**Introduction**

Big data are data sets which are so large or complex that traditional data processing or learning applications are inadequate. Some challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating, etc. Big data causes computational difficulties and intrinsic statistical difficulties due to the data set being of large dimensions. This can cause overfitting, false structures, data isolation, etc.

As data grows day by day, exploring different ways of dimension reduction is essential.

**Initialising**

* We attempt Euclidean dimension reduction in different methods to find an optimal solution
* We have a Data matrix **X** of **N** points in **Rn**
* We want to represent the **Rn** in a k dimensional subspace, where k << n, where k is the embedding dimension
* Do it through randomised linear mapping
* Also maintain pairwise distances i.e. Low distortion

**JL Transform**

: Projection onto random k-dimensional subspace

; by linearity

That is, for any pair of points in , the norm (distance) between and in the reduced space, is at most times original distance, and at least times the original distance.

There is an term because there is a loss.

**Matrix friendly version of JL**

Project onto random matrix using iid Gaussian, iid +/- 1

has k rows. Hence,

JL matrix friendly versions all require time , memory , randomness

So, for example, this overwhelms Nearest Neighbour Searching algorithm,

**Fast JL Transform** (FJLT) [Ailon, Chazelle 2010]

* Use of Walsh-Hadamard matrix, a discrete Fourier Transform.
  + Very simple to compute
  + Only steps
  + , where is the dot product of the -bit vectors, expressed in binary

: Sparse JL. Randomly chosen matrix with independent mixture of 0 with an unbiased normal distribution of variance 1/ , where

.

In other words, with probability and with probability .

: normalised Walsh-Hadamard matrix

: Diagonal matrix where each is drawn independently from with probability .

follows the mapping of any point and can be computed in time , where is the number of non-zero entries in P. Sparsity is approximately non-zeros.

Time =

Randomness =

This beats JL, bound for

Tail bound proof uses Hoeffding’s inequality, Chernoff bound

**FJLT** [A, Liberty 2007]

For , is small

; Deterministic non-trivial. Rows of are a subset of .

Runtime =

Proof uses Talagrand’s bound for vector valued Rademacher Random Vectors.

Looks at norms. (Assume ), then

After steps, ,

Then final step:

If is going above , then we require Restricted Isometry Property i.e. compressed sensing

By theory,

Matrix , has RIP with parameters if all -sparse , (there’s low distortion)

Recovery of sparse signals can be done through few linear observations.

**JL & RIP: The simple way** [Baraniuk et al 2008]

If JL with exist, then you can find a RIP with Which are optimal parameters.

Proof: uses -net arguments

**RIP Construction** [Rudelson Vershynin 2006] version

Obtain a RIP matrix by choosing k random rows of

The RIP has

**FJLT** [Ailon, Liberty 2011]

Almost JL: Embedding dimension

Choose k random rows from and then added the random diagonal

The rows of provides the RIP matrix

Run time:

Proof uses Adaptation of [Rudelson Vershynin 2006]

But JL here is not optimal since is to the power 4 and is to the power

**The Converse Theorem** [Krahmer Ward 2011]

Now JL to RIP conversion is easy. But lets try the more general way, from RIP to JL.

Here they tried to prove that every RIP matrix which just knows how to preserve sparse vectors, can be converted into a JL matrix which it can preserve any finite set of vectors

You have the RIP then, i.e. you precondition the RIP matrix, you get a JL matrix. Also requires, . Of course from RIP is the from JL.

JL not optimal because the RIP requires the extra

**Combining the Converse Theorem and RIP Construction** [KW 11] [RV 06]

You get a JL with with runtime of .

Proof: You take s-sparse , and compute

Now you use Scalar Rademacher chaos of degree 2.

Given, ,

You have

[Hanson Wright], Bound norms of using RIP property.

**Near Equivalence of JL, RIP**

* JL to RIP is efficient and if optimal JL exists (that is randomised), with high probability, you will get an optimal RIP matrix
* RIP to JL is preconditioned with random diagonal and factor is wasted in embedding dimension

Refer to idea previously.

Try to apply the repeated transformation for the less ambitious purpose of RIP.

[A Rauhut 2013]

Result 1:

Arbitrary deterministic , i.e. take any deterministic section of k rows (*together*) of the resulting matrix.

This has optimal RIP parameters for any

Comparing with first FJLT, more complex selection matrix there ( non-zeros).

Result 2:

For a best RIP distortion parameter, let

In words: Infimum over all possible distortion parameters such that has RIP with that parameter. i.e. Maximal distortion of an s-sparse vector of this matrix .

Let your next attempt, to be,

Then, , where constant .

Bootstrap: uses previous RIP to get a better RIP.

Conclusion: Start with anything reasonable (ex: rows of norm 1 and orthogonal to each other), (left to right)

Optimal RIP: . Much easier than RIP obtained in because does not have to be a code matrix. is anything having non-identical rows.

The proof just uses

* s-sparse , unit norm.
* Use of vector valued Rademacher chaos of degree 2
* Sum over of product of two random signs and vector
* This gives rise to 3-dimensional tensor i.e. matrix of vectors
* Proof used Talagrand tail bounds
* Bounding in
* Definition of almost (look like) an s-sparse vector

Limitations

* Repeated method might fail for embedding dimension
* Is the best that can be achieved?

Next steps

* Nelson et. Al improve on [Krashmer Ward]
  + Reduction of an extra factor from FJLT in regime.
  + [Kane, Nelson] on Sparse JL

Clarification

* Binary embeddings
* Hash projections