1 Introduction

1.1 JL Transform reminder

In our last lesson we discussed the JL Transform [JL84]. We showed that given $X \subset \mathbb{R}^n$ a group of $N$ vectors we can find (with high probability) a transformation $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^k$ where $k \ll n$ that will preserve distances between vectors in $X$ up to a pre-chosen ”distortion factor”.

More formally we showed that: given a failure probability $\delta > 0$ and a distortion parameter $\epsilon > 0$, there exists a random linear transform $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^k$ where $k = O\left(\log\left(\frac{N}{\delta}\right)\epsilon^2\right)$ that satisfies a “JL Property”.

The JL Property asserts that with probability $1 - \delta$, distances between vectors in $X$ are distorted to a maximum factor controlled by $\epsilon$:

$$Pr[\forall x, y \in X, (1 - \epsilon)||x - y|| \leq ||\Phi(x) - \Phi(y)|| \leq (1 + \epsilon)] \geq 1 - \delta$$

• (Note that by $|| \cdot ||$ we mean the $L_2$ norm, here and throughout this text)

1.2 Running time of JL Transform

The transform we used was a $k \times n$ random matrix $\Phi$ where each cell was an i.i.d Gaussian variable (we discussed that actually sub-Gaussian can suffice). This matrix can be quite big, and due to its cells being i.i.d. random variables it does not lend itself well to compression, meaning that it requires $O(kn)$ space to store and $O(kn)$ running time to apply on a given vector.

We will now discuss the Fast JL Transform that address this shortcoming. It shows the existence of a random transform that maintains the JL property while requiring $O(n \log n)$ running time to apply on a given vector. This will be beneficial if we want $k = \Omega(\log n)$.
2 The Fast JL Transform

2.1 Intuition: Trouble with JL Transform using a sparse matrix

A naive approach to finding a fast JL transform would be to replace the random transform $A$ with a sparse matrix $S$ where $(S)_{i,j} = \begin{cases} g_{i,j} & \text{with probability } q \\ 0 & \text{with probability } 1-q \end{cases}$ where $g_{i,j}$ are Gaussian r.v and $q$ is very small.

But this will not work. An intuition as to why this will not work is that there are certain “bad” vectors that will be very distorted. Namely sparse vectors, and as an intuitive extreme case we can think of vectors with a single coordinate that is non zero. Only the non zeros in $M$ that are aligned with this one coordinate will contribute to $Mx$, and these will be too rare to promise us a JL property. The Fast JL Transform will remedy this by first transforming $x$ using a randomized Fourier transform and only then applying the sparse $S$. this will guarantee that the input for $S$ is not sparse with overwhelming probability.

2.2 Fast JL Theorem

Nevertheless a Fast JL Transform is indeed possible as shown by N. Ailon and B. Chazelle [AB06]. Here is the Fast JL Theorem:

There exists a distribution of matrices $D$ such that if $A \in \mathbb{R}^{k \times n}$ is drawn from $D$, it satisfies the above mentioned JL Lemma and given $x \in \mathbb{R}^n$, computing $Ax$ can be done in $O(n \log n)$ time.

To prove this theorem we will first construct an appropriate random matrix $A \in D$ and then show it holds the JL property.

$A$ will be a product of a sparse random matrix, a Hadamard matrix and a random diagonal matrix:

$$A = SHD$$

- $S \in \mathbb{R}^{k \times n}$ is a sparse random matrix such that it’s cells are:

  $$s_{i,j} = \begin{cases} g_{i,j} & \text{with probability } q \\ 0 & \text{with probability } 1-q \end{cases}$$

  where $q = O\left(\frac{k^2}{n}\right)$ and $g_{i,j} \sim N(0,q^{-1})$ Gaussian r.v.s.

- $H \in \mathbb{R}^{n \times n}$ is an $n \times n$ Hadamard transform matrix. Note that Hadamard matrix is only defined for $n = 2^m$ for some natural positive $m$. This is not a problem since $n$ (the size of the input vectors) can always be extended to be a power of 2.

- $D \sim diag(d_1, ..., d_n)$ is a diagonal matrix where each $d_i$ is a random variable defined: $d_i = \begin{cases} +1 & \text{w.p. } \frac{1}{2} \\ -1 & \text{w.p. } \frac{1}{2} \end{cases}$
Before we continue note that $H$ has a fast Fourier transform and thus $HDx$ is computable in $O(n\log n)$. $S$ is sparse $k \times n$ and has a mean of $knq = k^3$ non zero elements. This means if we take $k = O(n^{1/3})$ then computation of $SHDx$ takes $O(n\log n + n) = O(n\log n)$ time as we desire.

2.3 Proving of Fast JL Theorem

Given a vector $x \in \mathbb{R}^n$, assume without loss of generality that $||x|| = 1$. Let us first look at:

$$\hat{x} \triangleq HDx$$

the Hadamard transform $H$ is orthogonal, and $D$ is orthogonal as well so we have $||x|| = ||\hat{x}||$.

Let us look at the transform $(HD)$: $D$ is acting on $H$ by randomly multiplying each column by $\pm 1$. $H$ being a Hadamard matrix has it's cells: $H_{i,j} \in \{+1, -1\}$ multiplied by a $\frac{1}{\sqrt{n}}$ normalization factor. This means that within a single row of $(HD)$, its elements can be viewed as independent r.v. with equal probability of being $+\frac{1}{\sqrt{n}}$ or $-\frac{1}{\sqrt{n}}$.

If we look back at $\hat{x}$ now, its coordinates look like this:

$$\hat{x}_i = \sum_{j=1}^{n} (HD)_{i,j}x_j \sim \sum_{j=1}^{n} \frac{x_j}{\sqrt{n}}$$

As we see the terms $(HD)_{i,j}x_j$ here are Bernoulli r.v. so we can use Hoeffding’s inequality to bound $|\hat{x}_i|:

$$Pr[|\hat{x}_i| > t] \leq 2e^{-\frac{t^2n}{2\sigma_x^2}} = 2e^{-\frac{t^2n}{2}}$$

Note the second equality holds since we assumed $||x|| = 1$. We will now choose $t = \sqrt{\frac{k}{n}}$ and we get:

$$Pr[|\hat{x}_i| > \sqrt{\frac{k}{n}}] \leq 2e^{-\frac{k}{n}}$$

We are interested in bounding all $\hat{x}_i$ at the same time. these are $n$ independent events to to bound them together the probability is:

$$Pr[\forall \hat{x}_i : |\hat{x}_i| > \sqrt{\frac{k}{n}}] \leq 2ne^{-\frac{k}{n^2}} = e^{-\frac{1}{2} + \log(2n)}$$

As we have seen in the discussion before we can assume $k = \Omega(\log n)$, so we can rewrite the expression $1 - e^{-\frac{1}{2} + \log(2n)} = 1 - e^{-kc}$ for some large enough $c > 0$.

(1) This means that with probability $1 - e^{-kc}$ we get for all $\hat{x}_i$:

$$|\hat{x}_i| < \sqrt{\frac{k}{n}}.$$
• **Intuitively** this event is good for us, since this counters the sparseness of $x$ problem we had with the naïve approach: having a single (or a few) coordinates that are very big, leaving the rest very small. Here the coordinates of $\hat{x}$ are bound and cannot be too big.

We will assume this “good” event from here - and use it to show our transform holds the JL property. when we are done we will go back and make sure the probability of this assumption failing is not too high.

We made a claim without proving it in the lesson that in the case of our ‘good’ event, the worst situation possible for us is a vector $\hat{x}$ that looks like this:

$$\hat{x} = (\sqrt{\frac{k}{n}}, ..., \sqrt{\frac{k}{n}}, 0, ..., 0)$$

where the number of non zero coordinates is $n/k$. We assume here (without loss of generality) that the first $n/k$ coordinates are the non zero ones.

Now let us look at $y = SDx = S\hat{x}$. It looks like this:

$$y = \left( \begin{array}{c} s_{i,j} \end{array} \right) \cdot \left( \begin{array}{c} \sqrt{\frac{k}{n}} \\ \vdots \\ 0 \\ \vdots \\ 0 \end{array} \right)$$

recall that: $s_{i,j} = \begin{cases} g_{i,j} & \text{w.p. } q \\ 0 & \text{w.p. } (1-q) \end{cases}$ and $g_{i,j} \sim N(0, q^{-1})$

Let us now bound $\|y\|^2 = \sum_{i=1}^{n} y_{i}^2$:

It is clear that in $S$ we are interested only in the first $n/k$ columns since the rest do not affect the outcome. We will define a set of random variables counting the non zero entries in each row:

$$z_{i} = \#\{S_{i,j} \neq 0 | j \in [1..n/k]\}$$

$z_i$ are therefore Binomial r.v.:

$$z_{i} \sim Bin(n/k, q)$$

Now, assume $z_i$ are fixed (assume we have prechosen $S$), then let’s evaluate $y_{i}$ given $z_{i}$: these are a sum of $z_{i}$ Gaussian variables multiplied by $\sqrt{\frac{k}{n}}$:

$$y_{i}|z_{i} \sim N(0, z_{i}q^{-1}\sqrt{\frac{k}{n}})$$
Here we can use Chernoff to bound all $z_i$ around their common mean. When doing so we get: $\Pr[\forall i: \frac{q}{k} q_{1/2} \leq z_i \leq \frac{q}{2} q_{2}] \geq 1 - 2e^{-\frac{q}{k} q_{\tilde{c}}} \text{ for some constant } \tilde{c}$. Note we are bounding all $z_i$ at the same time here.

Here we made a claim in class without proving it: If there is a number $\alpha$ such that for all $z_i$: $\frac{q}{k} \leq z_i \leq 2\alpha$ then $\|y\|_2^2$ is distributed like Chi Square:

$$\|y\|_2^2 = \sum_{i=1}^{n} y_i^2 \sim \chi^2(k) \cdot \beta$$

$\beta$ is some normalization constant here. Now using the Chi Square distribution we can prove the JL property the same way we did in the previous lesson when talking about JL.

To finish the argument we need to determine $q$. We will take $q = \tilde{c} \frac{k^2}{n}$, this value will promise us that the probability of bounding $z_i$ as we have done in (2) is $\geq 1 - 2e^{-\frac{q}{k} q_{\tilde{c}}} = 1 - 2e^{-k}$.

Now if we go back and look at the entire argument we can make sure the probability of 'success' is high: meaning the probability of all the events we have assumed to get the Chi Square distribution and the JL property are high. We will bound the probability of failure of our assumptions: recall (1), this 'good' event was our first assumption, the probability of success was

$$1 - e^{-k \tilde{c}}$$

Assuming (1) happened, we then assumed (2) as well with a probability of

$$1 - 2e^{-k}$$

The probability of both events happening is therefore:

$$(1 - e^{-k \tilde{c}})(1 - 2e^{-k}) = 1 - e^{-k}(e^\tilde{c} + 2) + 2e^{-k(\tilde{c}+1)} \geq 1 - e^{-k \hat{c}}$$

for some $\hat{c} > 0$. This sets $\delta = e^{-k \hat{c}}$ for the JL lemma stated in the beginning of the discussion and is in-line with the requirement that $k = O(\log(N/\delta))$ (satisfies that $k^{-1} = O(\tilde{c})$, if we disregard the other variables).

3 Using FJLT for Linear Regression (Only the start - the rest in next lecture)

3.1 Linear regression Introduction

In linear regression we get a matrix $M \in \mathbb{R}^{n \times d}$, we say that it's rows represent 'objects' and its columns represent 'features' (Within a computer science context. In other contexts these are usually called by different names). We also get a vector $b \in \mathbb{R}^n$ of labels (the known labels of the objects in $M$). Our goal is to explain the labels using the objects, or to predict a label of a new object given its features.
More formally we are looking for a vector \( x \in \mathbb{R}^d \) such that \( Mx = b \). This is usually not possible, namely because \( n \gg d \) and so it is not possible to span \( b \) using the rows of \( M \). So in linear regression we try to minimize the mean square error: we search for:

\[
x^* = \arg\min_{x \in \mathbb{R}^d} \{ ||Mx - b||_2 \} = \arg\min_{x \in \mathbb{R}^d} \{ \sum_{i=1}^{n} ((M)x - b_i)^2 \}
\]

The solution to this has a closed form and it is:

\[
x^* = M^\dagger b = (M^TM)^{-1}M^Tb
\]

\( M^\dagger \) is called the Pseudo Inverse of \( M \).

### 3.2 Dimensionality reduction

The time to calculate \( M^\dagger b \) is \( O(nd^2) \) (using naive matrix multiplication). For big matrices this can be very high and we will show an improvement on this by Sarlos [S06]. It is done using FJLT on the expense of a small amount of allowed ‘distortion’. (Note we can improve running time by using fast matrix multiplication when calculating \( M^\dagger \), but we will be able to use in the proposed solution as well so we will ignore this type of improvement).

The solution will be to use JL to reduce the dimensionality of the problem. Meaning to minimize an expression of the sort: \( ||AMx - Ab|| \) where \( A \) is a JL matrix.

Formally if we want to find: \( x^* = \arg\min_{x \in \mathbb{R}^d} \{ ||Mx - b||_2 \} \), we instead use an estimator:

\[
\tilde{x} = \arg\min_{x \in \mathbb{R}^d} \{ ||AMx - Ab||_2 \}
\]

We would like to claim that since \( A \) is JL then

\[
||M\tilde{x} - b||_2 \leq \frac{1}{1 - \epsilon} ||(AM)x^* - Ab||_2 \leq \frac{1 + \epsilon}{1 - \epsilon} ||Mx^* - b||_2
\]

By using the JL property for both inequalities but this logic is flawed. The problem is that \( \tilde{x} \) is dependent on \( A \) which in turn has to be dependent on \( \tilde{x} \) since by the JL lemma \( A \) is a good estimator for a finite number of vectors, where \( \tilde{x} \) has to be one of them in order for this to work. In other words the problem is that a JL transform has been shown to preserve the norm of a finite number of vectors, and here we want it to preserve the norm for the entire linear subspace \( \{Mx - b | x \in \mathbb{R}^d \} \) in order to estimate the \( \arg\min \) function.

The solution will be surprising: we will show that a JL matrix in fact can preserve the norm for a desired linear subspace. In order to do this we will use \( \epsilon \)-nets/\( \epsilon \)-grids
3.3 Subspace embedding using JL

Let $S \subseteq \mathbb{R}^n$ a linear subspace of dimension $d$. $A \sim JL, A \in \mathbb{R}^{k \times n}$ where $k = O\left(\frac{d}{\lambda \text{log} \epsilon}\right)$. Then with high probability $(1 - \delta)$ it holds that:

$$\forall z \in S, (1 - \epsilon)\|z\|_2 \leq \|Az\|_2 \leq (1 + \epsilon)\|z\|_2$$

This is the claim we want to prove and we will prove it in the next lesson.

References

