Lecture 18: Fast JL dimension reduction

Prof. Moses Charikar *Scribes: Vatsal Sharan*

1 Overview

Recall that the Johnson-Lindenstrauss transform (JLT) projects n points in $\mathbb{R}^d$ to $\mathbb{R}^k$ where $k = O(\epsilon^{-2}\log n)$ such that all pairwise distances are distorted by at most a $1 \pm \epsilon$ multiplicative factor with high probability. Note that the JLT is a linear map from $\mathbb{R}^d \to \mathbb{R}^k$ which takes time $O(dk)$ to compute. We can also think of the JLT as providing a guarantee that for any $x \in \mathbb{R}^d$, with probability $1-\delta$, the L_2 norm of x is preserved upto a multiplicative factor of $1 \pm \epsilon$ under the transformation $x \to \Pi x$, where $\Pi \in \mathbb{R}^{k\times d}$ is the projection matrix, for $k = O(\epsilon^{-2}\log n)$. In this lecture we'll explore variants of the JLT which speed up the computation.

2 Previous work

Achlioptas [1] gave a constant factor improvement for the running time of the JLT by sampling the entries of the projection matrix from $\{-1, 0, +1\}$ instead of the standard Normal distribution. Kane and Nelson [2] showed that having Π which has only $\tilde{O}(\epsilon^{-1}\log n)$ non-zero entries in each column suffices to preserve distances. Note that projection using the sparse Π only takes time $O(\epsilon^{-1}d\log n)$. They also showed a lower bound of $\tilde{O}((\epsilon\log 1/\epsilon)^{-1}\log n)$ on the sparsity of Π in order to preserve ℓ_2 distance for all x. The idea behind the lower bound is that the projection will fail for sparse x if the sparsity of Π if too low, as we might miss all non-zero entries of x.

3 Fast JL Transform

Ailon and Chazelle [3] gave an algorithm which beats the lower bound of Kane and Nelson, by providing a high-probability guarantee for any x instead of a worst-case bound for all x. They use a projection matrix Π of the following form-

$$\Pi = [P]_{k\times d}[H]_{d\times d}[D]_{d\times d} \tag{1}$$

Here P is a sparse projection matrix where every entry is 0 with probability $1-q$ and is drawn from a Normal distribution with mean 0 and variance q^{-1} otherwise. D is a diagonal matrix where $D_{ii} = \{\pm 1\}$ with equal probability. H is the Walsh-Hadamard matrix. The main intuition is to express the original signal in a different basis, such as the Fourier basis or the Walsh-Hadamard basis. Any signal which is originally sparse will be dense in the new basis and this allows us to bypass the lower bound for sparse x.

The key point is that the matrix multiplications (Dx) and $H(Dx)$ can be computed efficiently using the special structure of the matrices H and D. Note that D is diagonal and hence Dx can be computed in time $O(d)$ time. $H(Dx)$ can be computed in time $O(d \log d)$ by using the recursive structure of the H matrix (think of the FFT matrix which allows us to compute FFT in time $O(d \log d)$).

The key benefit of using the transformation HD is the following property-

$$\mathbb{P}\left[\| HDx \|_\infty \geq \sqrt{\frac{\log(d/\delta)}{d}} \right] < \delta \tag{2}$$

Note that this is almost the best possible spreading, as $\| HDx \|_2 = 1$ hence on an average each entry would be $\sqrt{\frac{1}{d}}$ hence the spreading is almost optimal.

Computing the projection $P(HDx)$ takes time $Kdq = \frac{\log n}{\epsilon^2} d \frac{\log^2 n}{d} = \frac{\log^3 n}{\epsilon^2}$. Hence the total time is $O(d \log d + \epsilon^{-2} \log^3 n)$. Let's compare this to Kane and Nelson's algorithm which takes time $O(\epsilon^{-2} nd \log d)$. The FJLT outperforms this when $\log d \ll \epsilon^{-2} \log n$, which is often the case.

4 Solving system of linear equations with noise

Suppose we want to solve the system of linear equations $Ax + \epsilon = b$, where A is a $n \times d$ matrix, x is a d-dimensional vector, b is a n dimensional vector, ϵ random noise independent of x. If we are trying to minimize the ℓ_2 error $\| Ax - b \|_2$ and $A^T A$ is invertible, the optimal value of x is given by the well-known least squares solution $x = (A^T A)^{-1} A^T b$ where $(A^T A)^{-1} A^T = A^\dagger$ is also referred to as the Moore-Penrose pseudoinverse of A. Note that computing the least squares solution will take time $O(nd^2)$. We will focus on the setting where n or the number of data points is very large hence we want to reduce the dependence of the runtime on n. We can also consider other norms, for the ℓ_1 norm the problem can be formulated as an LP, which also takes time polynomial in the number of constraints n to solve.

4.1 Speeding up computation using Fast JLT

Recall that the objective is to minimize $\| Ax - b \|_2$. If we can project the data into a lower dimension which still preserves ℓ_2 distances within a factor $1 \pm \epsilon$, then we could still solve the original problem approximately. Based on this idea, we consider a $r \times n$ projection matrix S, and the optimization problem-

$$\min \| (SA)x - (Sb) \|_2 \tag{3}$$

We will set $r = O(\epsilon^{-2} d)$, which will give us significant savings in the runtime if $n >> \epsilon^{-2} d$. In order to still be able to approximately solve the original problem, we need our projection S to satisfy-

$$(1 - \epsilon) \| Ax - b \|_2 \leq \| S(Ax - b) \|_2 \leq (1 + \epsilon) \| Ax - b \|_2 \tag{4}$$

Consider a $n \times (d+1)$ matrix U with orthonormal columns such that colspace(U) = colspace$([A\ b])$. Sample the entries of our projection matrix S from $N(0, 1/r)$. We claim that SU is a set of independent random vectors. To verify, note that each row of SU is definitely independent of all other rows as the rows of S are independent. We claim that the entries within each row are also independent. To verify, note that each entry of SU is a Gaussian random variable, and Gaussian random variables are independent if they are uncorrelated. It is easy to verify that the entries in each row are uncorrelated as the columns of U are orthogonal.

Let $Ax - b = Uy$. Note that $\| Uy \|_2 = \| y \|_2$ as U is orthonormal. We need to ensure that $\| SUy \|_2 \approx \| y \|_2$. If all the singular values of $SU \in [1-\epsilon, 1+\epsilon]$, then $\| SUy \|_2 \approx \| y \|_2$. By using an $\epsilon-$net argument, Rudelson and Vershynin [4] showed that taking $r = O(\epsilon^{-2} d)$ suffices to ensure that all the singular values of $SU \in [1-\epsilon, 1+\epsilon]$, with probability $1 - e^{-d}$. Hence ℓ_2 distances are approximately preserved under this transformation.

In order to speed-up the projection step, we can now use the FJLT instead of the JLT. This brings the total time to $O(rd \log d) + \text{poly}(d/\epsilon)$, which has no dependence on n and can be a significant saving if the number of constraints/observations n is large.

In the next lecture we'll see a further improvement of the result due to Clarkson and Woodruff [5] who obtained a surprising runtime of $O(\text{nnz}(A)) + \text{poly}(d/\epsilon)$. They used the count-sketch matrix for their construction.

References

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

[2] Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):4, 2014.

[3] Nir Ailon and Bernard Chazelle. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

[4] Mark Rudelson and Roman Vershynin. Non-asymptotic theory of random matrices: extreme singular values. *arXiv preprint arXiv:1003.2990*, 2010.

[5] 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.