Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Interim Report 2018

Project Title:	**Dimension Reduction For Big Data**
Student:	**Devin V. Nanayakkara**
CID:	**00932538**
Course:	**EIE4**
Project Supervisor:	**Dr Cong Ling**
Second Marker:	**Prof. A. Manikas**

Contents

1. Project Specification

1.1 Motivation

The existence of data has changed drastically since the industrial revolution in the early 1800. Thereafter, data has grown in size and complexity, giving rise to the age of Big Data. Some big data applications are as follows:

- Image and video storing where each pixel is represented by a dimension and an extra time dimension for videos [3]
- Genetics research where the data storage demands will run to 2 – 40 exabytes (1 exabyte = 10^{18} bytes) by 2025 [14]
- Big data analytics in video surveillance [15]
- Healthcare monitoring systems

The term big data arises when data is very large and complex that traditional data processing or running learning algorithms on a standard machine becomes inadequate. Some challenges faced due big data include [13]:

- Capturing data: inability to obtain all the relevant information
- Difficulty in storing data
- Complex data analysis
- Difficulty in data querying and updating
- Expensive to transmit data between machines

This project attempts to address the computational and statistical difficulties that are caused by big data. The idea is to embed the high-dimensional space of big data to a low-dimensional subspace through random projections. The project develop, test, evaluate, and compare random projection techniques for dimensional reduction. In this project, the projection will be from a given high-dimensional euclidean space data to a random low-dimensional euclidean subspace. This will help to evaluate the lower bounds of the embedding dimension and its performance on a standard data learning or statistical algorithm. The performance will be determined by the corresponding error rate resulting from a test dataset.

1.2 Objectives

The objectives for this project are as follows:

1. Implement 2 random projection methods for dimensional reduction
2. Evaluate lower bounds of the embedded dimension for each method
3. Evaluate the error in reducing the dimension for each method
4. Apply learning and statistical algorithms on the embedded subspace and evaluate its performance
5. Compare the implementations and results to identify the best of the 2 methods

These objectives are set to asses the success of the project. Refer to the implementation plan (section 4) and evaluation plan (section 5) for the evaluation of these objectives.

1.3 Deliverables

This project has academic deliverables and internal deliverables. The academic deliverables are the deadlines set by the university. They are namely the inception report, interim report, final report, and the final presentation. Where as the internal deliverables are mainly towards the planning and success of the project. The MATLAB scripts that implement the random projection techniques, evaluation, and comparison will be the main deliverables for this project along with the final report.

2. Background

2.1 The Curse of Dimensionality

Define a dataset $X_{n\times m}$, where n is the number of rows (parameters/dimensions) and m is the number of columns (observations/samples). In classical data processing, theres are small number of dimensions and large number of samples ($m > n$). But in modern applications there are large number of dimensions but a smaller sample size ($m \approx n$ or $m < n$). The curse of dimensionality [12] arises as the dimensions become much larger than the sample size. This leads to computational difficulties where the computation power of a machine is insufficient to handle large datasets. Another aspect is the intrinsic statistical difficulty where some data are left in isolation, providing false structures and overfitting which describes the noise and not the relationship within the data.

2.2 Random Projection

Random projections are proven to be a useful tool for dimension reduction in the euclidean space by various research institutes and researchers. It uses the pairwise distance between points to manipulate large datasets to small subspaces. In addition, such projections are proven to be computationally fast [4]. The basic idea in high-dimensional to low-dimensional space is shown in Figure 1.

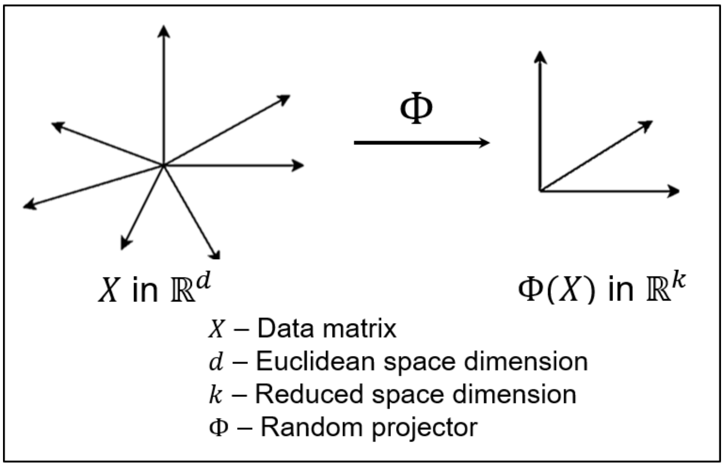

Φ
X in $\mathbb{R}^d$
$\Phi(X)$ in $\mathbb{R}^k$
X – Data matrix
d – Euclidean space dimension
k – Reduced space dimension
Φ – Random projector

Figure 1: Projecting the data space to a lower dimension

Random dimension reduction is used in many fields. In signal processing, compressed sensing reduces the number of samples through sparsity and hence reduces the cost and time for sensing. In coding theory, error-correcting codes and parity-check codes are used to detect and correct errors after receiving a transmission. In neuroscience [22], it was found that the brain compresses large-scale complex information is random ways. A question arising at this point is that why is a degrees of randomness necessary? This is because the information input is not always the same. Hence the parameters of a projection must be independent of the dataset. On some occasions data can arrive in a stream hence the projection or algorithm is unaware of the state or form of the next arrival. Hence for randomised dimensional reduction, universality laws were introduced [21]. The universality laws fundamentally builds relationships between random linear mappings and preserving geometric structure after embedding onto a smaller dimensional space.

2.2.1 Johnson-Lindenstrauss Transform

Random projections for dimensional reduction were revolutionised by the theorem introduced by William Johnson and Joram Lindenstrauss in the 1980s. They introduced the Johnson-Lindenstrauss (JL) transform which is given by the equation below.

$$\forall (x,y) \in X, \quad (1-\epsilon)\|x-y\|_2 \le \|\Phi(x)-\Phi(y)\|_2 \le (1+\epsilon)\|x-y\|_2 \tag{1}$$

What is implied by Equation 1 is that, projecting any points in the dataset X from a high-dimensional space to a low-dimensional space must, up to a distortion of $(1 \pm \epsilon)$, preserve pairwise distances [2]. Based on pairwise distances, they defined the random projector matrix $\Phi_{k\times d}$ with the rows being random unit vectors orthogonal to each other. However, this original JL projection has a computational time of order $k \times n$, where n is the size of the dataset X and the embedding dimension, $k \approx \epsilon^{-2} \log n$ [1, 3]. Since then numerous research took place to reduce the value of k whilst improving on the computation time and complexity.

2.3 Fast Johnson-Lindenstrauss Method 1

2.3.1 Previous Work

After introducing the JL projection, many amended and refined the original JL but the computation and complexity did not significantly improve. Through a clever observation by Dimitris Achlioptas [5], sparsity was brought into the random projection matrix. Using the properties of l_2 norm and distortion vectors between points, he defined the elements of the random projection matrix as follows:

$$\Phi_{i,j} = \begin{cases} \frac{\sqrt{3}}{d} & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \\ -\frac{\sqrt{3}}{d} & \text{with probability } \frac{1}{6} \end{cases} \tag{2}$$

A matrix is defined to be sparse when most of its elements are 0. In this case $\frac{2}{3}$ of the elements of Φ is 0, making Φ a sparse matrix. The sparsity of this matrix helps to reduce the computation time significantly because most of its elements are 0 hence these entries can be ignored. However there must be a restriction to limit the level of sparsity a matrix can have. This is due to instances when certain data points (vectors) are very sparse themselves.

$$\Phi = \begin{bmatrix} c_1 & 0 & c_6 & 0 & 0 & 0 \\ 0 & c_4 & 0 & 0 & c_9 & 0 \\ c_2 & 0 & 0 & c_8 & 0 & c_{11} \\ 0 & 0 & c_7 & 0 & 0 & 0 \\ c_3 & 0 & 0 & 0 & 0 & c_{12} \\ 0 & c_5 & 0 & 0 & c_{10} & 0 \end{bmatrix} \qquad x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

In the example above, Φ is a sparse random projection matrix with $\frac{2}{3}$ of its elements being 0 and x is a sparse vector. By observation, only the element c_8 is able to carry the information of x. Therefore in some instances information carried in very sparse vectors such as x can be ignored or isolated. Hence there was a necessity to densify the projection matrix to avoid loss of information due to too few non-zero coordinates.

2.3.2 Fast JL with Sparsity

Since the need to densify the matrix, Nir Ailon and Bernard Chazelle introduced the fast JL transform which included Fourier transform properties in addition to sparsity [3]. The random projection matrix was defined as:

$$\Phi_{k\times d} = S_{k\times d}.H_{d\times d}.D_{d\times d} \tag{3}$$

$H_{d\times d}$ represents an extended form of discrete Fourier transform over the field $\mathbb{F}_{n^d}$ called Walsh-Hadamard matrix [3] (where n is size of the dataset). The elements of the matrix are defined as:

$$H_{i,j} = d^{-\frac{1}{2}}(-1)^{<i-1.j-1>} \tag{4}$$

where $< a, b >$ is the dot product between a and b.

This matrix has a runtime of $O(d \log d)$. The underlying concept is based on the Uncertainty Principle for signal processing where "a signal cannot be localized in both signal's time domain and its spectrum's frequency domain" and the ability to obtain precise values (measurements) [6, 7]. The Walsh-Hadamard matrix is essentially preconditioning the input with a fast Fourier transform. This step densifies the isometry of any sparse vectors in the input [3]. There must exist a certain degree of randomness in the Fourier matrix in order to avoid dense vectors being sparse. Hence a random matrix needs to be included.

$D_{d\times d}$ is a random diagonal matrix whose elements are ±1 with probability $\frac{1}{2}$.

$$D = \begin{pmatrix} \pm1 & 0 & \cdots & 0 \\ 0 & \pm1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \pm1 \end{pmatrix}$$

This matrix adds a certain degree of randomness to the projection. Hence avoids sparsification of dense vectors. Note that the H and D are orthogonal square matrices and hence the projection from a high-dimensional space to a low-dimensional space results in no distortion [3]. Now that the input is preconditioned by densification of any sparse vectors, it is ready to be computed with the sparse matrix.

$S_{k\times d}$ is a sparse matrix with sparsity approximately k^3 non-zero entries. The elements of the matrix are an independent mixture, $S_{i,j} \sim \mathcal{N}(0, \frac{1}{q})$ with some probability q and $S_{i,j} = 0$ with probability $1-q$. Only k^3 elements are non-zeros and hence only these elements are required for computation of this matrix. So only a $O(k^3)$ factor is added to the runtime [3].

2.3.3 Method 1 Summary

This method has a computation time of $O(d \log d + |S|)$, where $d \log d$ is the runtime for the Walsh-Hadamard Fourier transform matrix and $|S|$ being the l_0 pseudo-norm of the sparse matrix ($|S|$: number of non-zero entries). When the embedding dimension (k) is very small, the runtime becomes approximately $O(d \log d)$ and $k \leq d^{\frac{1}{3}}$ [3]. The fast JL transform lemma is given below, for proofs refer [3].

Lemma 1 *Given a fixed set of X of n points in $\mathbb{R}^d$, $\epsilon < 1$, and $p \in \{1,2\}$, draw a matrix Φ from FJLT (Fast Johnson-Lindenstrauss Transform). With probability at least $\frac{2}{3}$, the following 2 events occur:*

1. *For any $x \in X$,*

   $$(1-\epsilon)\alpha_p\|x\|_2 \leq \|\Phi x\|_2 \leq (1+\epsilon)\alpha_p\|x\|_2$$

   where $\alpha_1 = k\sqrt{2\pi^{-1}}$ and $\alpha_2 = k$.

2. *The mapping $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^k$ requires,*

   $$O(d \log d + \min\{d\epsilon^{-2} \log n, \epsilon^{p-4} \log^{p+1} n\})$$

   operations.

2.4 Fast Johnson-Lindenstrauss Method 2

2.4.1 Previous Work

The fast JL method 1 made an improvement on the runtime performance and the embedding dimension limit compared to Dimitris Archlioptas's approach through sparsity. This gave an runtime of $O(d \log d)$ for values of $k = O(d^{\frac{1}{3}})$ [3]. The difficulty was that once the upper-bound of k is exceeds beyond the limit, the method fails. More specifically, the modeling of the random projection matrix Φ becomes invalid because the distortion increases. Hence the Johnson-Lindenstrauss theorem no longer holds. Further research by Nir Ailon and Edo Liberty, increased the upper-bound of k for the same runtime of $O(d \log d)$ [8]. The research used the same approach of using a randomized extended Fourier transform but uses concepts from coding theory rather than sparsity. This section covers the fast JL method technique using coding theory.

2.4.2 Error-Correcting Codes

Since the information age, coding theory has played a major role in the digital revolution. As the size of an electronic device decreased, the data communication between these machines has drastically increased [9]. For example, a Radio Common Carrier used by the military in the 1960s for just communication, has now become a smartphone which is the size of a hand but has significantly larger range in terms of functionality. Due to the increase in data demand, it required an error detecting and correcting mechanism on the machines. Hence Error-correcting codes became popular in the communication world.

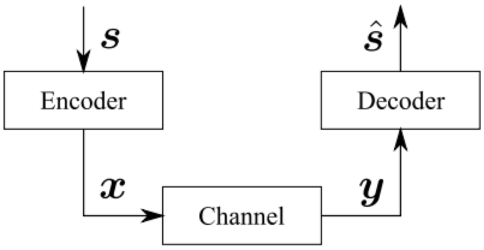


Figure 2: Communication between sender and receiver through a channel [10]

In any device it is important that the input signal (S) and output signal ($\hat{S}$) in Figure 2 are the same. The error-correcting code will handle the errors produced by the channel before sending out the information. The concepts of BCH codes in error-correcting codes are used in this fast JL method. The Bose-Chaudhuri-Hocquenghem (BCH) belong to the cyclic codes class. A matrix is cyclic if for each vector x in the matrix X, the vector x' obtained by shifting the elements of the vector x cyclically to the right by one unit, is also a vector of the matrix X [9].

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \qquad \begin{aligned} a &= \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \\ a' &= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \end{aligned}$$

The above matrix A is a cyclic code because all its vectors (rows) are a right cyclical shift of one vector. The cyclic codes with generator polynomials that "make the minimum distance guaranteed by the BCH bound, large" are classified as BCH codes [9].

2.4.3 Fast JL with Error-Correcting Codes

The random projection for this method is defined as follows [8]:

$$\Phi = B_{k\times d}.D_{d\times d}.\Theta^{(r)} \tag{5}$$

where $\Theta^{(r)} = H.D^{(r)}.H.D^{(r-1)}.....H.D^{(2)}.H.D^{(1)}$. The matrix $D_{d\times d}$ and the matrices $D' = D^{(1)}, D^{(2)}, \ldots, D^{(r)}$ are essentially the $d \times d$ random diagonal matrix from the fast JL method 1 whose diagonal elements are ± 1 with probability $\frac{1}{2}$. Similarly, H is the $d \times d$ extended discrete Fourier transform, Walsh-Hadamard matrix of method 1 and its elements are $H_{i,j} = d^{-\frac{1}{2}}(-1)^{<i-1.j-1>}$ where $< a, b >$ is the dot product between a and b. The matrix $B_{k\times d}$ is the code matrix, with underlying concept from BCH codes, contains deterministic non-trivial choice of copies of H placed side by side [8].

In this method, the input is initially multiplied by the orthogonal set of matrices including H and D, hence the input information is manipulated but there is no loss of information due to isometry. There is an iterative multiplication of the matrices H and D which is represented by $\Theta^{(r)}$. This is because the method uses the l_4-norm [8] rather than the l_2-norm [3] (in method 1) for distances and was able to prove the following lemma for an input x.

Lemma 2 (**l_4 reduction for** $k < d^{\frac{1}{2}-\delta}$ **with probability** $1 - O(e^{-k})$)

$$\|\Theta x\|_4 = O(d^{-\frac{1}{4}}) \tag{6}$$

for $r = \lceil \frac{1}{2\delta} \rceil$.

Using Lemma 2 and the repetitive structural properties of code matrix B, applying $\Theta^{(r)}$ to singletons of B separately, proved that the runtime of the random projector Φ to be $O(d \log k)$. This is because, a property underlying in the code matrix enabled the "decomposition of coordinates" of $\Theta^{(r)}$ which brought down the runtime from $O(d \log d)$ from $\Theta^{(r)}$ to $O(d \log k)$ after applying B. Since B is made up from obtaining a certain section of H and copied it side by side, it has the ability to multiple a single copy with a matrix H in Θ. Thus improving the performance by the use of submatrices [8]. Although the performance is improved, there is a distortion provided due to the decomposition where as there was no loss before applying the matrix B. However the distortion is well within the JL limits for all $k \leq d^{\frac{1}{2}-\delta}$ where δ is an arbitrary small constant.

2.4.4 Method 2 Summary

This method has improved the runtime performance by eliminating the $\log d$ factor to give a new runtime of $O(d \log k)$. In addition, the upper-bound of the embedding dimension is now $d^{\frac{1}{2}-\delta}$ compared to $d^{\frac{1}{3}}$ in method 1. However still this method will produce distortion above the JL limit if k exceeds $d^{\frac{1}{2}-\delta}$. This method uses random variables in Rademacher distribution for the proof of bounding the embedding dimension [8]. The following theorems were introduced for this fast JL method, for proofs refer [8].

Theorem 3 *For any code matrix A of size $k \times d$ for $k < d$, the mapping $x \mapsto Ax$ can be computed in time $O(d \log k)$.*

Theorem 4 *Let $\delta > 0$ be some arbitrarily small constant. For any d, k satisfying $k < d^{\frac{1}{2}-\delta}$, there exists an algorithm constructing a random matrix A of size $k \times d$ satisfying Johnson-Lindenstrauss properties, such that the time to compute $x \mapsto Ax$ for any $x \in \mathbb{R}^d$ is $O(d \log k)$. The constriction uses $O(d)$ random bits and applies to both the Euclidean and the Manhattan cases.*

Following this research, it made clear that whilst improving the runtime performance of the random projection, the bounds on the embedding dimension will reduce due to the strict rules implied by certain principles and techniques. Hence the future work was to increase the upper-bound of the embedding dimension as well as maintaining the performance simultaneously.

2.5 Other Fast Johnson-Lindenstrauss Methods

2.5.1 Johnson-Lindenstrauss Transforms and Restricted Isometry Property

By using sparsity and error-correcting codes, it allowed to perform dimension reduction transforms. However the embedding dimension k was restricted to values below $d^{\frac{1}{2}}$. To perform transforms beyond $d^{\frac{1}{2}}$, the restricted isometry property from compressed sensing must be introduced. The restricted isometry property preserves the l_2-norm of a matrix. The following theorem defines the restricted isometry property [23].

Theorem 5 *A matrix A in $\mathbb{R}_{m\times n}$ is said to satisfy the Restricted Isometry Property (RIP) with parameters (s, δ), if for all $\mathscr{T} \subset [n]$ such that $|\mathscr{T}| \leq s$ and for all $q \in \mathbb{R}^{|\mathscr{T}|}$, it holds that,*

$$(1-\delta)\|q\|_2^2 \leq \|A_{\mathscr{T}} q\|_2^2 \leq (1+\delta)\|q\|_2^2 \tag{7}$$

Theorem 6 *The Restricted Isometry Constant δ_s is defined as the smallest constant δ for which the s-RIP holds,*

$$\delta_s = \inf\{\delta : (1-\delta)\|q\|_2^2 \leq \|A_{\mathscr{T}} q\|_2^2 \leq (1+\delta)\|q\|_2^2 \quad \forall|\mathscr{T}| \leq s, \forall q \in \mathbb{R}^{|\mathscr{T}|}\} \tag{8}$$

By inspection, s represents a certain level of sparsity in the matrix A. This means that certain s rows are chosen from A. With the Equation 7, it tries to preserve the l_2-norm, which is the length. Hence the term isometry because it preserves the lengths. The smallest δ that can be used is known as the restricted isometry constant which is represented by Equation 8. The meaning of this is that when a vector q gets multiplied by a matrix A, the length will change. However if s-RIP submatrix of A is used, the change in the length is significantly small. Because it is bounded by $(1 \pm \delta)$ [23]. If the restricted isometry property is represented in a fast RIP matrix, this will be useful to perform fast recovery of sparse signals.

Using this approach, an RIP construction method was introduced. It was initially used for sparse signal recovery and later was used for dimensional reduction [24]. The method was to have a Fourier transform matrix (Walsh-Hadamard matrix) and then chose k rows of that matrix at random. This resulted in forming a matrix which a $k \times d$ RIP matrix with $\left(s, \delta = \left((s \log^4 d)/k\right)^{\frac{1}{2}}\right)$ [24]. From this stage to make the matrix suitable for random dimensional reduction projections, the $d \times d$ random (± 1) diagonal matrix was added to the end. Similar to fast JL method 1, this has the Fourier matrix and the random diagonal matrix. In addition, it had the same runtime performance of $O(d \log d)$ [25]. However the embedding dimension was now in terms of the number of samples n, giving $k \approx \epsilon^{-4} \log^4 n$ [25]. The following indication represents the construction of a projection matrix for the fast JL based on the restricted isometry property.

$$k\begin{array}{c}\to\\ \to\\ \to\\ \to\end{array}\underbrace{\begin{pmatrix} & & \\ & H & \\ & & \\ & & \end{pmatrix}}_{\text{RIP}(k\times d)}\underbrace{\begin{pmatrix}\pm1&0&\cdots&0\\0&\pm1&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&\pm1\end{pmatrix}}_{\text{Random Diagonal Matrix}(d\times d)}$$

2.5.2 Sparse Dimension Reduction

More recently, Jelani Nelson contributed towards the performance bounds of random dimension reduction projectors [26, 27, 28]. Most of the concepts were based on sparsity. One of the methods he defined was that for a projection matrix of $k \times d$, each column of the matrix has a limit for the number of non-zero elements it can have. This provides the same upper-bound as the sparse JL transform in Method 1. However using this theory on an RIP matrix, resulted in forming the optimal RIP matrix. Here the approach is for having sparse columns rather than rows [27]. The new approach produced great results in improving the bounds of a JL transform, where the lower-bound is now non-restricted. The sparse JL transform was then tested and was able to obtain positive results in areas such as linear algebra, compressed sensing and other statistical analysis applications [28]. However it was concluded than due to the bounds, there are limitations for some applications in reality, and there are always more improvements that can be applied to obtain better results through eliminating or trimming the limitations.

2.6 K-Nearest Neighbors Search

One of the simplest Machine Learning algorithms is the K-Nearest Neighbor Search [16]. When a query is run, the result of the query will be based on the K samples most closest to the query by distance. K-Nearest Neighbor search can be done in 2 ways. One is the classification method where the query point is assigned a class same as the majority class of the k nearest samples. The other method is by regression where the query point is assigned a value which is the average of the k nearest samples.

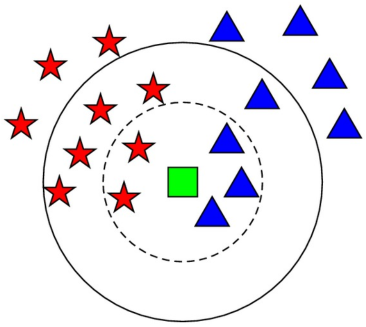


Figure 3: K-Nearest Neighbor Classification Search Example [17]

Figure 3 is a simple K-Nearest Neighbor Classification search query. With $K = 5$, the query point will be classified as blue triangles (▲). However with $K = 10$, the query point will be classified as a red star (★). In regression K-Nearest Neighbors, the query point will be assigned a value rather than a class.

Given a dataset with known target values, the K-Nearest Neighbors method can be used to test a models' accuracy and correctness. Hence this K-Nearest Neighbors Search algorithm is used for the testing and analysis phase of the Fast JL Transform models [11].

2.7 Euclidean Space and Complex Numbers

The data for the project will be in euclidean space of high dimension. This is due to the fact that the dimension reduction considers euclidean distances. To be more specific, euclidean norms will be used. The following equation describes the calculation of norms in n-dimensional space [20] which is one of the many properties of euclidean spaces.

$$l_p\text{-norm of } x \in \mathbb{R}^n, \qquad \|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}} \tag{9}$$

The distance between complex points in a complex space, can be performed using the same method as an l_2 in $\mathbb{R}^2$. However complex number transformations are restricted to l_2-norms [19] and the complex planes have more algebraic structure than an euclidean space of 2 dimensions. Therefore the high dimensional data that is used for this project will be on a real high dimensional euclidean space and projections will be on to a real low dimensional euclidean space.

2.8 Background Summary

The background shows multiple techniques for dimension reduction. It is clear that each method attempts to preserve pairwise distances up to a $1 \pm \epsilon$ factor for distances. This is mainly because the high-dimensional data is being projected onto a low-dimensional subspace. After the introduction to the Johnson-Lindenstrauss tranform, the fast JL methods showed great improvement in performance through means for sparse matrices, error-correcting code matrices, restricted isometry property conversions etc. The common characteristic to all these methods in the randomness. In any instance, there exists a certain degree of randomness in the projection, random diagonal matrices or even the elements of the projector as independent and identically distributed Gaussian entries [28]. In addition, most cases use a Fourier transform to speed up the computation similar to simple Fast Fourier Transforms. Combining these two will densify the input data to make sure that no information is isolated. Hence would result in lower distortions.

From these methods it is important and intriguing to understand how the distortions vary with respect to different embedding dimension sizes, number of data samples, high dimension sizes etc. Formally how this will affect the statistical analysis when applied to the resulting data in the subspace. It is also important to confirm that the bounds for embedding dimension and performance aspects for each method corresponds to the theoretical values from the papers when comparisons are made. This project covers all these aspects to identify the best performing method.

In terms of implementing the fast JL models, MATLAB will be used due to experience, familiarity and also the ability to perform fast matrix manipulation, computation using optimised libraries.

3. Implementation Plan

This section covers the implementation of the Fast Johnson-Lindenstrauss (JL) Transform Models and their statistical analysis used for testing and evaluation. Evaluation of these implementation is covered in section 4. The preliminary timeline and the Gantt chart provides the full project timeline including deliverables and deadlines in later stages of the project. The fast JL methods 1 and 2 are selected to be implemented for this project because of its strong relation towards the modules Topics in Large Dimensional Data Processing and Coding Theory.

The datasets which will be used for the models will be pre-made and each dataset will have some properties, namely:

1. Large dimensional structure ($d >> 40$)
2. Large amount of samples (N)
3. $d > N$
4. Target values assigned to the data (for classification and regression in statistical analysis)

3.1 Modeling Fast JL Method 1

As the literature review for the fast JL method 1, the input matrix goes through 3 stages. Initially the matrix will be multiplied by a random diagonal matrix. The resulting matrix will then be multiplied by the Fourier transform matrix. Finally it is multiplied by the sparse matrix which returns back the input, projected onto a low-dimensional space. These 3 stages are the implementation for this method. The implementation will be programmed using MATLAB.

The diagonal matrix in stage 1 and sparse matrix in stage 3 are independent random matrices which can be implemented using standard MATLAB library functions. Where as in the Walsh-Hadamard matrix is not random and only depends on the element which is being determined. Once all stages are combined in a single MATLAB script, input datasets can be applied to the model for analysis. To ensure accuracy and correctness of the model, the initial dataset will be small in size where the output can be calculated and use it to verify the output. Formally, large datasets will be used for the model.

3.2 Modeling Fast JL Method 2

This model has iterative multiplications compared to modeling fast JL transform 1. As described in the literature review for fast JL method 2, the input matrix will be multiplied by a series of random diagonal matrices and Walsh-Hadamard matrices in an alternating manner ending with a Walsh-Hadamard matrix. The resulting matrix is multiplied by another random diagonal matrix before multiplying with the error-correcting code matrix. The final result is the input projected onto a low-dimensional space. This implementation is also modeled using MATLAB.

Similar to the first model, the diagonal matrix is implemented using standard MATLAB library functions for creating random matrices. In addition, the Walsh-Hadamard matrix is implemented as stated in Equation 4. The initial series of multiplication is only the random diagonal and Walsh-Hadamard matrices, hence MATLAB functions will be used for this iterative process. However it is important to note that the diagonal matrix can change due to its randomness property in each iteration. Finally the error-correcting code matrix (dimensional reduction matrix) is implemented whose rows are deterministic non-trivial choice of rows of the Walsh-Hadamard matrix. Once the model is created, the datasets can be used to test and analyse the model. Similar to model 1, the model will see a relatively small dataset whose output can be easily distinguished and calculated to test correctness and accuracy.

3.3 Applying Statistical Analysis

The algorithm used for analysis will have classification and regression properties. It is easy to determine the correctness and the accuracy of the data in the reduced dimensional space through classification and regression techniques. The following is the implementation of the algorithm used.

3.3.1 K-Nearest Neighbor Search

The K-Nearest Neighbor search algorithm will be implemented on MATLAB. As mentioned in the literature review, the dataset produced by the model is used to run on this algorithm for performance analysis. Different datasets will be used for the algorithm to test on both classification and regression performance. The algorithm will be used to run on both original and reduced dimensional datasets.

3.4 Comparing Models and Results

The models will be compared based on the time taken to produce the reduced dimensional dataset for the same data input. Note that the models will run on standard lab machines in Department of Electrical and Electronic Engineering for fair performance comparison. The objective here is to compare obtained results with the theoretical value as well as identify and improve performance where necessary. Another measurement in comparing models is the distortion. The model will be run for several values of k (embedding dimension) and will record the average distortion in the resulting dataset. These results from all models will be used deduce the best performing model in terms of distortion.

The K-Nearest Neighbor search algorithm will be applied on the original data and its respective reduced dimension data. Then the algorithm will then run with a test query set on both datasets and the results will be recorded (Class if classification algorithm or value if regression algorithm). The results of the test query set is known prior to applying on the data. Hence the performance on accuracy will be evaluated by an error percentage that is used to compare amongst different models and data (original/reduced). The results will be used to determine the performance of the models and possible applications.

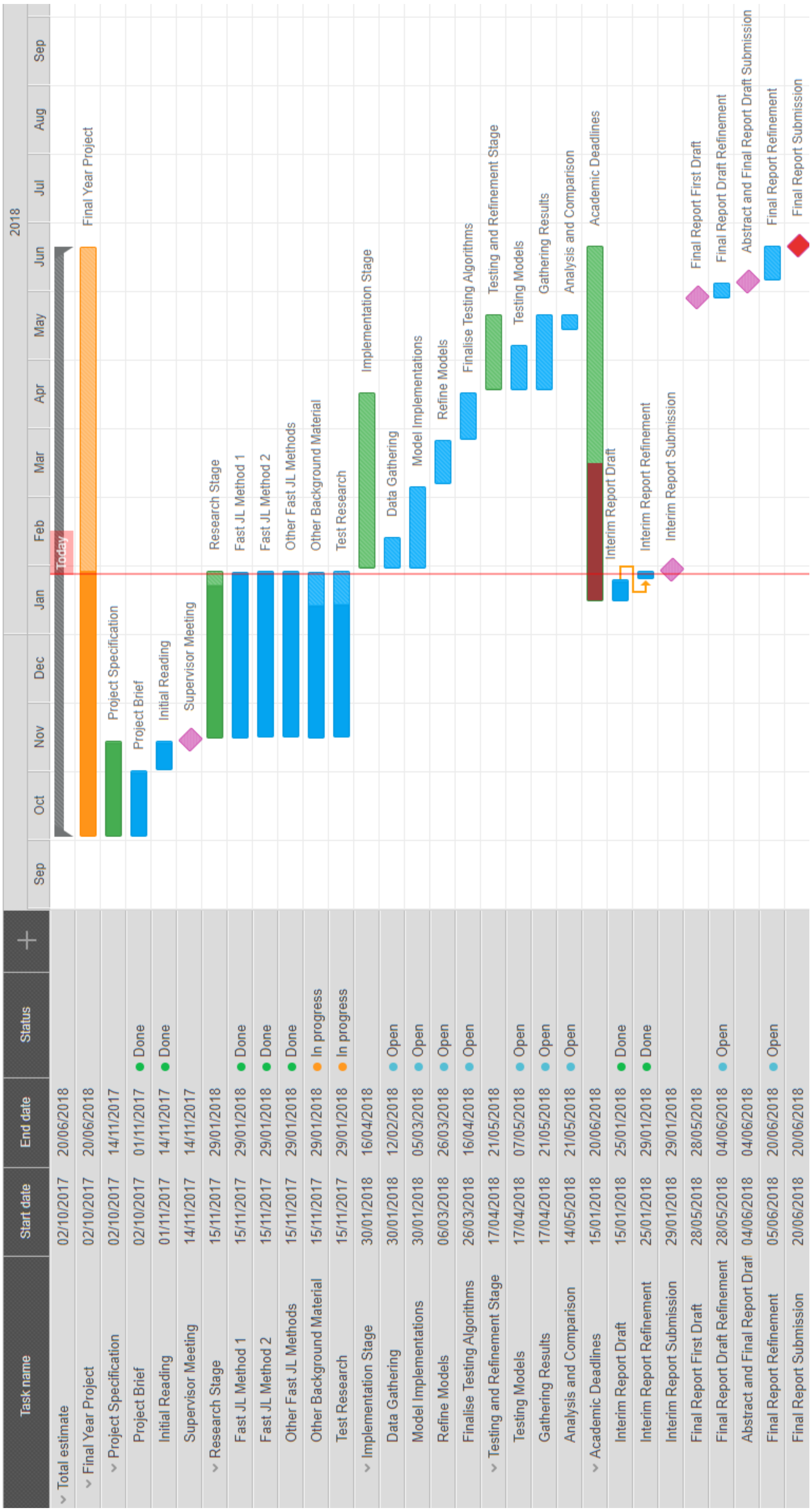
3.5 Preliminary Timeline

The Table 1 represents the timeline for the completion of the project.

Date	Task
12/02/2018	Finalise the datasets that are required for the testing phase. Some data sets must include sampling points with target values.
	No fallbacks on this stage since if data becomes inaccurate or insufficient, there is enough time to gather more datasets at a later stage.
05/03/2018	Finalise MATLAB scripts for the implementation of the Fast JL Transform Models. Start pre-testing the model with small set of data with predetermined results to verify initial correctness and certain measurements can be clearly obtained after a running cycle.
	Only fallbacks here are due to faulty code, outdated MATLAB version or performance (speed) issue with machine. Fallbacks at this stage are unlikely.
26/03/2018	Finalise the results obtained from reducing dimensionality of large datasets. Compare the distortion results for each model.
	The results might not be sufficient to deduce any conclusions. In the case of this situation, run datasets which are proven to have performed certain requirements.
16/04/2018	Finalise the implementation of the K-Nearest Neighbor search algorithm for both classification and regression types.
	A fallback would be if the algorithms cannot be built in MATLAB. If the case arises, use readily available scripts of the algorithm or special MATLAB library for Machine Learning and Statistical Analysis.
21/05/2018	Finalise gathering results from applying the K-Nearest Neighbor search algorithm on datasets. Datasets include the original dataset and reduced dimension datasets. Begin the final stage of performing comparisons in terms of accuracy and runtime. Use all the information gathered to deduce conclusions on a models characteristics, properties and performance.
	The only fallback is the insufficient amount of data to deduce conclusions. In case of the fallback, prepare more datasets to gather more information.
04/06/2018	Abstract and Draft Report Submission.
20/06/2018	Final report submission. Finalise the MATLAB scripts, final report with all the information on results obtained and methods used to present the statements and conclusions.
	There are no fallbacks at this stage.

Table 1: Project Timeline. (*Dates and tasks are subject to change*)

3.6 Gantt Chart



Task name	Start date	End date	Status
› Total estimate	02/10/2017	20/06/2018	
› Final Year Project	02/10/2017	20/06/2018	
› Project Specification	02/10/2017	14/11/2017	
Project Brief	02/10/2017	01/11/2017	● Done
Initial Reading	01/11/2017	14/11/2017	● Done
Supervisor Meeting	14/11/2017	14/11/2017	
› Research Stage	15/11/2017	29/01/2018	
Fast JL Method 1	15/11/2017	29/01/2018	● Done
Fast JL Method 2	15/11/2017	29/01/2018	● Done
Other Fast JL Methods	15/11/2017	29/01/2018	● Done
Other Background Material	15/11/2017	29/01/2018	● In progress
Test Research	15/11/2017	29/01/2018	● In progress
› Implementation Stage	30/01/2018	16/04/2018	
Data Gathering	30/01/2018	12/02/2018	● Open
Model Implementations	30/01/2018	05/03/2018	● Open
Refine Models	06/03/2018	26/03/2018	● Open
Finalise Testing Algorithms	26/03/2018	16/04/2018	● Open
› Testing and Refinement Stage	17/04/2018	21/05/2018	
Testing Models	17/04/2018	07/05/2018	● Open
Gathering Results	17/04/2018	21/05/2018	● Open
Analysis and Comparison	14/05/2018	21/05/2018	● Open
› Academic Deadlines	15/01/2018	20/06/2018	
Interim Report Draft	15/01/2018	25/01/2018	● Done
Interim Report Refinement	25/01/2018	29/01/2018	● Done
Interim Report Submission	29/01/2018	29/01/2018	
Final Report First Draft	28/05/2018	28/05/2018	
Final Report Draft Refinement	28/05/2018	04/06/2018	● Open
Abstract and Final Report Draf	04/06/2018	04/06/2018	
Final Report Refinement	05/06/2018	20/06/2018	● Open
Final Report Submission	20/06/2018	20/06/2018	

Figure 4: Gantt Chart for the Project (*Dates are subject to change*)

4. Evaluation Plan

Following the objectives mentioned under the project specification and their implementation described in the implementation plan, the below methods of evaluation are used to determine the success of the implementation and hence success of the project.

4.1 Fast JL Method 1 Model Evaluation

The evaluation of the implemented model of the Fast JL Transform with sparsity is discussed in this section. The Lemma 1 mentioned in Section 2.3.3 will be the main validation test. As mentioned in [3] the model should hold the 2 events with probability $\frac{2}{3}$. The statement, "if the construction is repeated $O(\log(\frac{1}{\delta}))$ times, the probability of failure drops to δ" [3] will be the evaluation for the functionality of the model. If the statement holds, this model is successful. For a dataset, the distortion will be recorded in comparison to the reduced dimensional dataset for each k value and these results along with the runtime records will be used for the comparison evaluation stage.

4.2 Fast JL Method 2 Model Evaluation

The Fast JL Transform model with error-correcting codes will be evaluated by the models performance. The runtime of this model must be of $O(d \log k)$ and the upper-bound must hold for $k < d^{\frac{1}{2}-\delta}$ [8]. The evaluation of the upper-bound on the reduced dimension will be tested by multiple iterations of running the model for increments of k and analysing the distortion amongst the data points. The distortion in each k must not have abnormal behavior until k reaches the limit of $d^{\frac{1}{2}-\delta}$. No abnormal behavior means that there are no sudden increase in the distortion. Once both aspects are satisfied, implies that the Theorems 3 and 4 are satisfied as well. Hence the model is successful and will progress on to the analysis stage.

4.3 Statistical Analysis Evaluation

Initially the script for this algorithm must be validated prior applying to the original datasets or the reduced dimensional datasets. This evaluation will be using a test data set with known outputs and perform an N-fold cross validation and observe its standard deviation of accuracy. The standard deviation should be well below 10% [18]. Following successful evaluation, the algorithm will be used with the obtained datasets.

The original data and the reduced dimensional data from the model will run on the algorithm also as N-fold cross validation technique. Regardless whether it is classification or a regression type, the standard deviation will be obtained from the results of all the datasets. An evaluation at this point is successfully obtaining the standard deviation or average error graphs from the datasets. Formally successful evaluation will progress on to the comparison stage.

4.4 Models and Results Comparison Evaluation

For model comparison, the distortion results (graphical) from the models off the same dataset will be compared along with their respective runtime. Following analysis of the distortion and their runtimes, the model producing the least distortion will be deduced at this stage. In addition, the results will be compared with the performance proofs in [3] and [8].

Formally, we analyse the results from the cross validation of the K-Nearest Neighbors search algorithm. The graphical results of the standard deviation behavior and the average error behavior of each dataset will be used for analysis. The analysis will result in identifying the most accurate model which will be deduced by the model that produces the least standard deviation and least average error in the reduced dimensional space compared to its respective original dataset, in all iterations.

These evaluation of models and results from statistical analysis, the best performing model can be deduced by performance in terms of accuracy and speed. Following the error calculations and performance, the applications which these techniques become useful can be determined. However, it will also help to identify the limitations of these techniques in certain industries.

5. Ethical, Legal and Safety Plan

This project do not conduct any questionnaires or any form of user experience testing. The participants to this project limited to myself. Hence this project will not raise any form of ethical issues.

The research for this project is purely based on past work, publications and online resources. No patents are involved in this project. Hence this project will not raise any form of legal issues.

No lab work is required for this project. The models and all computation will be done in MATLAB on a standard machine. Hence this project will not raise any safely issues.

References

[1] Johnson, W. and Lindenstrauss, J. (1984). *Extensions of Lipschitz mappings into a Hilbert space.* Conference on Modern Analysis and Probability, [Online] pp.189-206. Available at: https://www.researchgate.net/publication/235008656_Extensions_of_Lipschitz_maps_into_a_Hilbert_space [Accessed 19 Jan. 2018].

[2] Dasgupta, S. and Gupta, A. (2002). *An elementary proof of a theorem of Johnson and Lindenstrauss.* Random Structures and Algorithms, [Online] 22(1), pp.60-65. Available at: http://onlinelibrary.wiley.com/doi/10.1002/rsa.10073/epdf [Accessed 24 Jan. 2018].

[3] Ailon, N. and Chazelle, B. (2010). *Faster dimension reduction.* Communications of the ACM, [Online] 53(2), pp.97-104. Available at: https://www.cs.princeton.edu/~chazelle/pubs/fasterdim-ac10.pdf [Accessed 19 Jan. 2018]

[4] Bingham, E. and Mannila, H. (2001). *Random projection in dimensionality reduction: Applications to image and text data.* Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01. [Online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.8124&rep=rep1&type=pdf [Accessed 22 Jan. 2018].

[5] Achlioptas, D. (2001). *Database-friendly random projections.* Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '01. [Online] Available at: https://dl.acm.org/citation.cfm?doid=375551.375608 [Accessed 23 Jan. 2018].

[6] Tsitsvero, M., Barbarossa, S. and Di Lorenzo, P. (2016). *Signals on Graphs: Uncertainty Principle and Sampling.* IEEE Transactions on Signal Processing, [Online] 64(18), pp.4845-4860. Available at: https://arxiv.org/pdf/1507.08822.pdf [Accessed 23 Jan. 2018].

[7] En.wikipedia.org. (2018). *Uncertainty principle.* [Online] Available at: https://en.wikipedia.org/wiki/Uncertainty_principle [Accessed 23 Jan. 2018].

[8] Ailon, N. and Liberty, E. (2008). *Fast Dimension Reduction Using Rademacher Series onDual BCH Codes.* Discrete & Computational Geometry, [Online] 42(4), pp.615-630. Available at: https://www.math.ias.edu/csdm/files/06-07/nailon_fast_dimension_reduction_using_rademacher.pdf [Accessed 20 Nov. 2017].

[9] Peterson, W. and Weldon, E. (1972). *Errorcorrecting codes.* 2.ed. Cambridge,Mass.: MIT press, pp.206-308.

[10] Dai, W. (2017). *Error Correcting Codes: Fundamentals.*

[11] Beyer, K., Goldstein, J., Ramakrishnan, R. and Shaft, U. (1999). *When Is "Nearest Neighbor" Meaningful?.* Database Theory — ICDT'99, [Online] 1540, pp.217-235. Available at: https://link.springer.com/content/pdf/10.1007\%2F3-540-49257-7_15.pdf [Accessed 25 Jan. 2018].

[12] En.wikipedia.org. (2018). *Curse of dimensionality.* [Online] Available at: https://en.wikipedia.org/wiki/Curse_of_dimensionality [Accessed 22 Jan. 2018].

[13] En.wikipedia.org. (2018). *Big Data.* [Online] Available at: https://en.wikipedia.org/wiki/Big_data [Accessed 15 Jan. 2018]

[14] Stephens, Z., Lee, S., Faghri, F., Campbell, R., Zhai, C., Efron, M., Iyer, R., Schatz, M., Sinha, S. and Robinson, G. (2015). *Big Data: Astronomical or Genomical?.* PLOS Biology, [Online] 13(7), p.e1002195. Available at: http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1002195 [Accessed 22 Jan. 2018].

[15] Mamuscia, C. (2018). *Big Data and Video Surveillance: A Perfect Match.* [Online] Blog.verint.com. Available at: http://blog.verint.com/security-intelligence/big-data-and-video-surveillance-a-perfect [Accessed 22 Jan. 2018].

[16] En.wikipedia.org. (2018). *K-nearest neighbors algorithm.* [Online] Available at: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm [Accessed 25 Jan. 2018].

[17] Wu, J., Cui, Z., Sheng, V., Shi, Y. and Zhao, P. (2014). *Mixed Pattern Matching-Based Traffic Abnormal Behavior Recognition.* [Image] Available at: https://www.researchgate.net/publication/260612049_Mixed_Pattern_Matching-Based_Traffic_Abnormal_Behavior_Recognition [Accessed 25 Jan. 2018].

[18] Kohavi, R. (1995). *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection.* Proceedings of the Fourteenth International Joint Conference on Artificial

Intelligence, [Online] 2(12), pp.1137–1143. Available at: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=53F858FEC1A76E267DC6AA05C2C28E06?doi=10.1.1.48.529&rep=rep1&type=pdf [Accessed 26 Jan. 2018].

[19] Berger, M. (1987). *Euclidean vector spaces.* In: M. Berger, ed., Geometry I, 1st ed. [Online] Universitext: Springer-Verlag Berlin Heidelberg, pp.151-152. Available at: https://doi.org/10.1007/978-3-540-93815-6_9 [Accessed 26 Jan. 2018].

[20] En.wikipedia.org. (2018). *Norm (mathematics).* [Online] Available at: https://en.wikipedia.org/wiki/Norm_(mathematics) [Accessed 26 Jan. 2018].

[21] Oymak, S. and Tropp, J. (2015). *Universality laws for randomized dimension reduction, with applications.* 2010 Mathematics Subject Classification, [Online] 3. Available at: https://arxiv.org/pdf/1511.09433.pdf [Accessed 27 Jan. 2018].

[22] Gao, P. and Ganguli, S. (2015). *On simplicity and complexity in the brave new world of large-scale neuroscience.* Current Opinion in Neurobiology, [Online] 32, pp.148-155. Available at: https://www.sciencedirect.com/science/article/pii/S0959438815000768 [Accessed 27 Jan. 2018].

[23] Dai, W. (2017). *Greedy Algorithms from Topics in Large Dimensional Data Processing.*

[24] Rudelson, M. and Vershynin, R. (2006). *Sparse reconstruction by convex relaxation: Fourier and Gaussian measurements.* 2006 40th Annual Conference on Information Sciences and Systems. [Online] Available at: http://ieeexplore.ieee.org.iclibezp1.cc.ic.ac.uk/document/4067804/?reload=true [Accessed 28 Jan. 2018].

[25] Ailon, N. and Liberty, E. (2013). *An Almost Optimal Unrestricted Fast Johnson-Lindenstrauss Transform.* ACM Transactions on Algorithms, [Online] 9(3), pp.1-12. Available at: https://www.researchgate.net/publication/45920633_An_Almost_Optimal_Unrestricted_Fast_Johnson-Lindenstrauss_Transform [Accessed 28 Jan. 2018].

[26] Kane, D. and Nelson, J. (2014). *Sparser Johnson-Lindenstrauss Transforms.* Journal of the ACM, [Online] 61(1), pp.1-23. Available at: https://arxiv.org/pdf/1012.1577.pdf [Accessed 19 Jan. 2018].

[27] Nelson, J. and NguyÅn, H. (2013). *Sparsity lower bounds for dimensionality reducing maps.* Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13. [Online] Available at: https://arxiv.org/pdf/1211.0995.pdf [Accessed 19 Jan. 2018].

[28] Bourgain, J., Dirksen, S. and Nelson, J. (2015). *Toward a unified theory of sparse dimensionality reduction in Euclidean space.* Geometric and Functional Analysis, [Online] 25(4), pp.1009-1088. Available at: https://arxiv.org/pdf/1311.2542.pdf [Accessed 19 Jan. 2018].

[29] GanttPRO. (2018). *Create a Project Plan with Free Online Gantt Chart Software | GanttPRO.* [Online] Available at:https://app.ganttpro.com [Accessed 29 Jan. 2018].

[30] En.wikipedia.org. (2018). *MATLAB.* [Online] Available at: https://en.wikipedia.org/wiki/MATLAB [Accessed 29 Jan. 2018].