_j)$ of two strings x_i, x_j, just the cost of $\Theta(n^2)$ kernel evaluations to construct K rapidly becomes prohibitive.

The uniform sparsification and quantization techniques of the previous section are ideally suited for speeding up KPCA. In particular, "sparsification" here means that we actually only construct a matrix $\widehat{K}$ by computing $k(x_i, x_j)$ for a uniformly random subset of all input pairs x_i, x_j and filling in 0 for the remaining pairs. In [3], it was proven that as long as K has strong linear structure (which is what justifies KPCA in the first place), with high probability, the invariant subspaces of $\widehat{K}$ will be very close to those of K.

Also, akin to quantization, we can replace each exact evaluation of $k(x_i, x_j)$ with a more easily computable unbiased estimate for it. In [3], it was shown that for kernels where: i) $\mathcal{X} \subseteq \mathbb{R}^d$, and, ii) $k(x_i, x_j)$ depends only on $\|x_i - x_j\|$ and/or $x_i \cdot x_j$, one can use random projections, as described in Section 2 for this purpose. Note that this covers some of the most popular kernels, e.g., radial basis functions (RBF) and polynomial kernels.

5 Future Work

Geometric and spectral properties of random matrices with zero-mean, independent entries are the key ingredients in all three examples we considered [1–3]. More general ensembles of random matrices hold great promise for algorithm design and call for a random matrix theory motivated from a computational perspective. Two natural directions are the investigation of matrices with limited independence, and the development of concentration inequalities for non-linear functionals of random matrices.

We saw that sampling and quantizing matrices can be viewed as injecting "noise" into them to endow useful properties such as sparsity and succinctness. The distinguishing feature of this viewpoint is that the effect of randomization is established without an explicit analysis of the interaction between randomness and computation. Instead, matrix norms act as an interface between the two domains: (i) matrix perturbation theory asserts that matrices of small spectral norm cannot have a large effect in eigencomputations, while (ii) random matrix theory asserts that matrices of zero-mean, independent random variables with small variance have small spectral norm. Is it possible to extend this style of analysis to other machine-learning settings, e.g. Support Vector Machines?

Acknowledgments. Many thanks to Robert Kleinberg, Heikki Mannila, and Frank McSherry for reading earlier drafts and providing helpful suggestions.

References

1. Dimitris Achlioptas, *Database-friendly random projections: Johnson-Lindenstrauss with binary coins*, JCSS **66** (2003), no. 4, 671–687.
2. Dimitris Achlioptas and Frank McSherry, *Fast computation of low rank matrix approximations*, JACM, to appear.
3. Dimitris Achlioptas, Frank McSherry and Bernhard Schölkopf, *Sampling techniques for kernel methods*, NIPS 2002, pp. 335–342.
4. Zoltán Füredi and János Komlós, *The eigenvalues of random symmetric matrices*, Combinatorica **1** (1981), no. 3, 233–241.
5. Sanjoy Dasgupta and Anupam Gupta, *An elementary proof of the Johnson-Lindenstrauss lemma*, Technical report 99-006, UC Berkeley, March 1999.
6. Piotr Indyk and Rajeev Motwani, *Approximate nearest neighbors: towards removing the curse of dimensionality*, STOC 1998, pp. 604–613.
7. William B. Johnson and Joram Lindenstrauss, *Extensions of Lipschitz mappings into a Hilbert space*, Amer. Math. Soc., Providence, R.I., 1984, pp. 189–206.
8. Bernhard Schölkopf, Alex J. Smola and Klaus-Robert Müller, *Nonlinear component analysis as a kernel Eigenvalue problem*, Neural Computation **10** (1998), no. 5, 1299–1319.