Randomized Algorithms.
Lecture Notes on Faster Dimensionality Reduction.
Kasper Green Larsen.

1 Faster Dimensionality Reduction

In this lecture note, we survey three different approaches used to speed up the Johnson-Lindenstrauss transform [3]. Recall from last lecture that the classic Johnson-Lindenstrauss transform works by drawing a random matrix $A \in \mathbb{R}^{m \times d}$ with coordinates i.i.d. either $\mathcal{N}(0,1)$ distributed, or uniform random among $\{-1,+1\}$ and then embedding a vector $x \in \mathbb{R}^d$ to $m^{-1/2}Ax$. Here $m = O(\varepsilon^{-2} \lg n)$ if one wants to preserve all pairwise distances to within $(1 \pm \varepsilon)$ and embedding time is $O(dm)$. The target dimensionality is known to be optimal via a result by Larsen and Nelson [5].

1.1 Sparse Embeddings

The first approach to speeding up JL is to use an embedding matrix A that is *sparse*. A sparse matrix is one with few non-zeroes. In particular, if one can use an embedding matrix A where each column has only t non-zeroes, then the embedding time improves to $O(dt)$. Moreover, in many applications, one also knows that the data vectors/points are sparse. Consider as an example a collection of documents. A classic representation of such documents in machine learning is as a "bag of words". That is, one creates a vector with one coordinate per word in the English dictionary and then each coordinate gives a count of how many times that word occur in the text. With this representation, documents with similar words map to points that are close and vice versa. In this setup data vectors are very sparse. The sparsity of a vector x is denoted $\|x\|_0$ which equals the number of non-zeroes in x. With this notation, the embedding time of classic JL is $O(m\|x\|_0)$ and with a matrix having t-sparse columns, it is $O(t\|x\|_0)$.

A series of papers proposed increasingly sparse embedding matrices, culminating with the work of Kane and Nelson [4]. They showed that the following construction with high probability preserves all pairwise distances for a set of n points: For each column of the matrix $A \in \mathbb{R}^{m \times d}$, pick a uniform random set of $t = O(\varepsilon^{-1} \lg n)$ rows and assign the corresponding entries either -1 or $+1$ uniformly at random and independently. Then embed x as $t^{-1/2}Ax$. Hence the embedding time compared to standard JL went from $O(\|x\|_0\varepsilon^{-2} \lg n)$ to $(\|x\|_0\varepsilon^{-1} \lg n)$, that is, an ε^{-1} improvement. The target dimensionality m remains optimal $O(\varepsilon^{-2} \lg n)$. We will not prove that the transform has this guarantee, but only show that the *expected* length of the embedded vector is the same as the original vector:

$$\begin{aligned}
\mathbb{E}[\|t^{-1/2}Ax\|_2^2] &= t^{-1}\mathbb{E}[\|Ax\|_2^2] \\
&= t^{-1}\sum_{i=1}^{m}\mathbb{E}\left[\left(\sum_{j=1}^{d} a_{i,j}x_j\right)^2\right] \\
&= t^{-1}\sum_{i=1}^{m}\left(\sum_{j=1}^{d}\mathbb{E}[a_{i,j}^2x_j^2] + \sum_{j=1}^{d}\sum_{h\neq j}\mathbb{E}[a_{i,j}a_{i,h}x_jx_h]\right) \\
&= t^{-1}\sum_{i=1}^{m}\sum_{j=1}^{d}\mathbb{E}[a_{i,j}^2]x_j^2 \\
&= t^{-1}\sum_{i=1}^{m}\sum_{j=1}^{d}(t/m)x_j^2 \\
&= \sum_{j=1}^{d}x_j^2 \\
&= \|x\|_2^2.
\end{aligned}$$

In the above, we used that $\mathbb{E}[a_{i,j}a_{i,h}] = \mathbb{E}[a_{i,j}]\mathbb{E}[a_{i,h}] = 0$ by independence and the fact that $a_{i,j}$ is symmetric around 0. Moreover, we used that $\mathbb{E}[a_{i,j}^2] = t/m$ since $a_{i,j}$ takes either the value -1 or 1 with probability t/m and in both cases, $a_{i,j}^2$ takes the value 1.

Nelson and Nguyen [6] later proved a lower bound showing that any sparse embedding matrix A must have $t = \Omega(\varepsilon^{-1}\lg n/\lg(1/\varepsilon))$, i.e. the upper bound of Kane and Nelson [4] can be improved by at most a $\lg(1/\varepsilon)$ factor. Settling the largest possible sparsity remains a fundamental open problem in dimensionality reduction.

Feature Hashing. Another line of work tries to improve the embedding time even further by making assumptions on the input data. Consider again the bag of words example. If one removes very frequently occuring words such as "the", then it seems reasonable to assume that the vector representing a document has no large coordinates. More formally, the ratio between $\|x\|_\infty$ (the largest coordinate) and $\|x\|_2$ is small. Under such assumptions, it is possible to speed up the sparse transforms from above. Feature hashing by Weinberger et al. [7] takes this to the extreme by letting A have exactly one non-zero per column, chosen at a uniform random row and with a value that is uniform among -1 and $+1$. So feature hashing is really the construction of Kane and Nelson with $t = 1$. Therefore it follows from the above calculations that the expected length of embedded vectors are correct. Feature Hashing clearly has the fastest possible embedding time of just $O(\|x\|_0)$.

This of course says nothing about the probability of preserving distances to within $(1 \pm \varepsilon)$. A sequence of papers tries to understand for which ratios of $\|x\|_\infty/\|x\|_2$ that Feature Hashing works. This was settled by Freksen, Kamma and Larsen [2] who showed that Feature Hashing into the optimal $m = O(\varepsilon^{-2}\lg n)$ dimensions has the Johnson-Lindenstrauss guarantees (preserve all distances to within $(1 \pm \varepsilon)$) exactly when

$$\frac{\|x\|_\infty}{\|x\|_2} = O\left(\varepsilon^{-1/2}\cdot\min\left\{\frac{\lg(1/\varepsilon)}{\lg n}, \sqrt{\frac{1}{\lg n}}\right\}\right).$$

And more generally, if one embeds into $m \geq \varepsilon^{-2}\lg n$ dimensions, then distances are preserved when

$$\frac{\|x\|_\infty}{\|x\|_2} = O\left(\varepsilon^{-1/2}\cdot\min\left\{\frac{\lg(\varepsilon m/\lg n)}{\lg n}, \sqrt{\frac{\lg(\varepsilon^2 m/\lg n)}{\lg n}}\right\}\right).$$

1.2 Fast Johnson-Lindenstrauss Transform

Instead of using sparse matrices for embeddings, another approach is to use matrices for which there are efficient algorithms to compute the product Ax. The Fast Johnson-Lindenstrauss transform by Ailon and Chazelle [1] was the first to introduce this direction. Their first observation is very similar to what is used in Feature Hashing: If data vectors have only small coordinates, i.e. the ratio $\|x\|_\infty/\|x\|_2$ is small, then one can use very sparse matrices for the embedding. The main idea now is to first multiply x with a matrix which ensures that coordinates becomes small, and then use a sparse matrix afterwards.

Formally, this is done as follows: Assume without loss of generality that d is a power of 2 (one can always pad with zeroes). Let $\bar{H}_d$ be the $d \times d$ Walsch-Hadamard matrix. $\bar{H}_d$ can be defined recursively as follows: The 2×2 Walsch-Hadamard matrix H_2 has $\bar{h}_{1,1} = \bar{h}_{1,2} = \bar{h}_{2,1} = 1$ and $\bar{h}_{2,2} = -1$. Clearly all rows are orthogonal. The $2d \times 2d$ Walsch-Hadamard matrix $\bar{H}_{2d}$ consists of four $d \times d$ Walsch-Hadamard matrices $\bar{H}_d$, one in each of the four quadrants. The matrix in the lower right corner is negated.

$$\bar{H}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\bar{H}_{2d} = \begin{pmatrix} \bar{H}_d & \bar{H}_d \\ \bar{H}_d & -\bar{H}_d \end{pmatrix}$$

$$\bar{H}_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Again, all rows of $\bar{H}_{2d}$ are orthogonal. This can be seen as follows: For two distinct rows in the top half of the matrix, the orthogonality follows by induction. Similarly for two in the bottom half. For one row in the top half of the matrix and one from the bottom half, orthogonality follows since the top row vector has the form $(v \circ v)$ and bottom has the form $(w \circ -w)$ where $\circ$ denotes concatenation. Hence their inner product is $\langle v, w\rangle - \langle v, w\rangle = 0$.

The crucial part about Walsch-Hadamard matrices, is that one can compute the matrix-vector product $\bar{H}_d x$ efficiently. Write x as $\binom{x_1}{x_2}$ where x_1 is the first $d/2$ coordinates of x and x_2 is the last $d/2$ coordinates. Then $\bar{H}_d x = \binom{\bar{H}_{d/2}x_1 + \bar{H}_{d/2}x_2}{\bar{H}_{d/2}x_1 - \bar{H}_{d/2}x_2}$. Thus if we compute $\bar{H}_{d/2}x_1$ and $\bar{H}_{d/2}x_2$ recursively, we can compute $\bar{H}_d x$ in $O(d)$ time. The total time to compute $\bar{H}_d x$ thus satisfies the recurrence $T(d) = 2T(d/2) + O(d)$, which solves to $O(d \lg d)$.

Let H be the normalized Walsch-Hadamard matrix, i.e. $H = d^{-1/2}\bar{H}_d$. Then all rows are still orthogonal and the norm of any row is 1. Hence H is an orthogonal matrix and $\|Hx\|_2^2 = \|x\|_2^2$ for all vectors $x \in \mathbb{R}^d$. That is, H preserves all norms. The construction of Ailon and Chazelle is then as follows: Draw a random diagonal matrix $D \in \mathbb{R}^{d \times d}$ where each entry is uniform among -1 and 1. Draw a matrix $P \in \mathbb{R}^{m \times d}$ with $m = O(\varepsilon^{-2} \lg n)$ such that, for every entry, with probability $1 - q$ we set the entry to 0, and otherwise, we let the entry be $\mathcal{N}(0, (mq)^{-1})$ distributed. The expected number of non-zeroes of P is thus qmd. The embedding of x is then $PHDx$. Computing Dx takes $O(d)$ time. Multiplying the result with H takes $O(d \lg d)$ time and multiplying with P takes expected $O(qmd)$ time. Ailon and Chazelle showed that the construction achieves the Johnson-Lindenstrauss guarantee if one sets $q = O(\lg^2 n/d)$, resulting in an embedding time of $O(d \lg d + m \lg^2 n) = O(d \lg d + \varepsilon^{-2} \lg^3 n)$.

Proving that the construction satisfies the Johnson-Lindenstrauss guarantees consists of two steps. In the first step, one shows that $\|HDx\|_\infty = O(\sqrt{\lg(nd)/d})$ with probability $1 - 1/n^3$ for any unit vector $x \in \mathbb{R}^d$. That is, HD ensures that most coordinates of x are small. In the second step, one shows that, assuming $\|HDx\|_\infty = O(\sqrt{\lg(nd)/d})$, the application of P preserves norms to within $(1 \pm \varepsilon)$ with good probability. Here we remark that HD preserves the norm of all vectors perfectly (but the dimension is still d). We will only prove the first step as the second is significantly more involved.

So consider the i'th coordinate of HDx. Each entry of H is either $-d^{-1/2}$ or $d^{-1/2}$ and D multiplies a random sign onto each coordinate of x. Hence the i'th coordinate is distributed as $\sum_i \sigma_i d^{-1/2} x_i$ where the

σ_i's are independent and uniform among -1 and 1. Clearly $\mathbb{E}[(HDx)_i] = 0$ by linearity of expectation. We will now invoke Hoeffding's inequality:

Theorem 1 (Hoeffding's Inequality)**.** *Let $X_1, \ldots, X_d$ be independent random variables where X_i takes values in the interval $[a_i, b_i]$. Let $X = \sum_i X_i$. Then*

$$\Pr[|X - E[X]| > t] < 2\exp\left(-\frac{2t^2}{\sum_{i=1}^d (b_i - a_i)^2}\right).$$

For our sum $\sum_i \sigma_i d^{-1/2} x_i$, we have that the random variable $X_i = \sigma_i d^{-1/2} x_i$ takes values in the interval $[-d^{-1/2}|x_i|, d^{-1/2}|x_i|]$ and thus $(b_i - a_i)^2 = (2d^{-1/2}|x_i|)^2 = 4d^{-1}x_i^2$. We thus have

$$\begin{aligned}
\Pr[|X - E[X]| > t] &< 2\exp\left(-\frac{2t^2}{\sum_{i=1}^d 4d^{-1}x_i^2}\right) \\
&= 2\exp\left(-\frac{2t^2}{4d^{-1}\|x\|_2^2}\right) \\
&= 2\exp(-t^2 d/2).
\end{aligned}$$

For $t = C\sqrt{\ln(nd)/d}$ for a big enough constant C, this probability is less than $1/(dn^3)$. A union bound over all d coordinates shows that $\|HDx\|_\infty = O(\sqrt{\ln(nd)/d})$ with probability at least $1 - 1/n^3$.

Let us conclude by showing that $\mathbb{E}[\|Px\|_2^2] = \|x\|_2^2$ for all vectors x, i.e. P preserves norms in expectation. We will not show the $(1 \pm \varepsilon)$ guarantee. Each entry of P is distributed as the product of a Bernouilli variable $b_{i,j}$ taking the value 1 with probability q and 0 otherwise, and a variable $n_{i,j} \sim \mathcal{N}(0, (mq)^{-1})$. We thus get:

$$\begin{aligned}
\mathbb{E}[\|Px\|_2^2] &= \sum_{i=1}^m \mathbb{E}\left[\left(\sum_{j=1}^d b_{i,j} n_{i,j} x_j\right)^2\right] \\
&= \sum_{i=1}^m \sum_{j=1}^d \mathbb{E}[b_{i,j}^2 n_{i,j}^2 x_j^2] + \sum_{j=1}^d \sum_{h \neq j} \mathbb{E}[b_{i,j} b_{i,h} n_{i,j} n_{i,h} x_i x_j] \\
&= \sum_{i=1}^m \sum_{j=1}^d \mathbb{E}[b_{i,j}^2]\mathbb{E}[n_{i,j}^2] x_j^2 + \sum_{j=1}^d \sum_{h \neq j} \mathbb{E}[b_{i,j}]\mathbb{E}[b_{i,h}]\mathbb{E}[n_{i,j}]\mathbb{E}[n_{i,h}] x_i x_j \\
&= \sum_{i=1}^m \sum_{j=1}^d q(mq)^{-1} x_j^2 \\
&= \|x\|_2^2.
\end{aligned}$$

In the above, we used that $\mathbb{E}[n_{i,j}] = 0$ and $\mathbb{E}[b_{i,j}^2] = \mathbb{E}[b_{i,j}] = q$. We also used that for $n_{i,j} \sim \mathcal{N}(0, \sigma^2)$ random variables, we have $\mathbb{E}[n_{i,j}^2] = \sigma^2$. We also used independence of the entries in P to split the expectation of the product into the product of expectations.

References

[1] N. Ailon and B. Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, May 2009.

[2] C. B. Freksen, L. Kamma, and K. Green Larsen. Fully understanding the hashing trick. In *Advances in Neural Information Processing Systems 31*, NeurIPS '18, pages 5389–5399. Curran Associates, Inc., 2018.

[3] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[4] D. M. Kane and J. Nelson. Sparser Johnson–Lindenstrauss transforms. *J. ACM*, 61(1):4:1–4:23, Jan. 2014.

[5] K. G. Larsen and J. Nelson. Optimality of the johnson-lindenstrauss lemma. In C. Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 633–638. IEEE Computer Society, 2017.

[6] J. Nelson and H. L. Nguy˜ên. Sparsity lower bounds for dimensionality reducing maps. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 101–110. ACM, 2013.

[7] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1113–1120. ACM, 2009.