3 Random Projection and Hashing

Random Projection is another class of methods used for low-rank matrix approximation. A random projection algorithm projects datapoints from a high-dimensional space $\mathbb{R}^n$ onto a lower-dimensional subspace $\mathbb{R}^r (r \ll n)$ using a random matrix $S \in \mathbb{R}^{r \times n}$. The key idea of random mapping comes from the *Johnson-Lindenstrauss* lemma[7] (we will explain later in detail) that says "if points in a vector space are projected onto a randomly selected subspace of suitably high dimension, then the distances between points are approximately preserved".

Random projection methods are computationally efficient and sufficiently accurate in practice for dimensionality reduction of high-dimensional datasets. Moreover, since complexity of many geometric algorithms depends significantly on the dimension, applying random projection as pre-processing is a common task in many data mining applications. As opposed to column sampling methods, that need to access data for approximating the low-rank subspace, random projections are data oblivious, as their computation involves only a random matrix S.

3.1 Subspace Embedding

We start with the basic concepts and definitions required to understand random projections techniques.

Definition 1 (Column Space)**.** *Consider a matrix $A \in \mathbb{R}^{n \times d}(n > d)$. Notice that as one ranges over all vectors $x \in \mathbb{R}^d$, Ax ranges over all linear combinations of the columns of A and therefore defines a d-dimensional subspace of $\mathbb{R}^n$, which we refer to as the column space of A and denote it by $C(A)$.*

Definition 2 (ℓ_2-Subspace Embedding)**.** *A matrix $S \in \mathbb{R}^{r \times n}$ provides a subspace embedding for $C(A)$ if $\|SAx\|_2^2 = (1 \pm \varepsilon)\|Ax\|_2^2, \ \forall x \in \mathbb{R}^d$. Such matrix S provides a low distortion embedding, and is called a $(1 \pm \varepsilon)$ ℓ_2-subspace embedding.*

Using an ℓ_2-subspace embedding, one can work with $SA \in \mathbb{R}^{r \times d}$ instead of $A \in \mathbb{R}^{n \times d}$. Typically $r \ll n$, so we are working with a smaller matrix that reduces the time/space complexity of many algorithms. However note that definitely r needs to be larger than d if we are talking about the whole subspace $\mathbb{R}^d$[11]. Note that subspace embedding does not depend on a particular basis for $C(A)$, that means if we have a matrix U that is an orthonormal basis for $C(A)$, then Ux gives the same subspace as Ax. Therefore if S is an embedding for A, it will be an embedding for U too. Let's consider $U \in \mathbb{R}^{n \times t}$, where $t = \mathsf{rank}(A)$. Then $\|SUx\|^2 = (1 \pm \varepsilon)\|Ux\|^2$ for all $x \in \mathbb{R}^t$, and a more general form of this is $\|Sx\|^2 = (1 \pm \varepsilon)\|x\|^2$ for all $x \in \mathbb{R}^n$[11].

There are many ways of constructing a ℓ_2-subspace embedding for a matrix, e.g. sampling rows of A proportional to their squared norms or their leverage scores. However these embeddings are clearly data dependent. As stated before, random projections provide oblivious subspace embeddings, which are data independent. Below we define such embedding formally.

Definition 3 (Oblivious Subspace Embedding)**.** *Suppose Π is a distribution on $r \times n$ matrices S, where r is a function of n, d, ε, and δ. Suppose that with probability at least $1 - \delta$, for any fixed $n \times d$ matrix A, a matrix S drawn from distribution Π has the property that S is a $(1 \pm \varepsilon)$ ℓ_2-subspace embedding for A. Then we call Π an (ε, δ) oblivious ℓ_2-subspace embedding[11].*

But why is this even possible? Why would a random matrix S exists that preserve the column space of any arbitrary matrix A?

3.2 Johnson-Lindenstrauss Transform

In 1980, two mathematicians Johnson and Lindenstrauss proved that d datapoints in any dimension (e.g $\mathbb{R}^n$ for $n \gg d$) can get embedded into roughly $\log d$ dimensional space, such that their pair-wise distances are preserved to some extent (upto a multiplicative factor $1 \pm \varepsilon$)[8]. More precisely, they defined a matrix $S \in \mathbb{R}^{r \times n}$ as an orthogonal projection on a random r-dimensional subspace of $\mathbb{R}^n$ with $r = O(\varepsilon^{-2} \log d)$, and showed that for any matrix $A \in \mathbb{R}^{n \times d}$, SA preserves pair-wise distances between d datapoints in A.

Although, there are many other ways to construct a matrix S that preserve pair-wise distances, this choice can still be considered the most intuitive geometrically. All of such matrices are called to have the *Johnson-Lindenstrauss Transform (JLT)* property. Below we define this property formally.

Definition 4 (Johnson-Lindenstrauss Transform). *A random matrix $S \in \mathbb{R}^{r \times n}$ forms a $JLT(\varepsilon, \delta, d)$ if for all vectors v, v' of any subset of $\mathbb{R}^n$ with size d, we get $\|S(v - v')\|^2 = (1 \pm \varepsilon)\|v - v'\|^2$ with probability atleast $1 - \delta$.*

JLT property can appear in dot product form too, i.e. $|\langle Sv, Sv' \rangle - \langle v, v' \rangle| \leq \varepsilon \|v\| \|v'\|$, both forms are equivalent because

$$\begin{aligned} \langle Sv, Sv' \rangle &= \left(\|S(v + v')\|_2^2 - \|Sv\|_2^2 - \|Sv'\|_2^2 \right) /2 \\ &= (1 \pm \varepsilon)\|v + v'\|_2^2 - (1 \pm \varepsilon)\|v\|_2^2 - (1 \pm \varepsilon)\|v'\|_2^2 \\ &= \langle v, v' \rangle \pm O(\varepsilon) \end{aligned}$$

which implies all inner products are preserved up to ε by rescaling ε by a constant[11].

There are many ways to construct a matrix S with JLT property. All different approaches either aim at finding a matrix S with a small number of rows, i.e. r, or in other words small number of non-zeros, or they aim at constructing a matrix S such that SA can be computed quickly. Below we review both group of approaches.

3.2.1 Different Constructions for JLT

As noted before, the first JLT matrix was by Johnson and Lindenstrauss[7], where they set $S = \sqrt{\frac{n}{r}} P \in \mathbb{R}^{r \times n}$, where P is an orthogonal projection matrix onto R^r, and $r = O(\varepsilon^{-2} \log d)$. Indyk and Motwani[6] showed that the condition of orthogonality can be dropped, and instead they constructed $S = \frac{1}{\sqrt{r}} P \in \mathbb{R}^{r \times n}$ where $r = O(1/\varepsilon^2 \log(d/\delta))$ and P has entries as independent random variables with the standard normal distribution $N(0, 1)$. Clearly, this matrix S is much easier to generate than the orthogonal projection one[8].

However, Achlioptas[1] noted that S can be generated in a computationally simpler way. Namely, he proved that the entries of S can be chosen as independent ± 1 random variables (each attaining values $+1$ and -1 with probability $1/2$). Another variant of his result has the entries of A attaining value 0 with probability $2/3$ and values $+1$ and -1 with probability $1/6$ each. This latter setting allows for computing the image Sx about 3 times faster than the former, since S is sparse, only about one third of the entries are nonzero[8]. The underlying intuition here is that for a vector $x \in \mathbb{R}^n$ whose norm is spread roughly uniformly across its n coordinates, sampling a small number of its coordinates uniformly at random and rescaling results in a good estimate of the ℓ_2 norm of x.

Dasgupta, Kumar,and Srlos[5] showed that it suffices for each column of S to have only $O(\varepsilon^{-1}\text{polylog}(d/\delta))$ non-zeros per column. Note that if $\text{polylog}(n/\delta)$ is much smaller than ε^{-1}, this is a significant improvement over $\Omega(\varepsilon^{-2} \log(d/\delta))$ number of non-zero entries per column achieved by previous schemes. The $O(\varepsilon^{-1}\text{polylog}(d/\delta))$ sparsity was later optimized by kane and nelson to $O(\varepsilon^{-1} \log(d/\delta))$ non-zero entries per column. The latter was shown to be almost tight by Nelson and Nguyen[9], who showed that $\Omega(\varepsilon^{-1} \log(d/\delta)/ \log(1/\varepsilon))$ column sparsity is required. In short, the above line of work shows that it is possible to apply a JLT(ε, δ, d) matrix S to a vector x in $O(nnz(x).\varepsilon^{-1}. \log(d/\delta))$ time.

A somewhat different line of work also came about in trying to speed up the basic construction. Instead of trying to achieve a sparse matrix S, they tried to achieve an S which could be quickly applied to a vector. The motivation was that if vector x's mass is not well-spread, e.g., it is sparse, then sampling is a very poor way to estimate the ℓ_2 norm of x, as typically most samples will be 0.

Ailon and Chazelle[2] came up with an ingenious extension of this idea. In order to deal with vectors x that are not well-spread, they defined the matrix S as $S = MHD$, where:

- $M \in \mathbb{R}^{r \times n}$ is a sparse random matrix. Its entries are independent random variables, and each of them attains value 0 with probability $1 - q$, and a value drawn from the normal distribution with zero mean and variance $1/q$ with probability q. Here $q \in (0, 1)$ is a "sparsity parameter", which can be chosen as $1/n$ times a factor polylogarithmic in n.

- $H \in \mathbb{R}^{n \times n}$ is a Walsh matrix, it acts as a (scaled) isometry and that, given x, the product Hx can be evaluated by $O(n \log n)$ arithmetic operations by a Fast Fourier Transform algorithm.

- D is a diagonal matrix with independent random ± 1 entries.

Matrix S can be applied to a vector in $O(n \log n)$ time and to a $n \times d$ matrix in $O(nd \log n)$ time. This was further improved to $O(nd \log(d/\varepsilon \log n))$ time. This construction which is often referred to as *Fast Johnson Lindenstrauss Transform*(FJLT) is significantly faster than the previous $O(\mathsf{nnz}(x)\varepsilon^{-1} \log(d/\delta))$ time for many reasonable settings of the parameters, e.g., in a number of numerical linear algebra applications in which $1/\delta$ can be exponentially large in d.

3.2.2 Hashing

Although there is a lower bound by[9] that states "any JLT requires $\Omega(\varepsilon^{-1} \log{(n/\delta)}/\log{(1/\varepsilon)})$ non-zero entries per column.", a recent work by Clarkson and Woodruff[4] showed that it is indeed **possible** to achieve a sparse ℓ_2 subspace embedding in $O(\mathsf{nnz}(A))$ time. The key to bypass the lower bound is that the matrix $S \in \mathbb{R}^{r \times n}$ they build is not a JLT, but is constructed based on the famous *CountSkech* from data stream literature[3].

They define two hash functions $h : [n] \to [r]$ and $\sigma : [n] \to \{-1, 1\}$. Then for each column $S_{:,i}$ of S, choose a random row $h(i) \in \{1, 2, \cdots, r\}$, and a random element of $\sigma(i) \in \{-1, 1\}$, and set $S_{h(i),i} = \sigma(i)$. Every other element of that column is set to zero. Thus, S has only a single non-zero entry per column. This construction is called *"sparse embedding matrix"*, has only $r = O(d^2/\varepsilon^2 \text{polylog}(d/\varepsilon))$ number of rows, and one non-zero entry per column, therefore SA can be computed in $O(\mathsf{nnz}(A))$ time.

3.3 Random Projection for Low-Rank Approximation

The Johnson Lindenstrauss Transform was first used by Sarlos[10] to speadup regression, matrix approximation and matrix multiplication. Specifically for matrix approximation, he showed projecting A onto the low-rank subspace provided by a JLT matrix S, i.e. SA, gives a good approximation to the best rank k subspace A_k. Below, we state his theorem without proving it.

Theorem 3.3.1. *Let $A \in \mathbb{R}^{n \times d}$ be the input matrix, and $S \in \mathbb{R}^{r \times n}$ be Johnson-Lindenstrauss matrix with iid zero-mean ± 1 entries and $r = O(k/\varepsilon + k \log k)$, then with probability at least $1/2$ it holds that*

$$\|A - \pi_{SA}(A)\|_F \leq (1 + \varepsilon)\|A - A_k\|_F$$

This algorithm runs in two passes over A, one for computing SA and another for projecting A onto SA, requires $O(\mathsf{nnz}(A)r + (d + n)r^2)$ time and $O((d + n)r^2)$ space.

Bibliography

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[2] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563. ACM, 2006.

[3] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, Languages and Programming*, pages 693–703. Springer, 2002.

[4] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2013.

[5] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse johnson: Lindenstrauss transform. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 341–350. ACM, 2010.

[6] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[7] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[8] Jiří Matoušek. On variants of the johnson–lindenstrauss lemma. *Random Structures & Algorithms*, 33(2):142–156, 2008.

[9] Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 117–126. IEEE, 2013.

[10] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 143–152. IEEE, 2006.

[11] David P Woodruff. Sketching as a tool for numerical linear algebra. *arXiv preprint arXiv:1411.4357*, 2014.