Imperial College London

25th June 2018

Dimension Reduction for Big Data

Devin Nanayakkara

Overview

- Introduction
  - Motivation
- Goal
- Implementation
- Results
- Conclusion

Applications

- Analytics
- Infrastructure
- Business Intelligence
- Other Technologies

Curse of Dimensionality

1. Data isolation

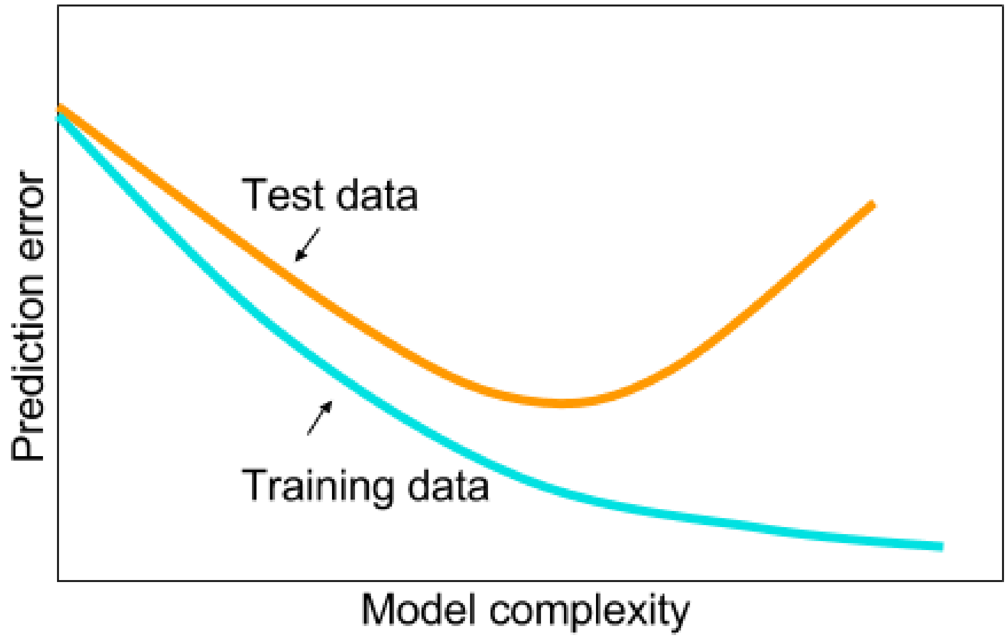
Curse of Dimensionality

1. Data isolation
   - Number of training samples required for a K-Nearest Neighbour search algorithm

Dimension size	10	15	20	30	50	100	150
Number of samples required	1	3	39	$45{,}378$	5.76×10^{12}	4.22×10^{39}	1.28×10^{72}

Curse of Dimensionality

1. Data isolation
2. Overfitting

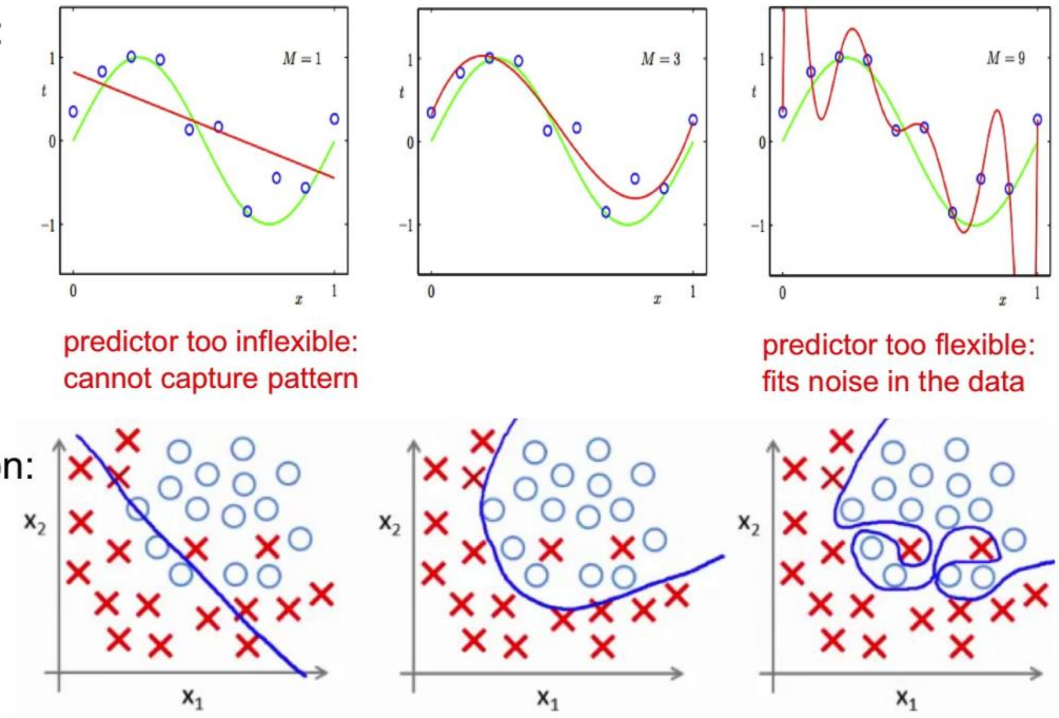


Curse of Dimensionality

1. Data isolation
2. Overfitting
3. False Structure

Regression:

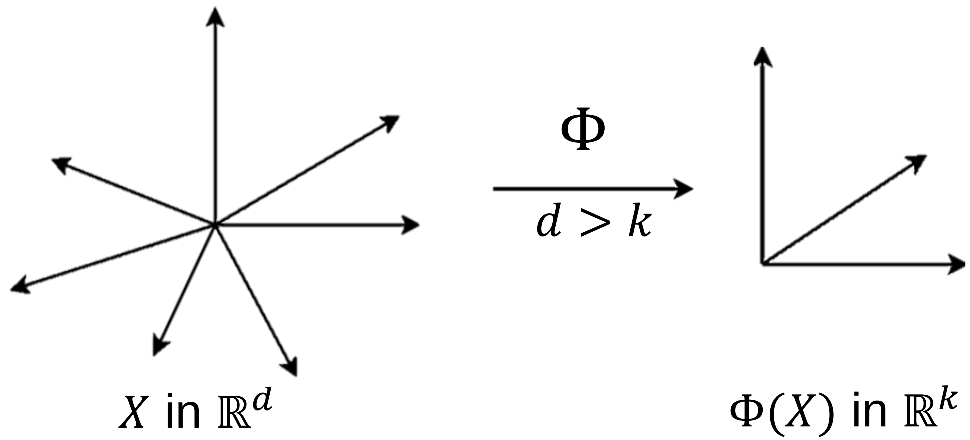
Classification:



Curse of Dimensionality

1. Data isolation
2. Overfitting
3. False Structure
4. Computation difficulty
   – Time
   – Complexity

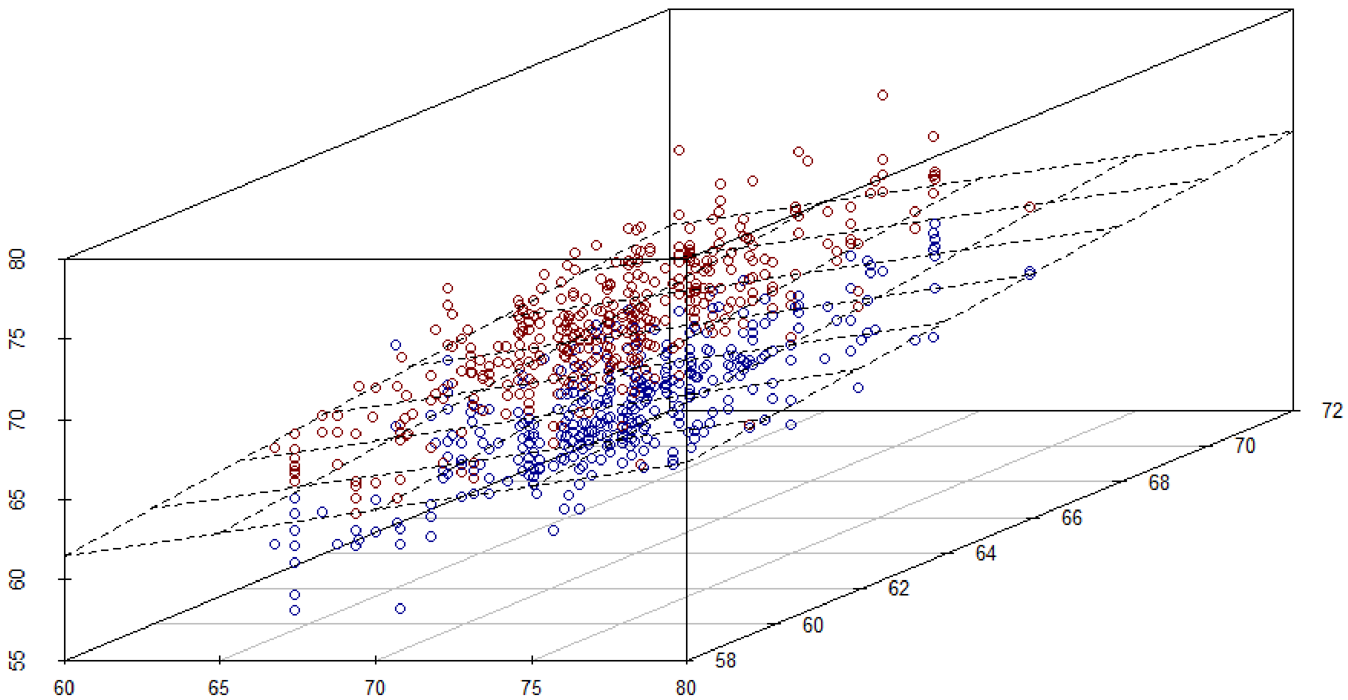
Random Projection (1)



- Assist applications whose data is represented geometrically in high dimensional vector spaces

X – Data matrix
d – Euclidean space dimension
k – Reduced space dimension

Random Projection (2)



Johnson-Lindenstrauss Transform

- Fundemental method for dimension reduction
- Theorem:

For any $(x, y) \in X$,

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

- Projecting any point in the data set (X) on to a random low dimensional subspace should, up to a distortion of $1 \pm \epsilon$, preserve pairwise distances
- Classic JL Transform (FJLT):
  - Runtime: $O(kd)$

k: reduced dimension; d: original dimension; n: sample size

Goal

- Implement structured random matrices to speed up the JL transform
  - Fast JL Transforms (FJLT)

Design Methods

1. FJLT projection method 1
   - Sparse matrices
2. FJLT projection method 2
   - Methods from coding theory

FJLT Method 1

$$\Phi = S \,.\, H \,.\, D$$

- $(k \times d)$ Sparse matrix.
  - $S_{ij} \sim N(0, {}^{1}/_{q})$, with probability q
  - $S_{ij} = 0$, with probability $(1 - q)$
  - $q = \min\left\{\frac{\log^2 n}{d}, 1\right\}$

k: reduced dimension; d: original dimension; n: sample size

FJLT Method 1

$$\Phi = S \,.\, H \,.\, D$$

- $(k \times d)$ Sparse matrix
  - S_{ij} non zero with probability q
  - $q = \min\left\{\frac{\log^2 n}{d}, 1\right\}$
- $(d \times d)$ Walsh-Hadamard matrix
  - Fourier transform matrix
  - Simple to compute; runtime of $O(d \log d)$
  - $H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
  - $H_{2^{2n}} = \begin{pmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{pmatrix}$

k: reduced dimension; d: original dimension; n: sample size

FJLT Method 1

$$\Phi = S\,.\,H\,.\,D$$

- $(k \times d)$ Sparse matrix
  - S_{ij} non zero with probability q
  - $q = \min\left\{\frac{\log^2 n}{d}, 1\right\}$
- $(d \times d)$ Walsh-Hadamard matrix
  - Fourier transform matrix
- $(d \times d)$ Random diagonal matrix
  - $D_{ii} = \{-1,1\}$ with probability $1/2$

$$\begin{pmatrix} \pm 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \pm 1 \end{pmatrix}$$

k: reduced dimension; d: original dimension; n: sample size

FJLT Method 1

$$\Phi = S \,.\, H \,.\, D$$

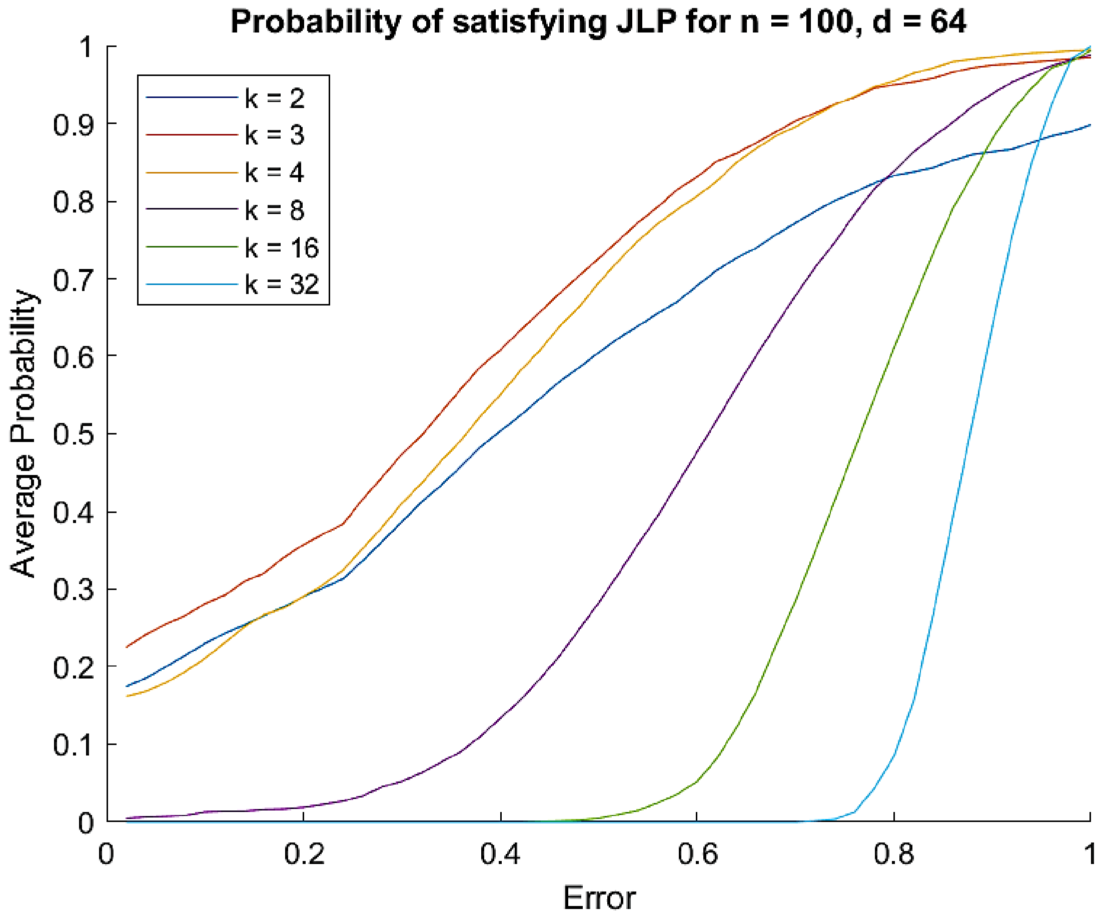
- $(k \times d)$ Sparse matrix
  - S_{ij} non zero with probability q
  - $q = \min\left\{\frac{\log^2 n}{d}, 1\right\}$
- $(d \times d)$ Walsh-Hadamard matrix
  - Fourier transform matrix
- $(d \times d)$ Random diagonal matrix
  - $D_{ii} = \{-1,1\}$ with probability $1/2$

- For $k \leq d^{1/3}$, the runtime is $O(d \log d)$

k: reduced dimension; d: original dimension; n: sample size

Results

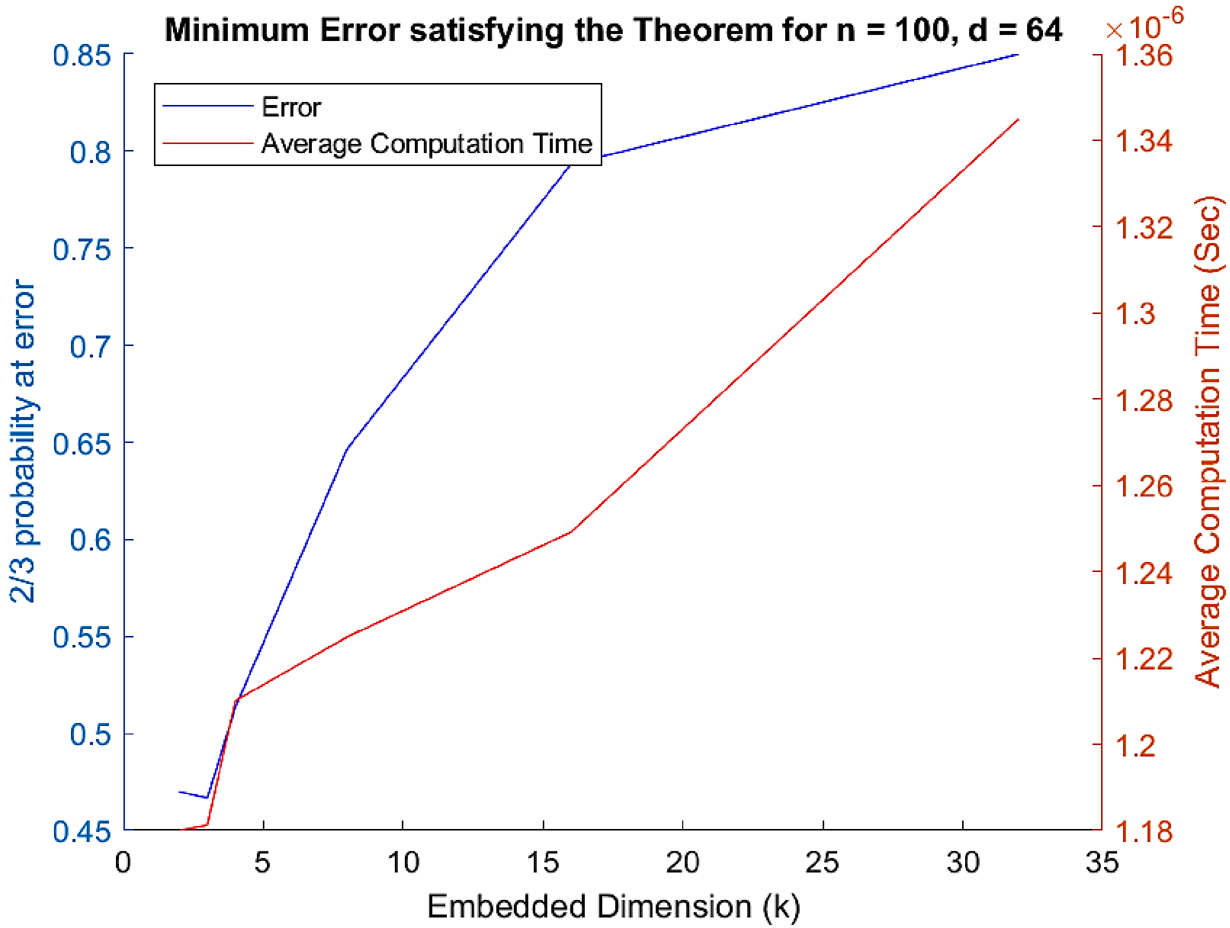
- FJLT 1
  - Test 1
  - $k \leq 4$



k: reduced dimension; d: original dimension; n: sample size

Results

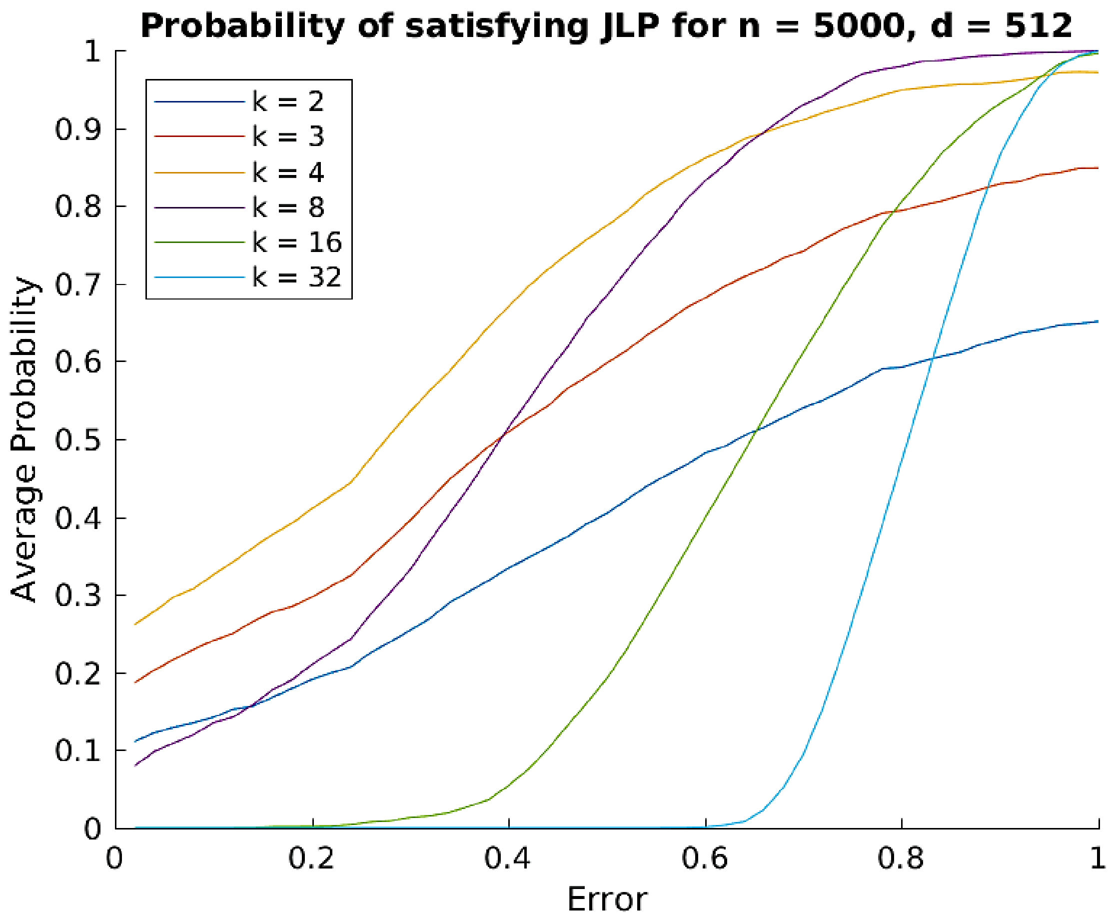
- FJLT 1
  - Test 1
  - $k \leq 4$



k: reduced dimension; d: original dimension; n: sample size

Results

- FJLT 1
  - Test 2
  - $k \leq 8$



k: reduced dimension; d: original dimension; n: sample size

Design Methods

1. FJLT projection method 1
   - Sparse matrices
   - Computation time: $O(d \log d)$
   - Reduced dimension: $k \leq d^{1/3}$
2. FJLT projection method 2
   - Methods from coding theory

k: reduced dimension; d: original dimension; n: sample size

FJLT Method 2

$$A = B \,.\, D \,.\, \Phi_d^r$$

- $(k \times d)$ code matrix
  - Matrix containing unit norm vector columns
- $(d \times d)$ Random diagonal matrix
  - $D_{ii} = \{-1,1\}$ with probability $1/2$
- $(d \times d)$ matrix
  - Combination of H_d and D_d

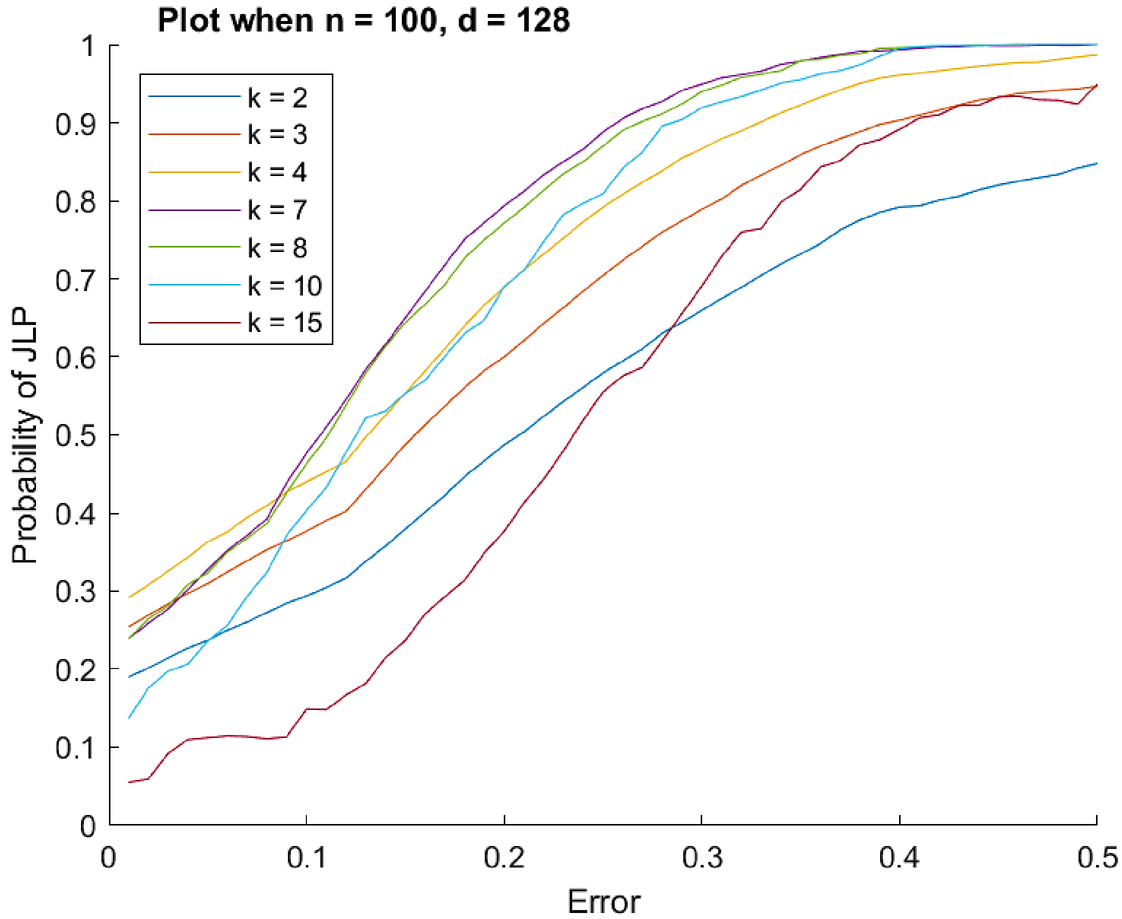
$$\Phi_d^{(r)} = HD^{(r)}HD^{(r-1)} \ldots HD^{(1)}$$

- For $k < d^{1/2}$, the runtime is $O(d \log k)$

k: reduced dimension; d: original dimension; n: sample size; H: Hadamard matrix

Results

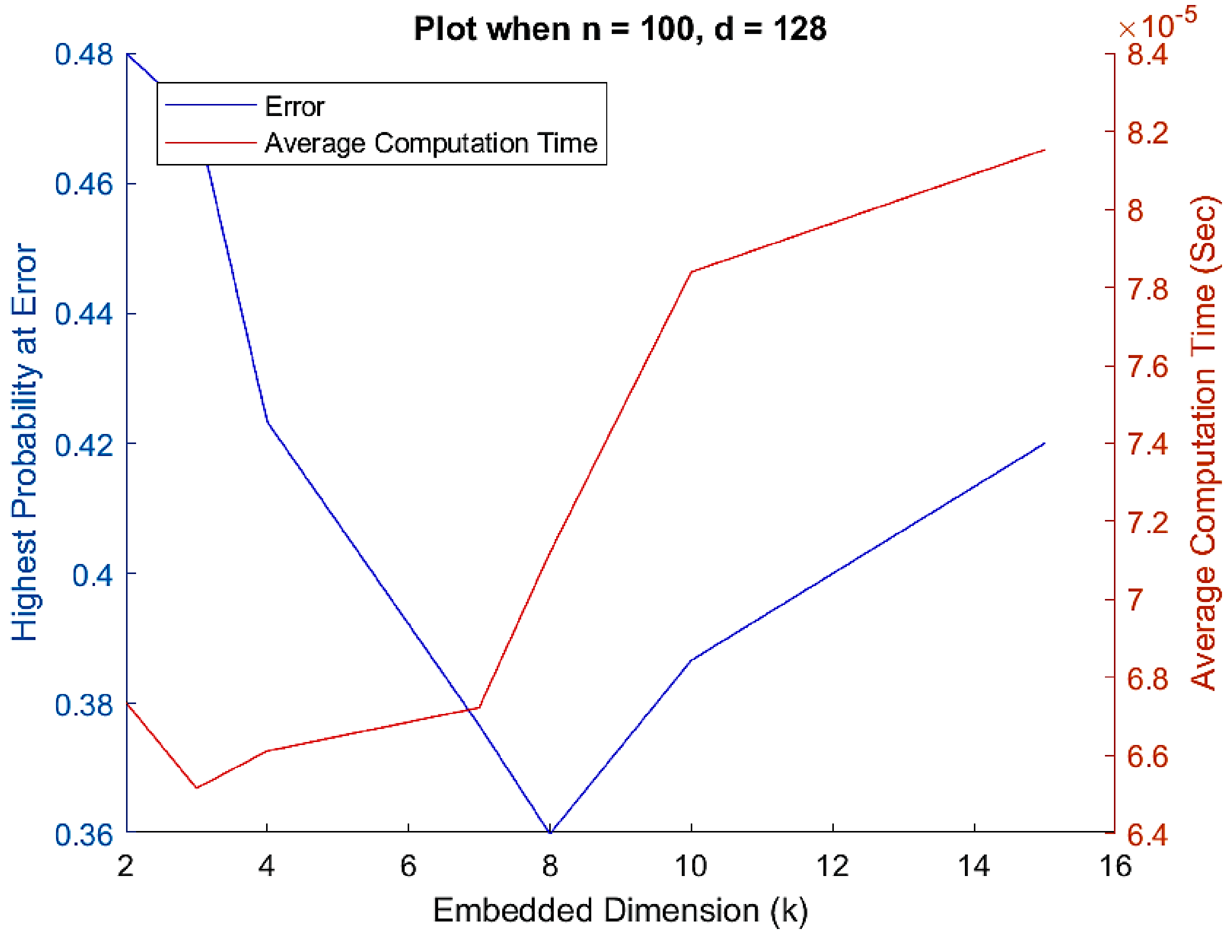
- FJLT 2
  - Test 1
  - $k < 12$



k: reduced dimension; d: original dimension; n: sample size

Results

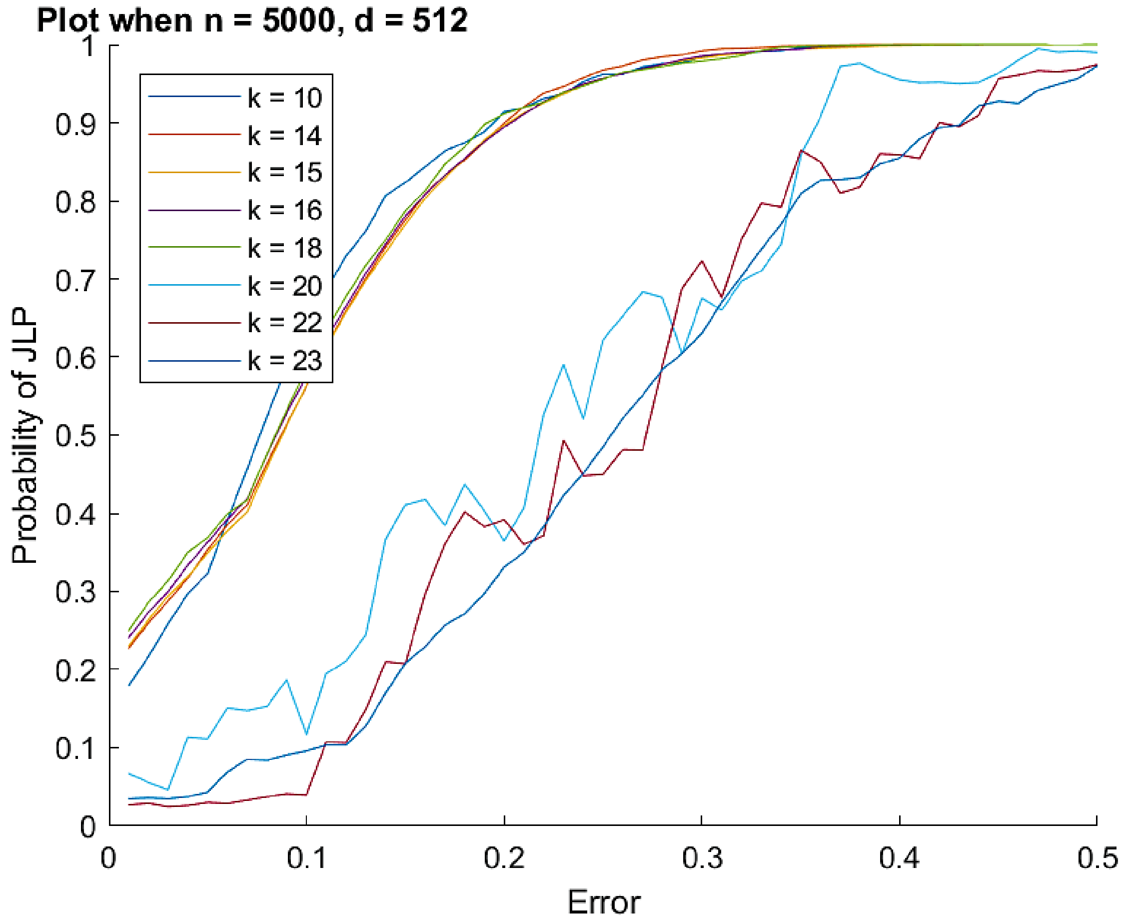
- FJLT 2
  - Test 1
  - $k < 12$



k: reduced dimension; d: original dimension; n: sample size

Results

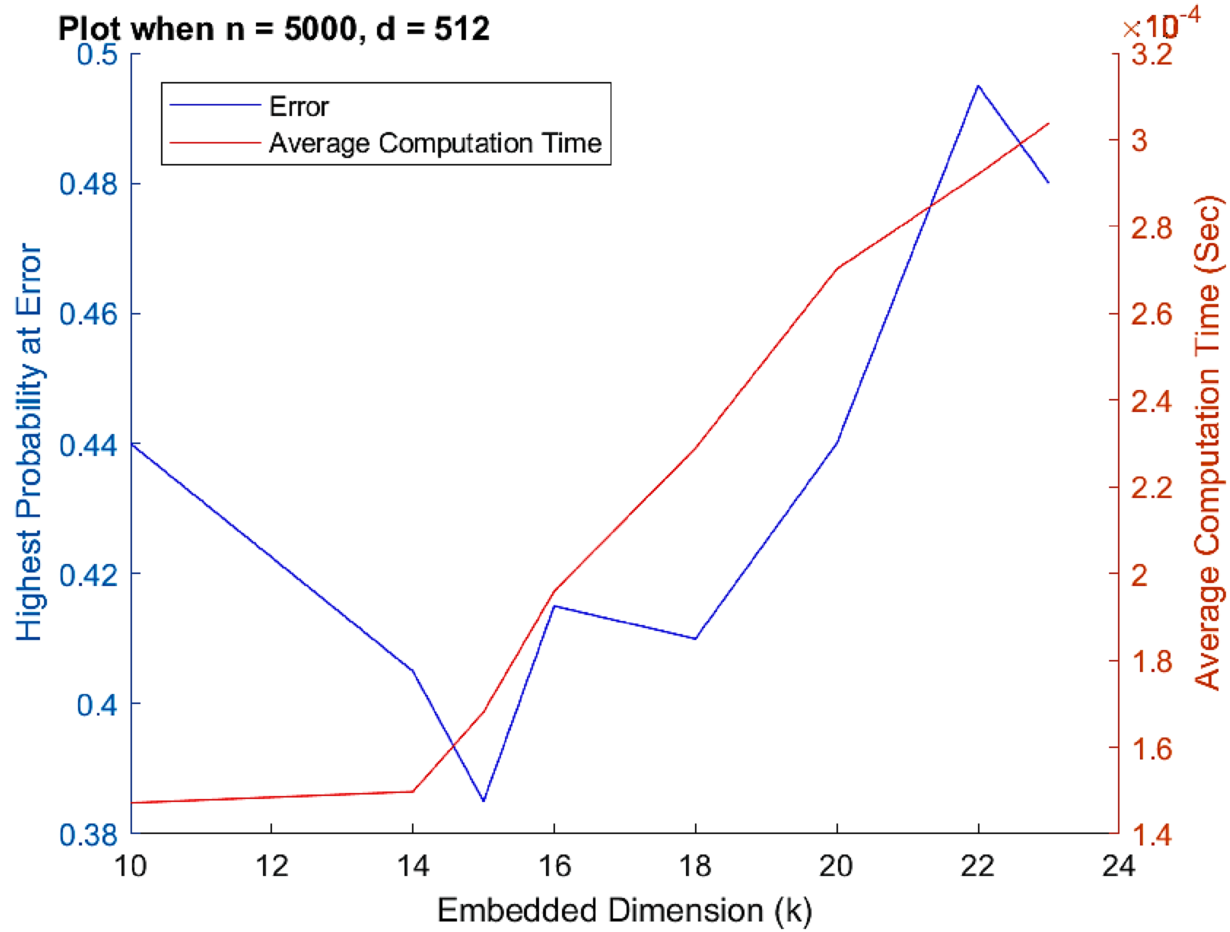
- FJLT 2
  - Test 2
  - $k < 23$



k: reduced dimension; d: original dimension; n: sample size

Results

- FJLT 2
  - Test 2
  - $k < 23$



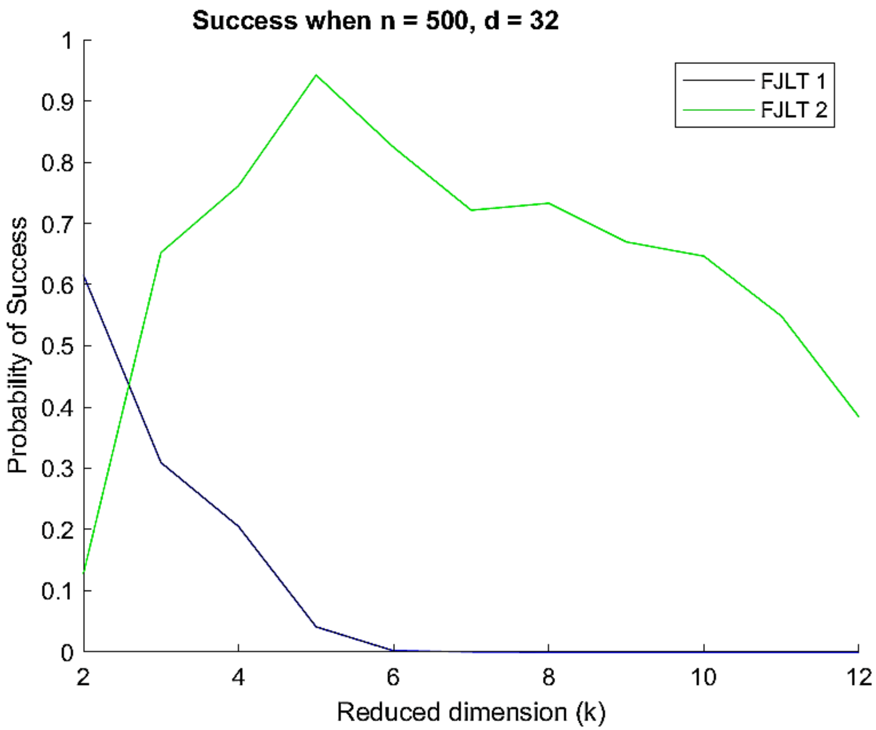
k: reduced dimension; d: original dimension; n: sample size

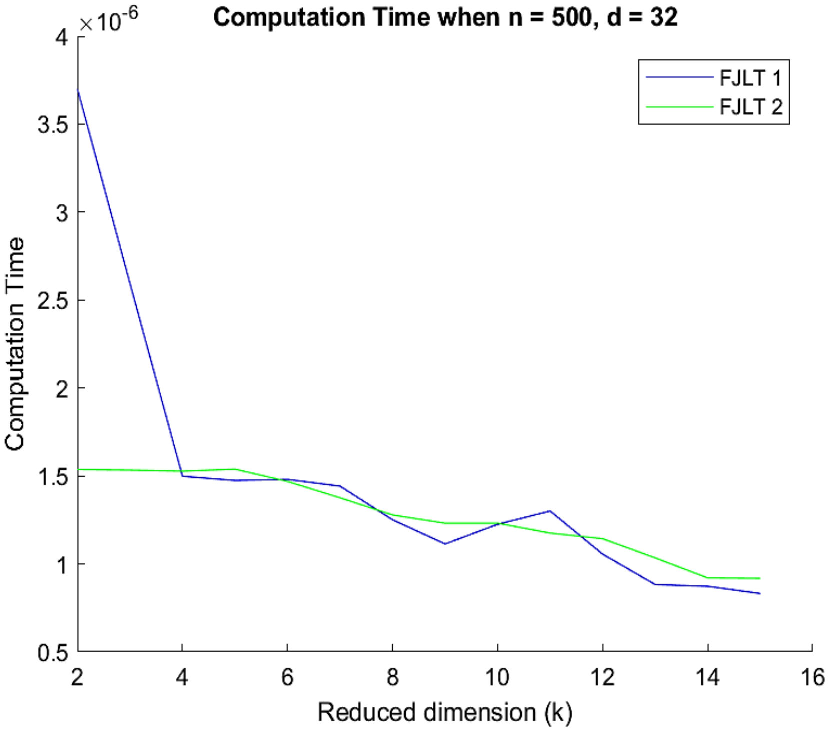
Summary

1. FJLT projection method 1
   - Sparse matrices
   - Computation time: $O(d\log d)$
   - Reduced dimension: $k \leq d^{1/3}$
2. FJLT projection method 2
   - Methods from coding theory
   - Computation time: $O(d\log k)$
   - Reduced dimension: $k < d^{1/2}$

k: reduced dimension; d: original dimension; n: sample size

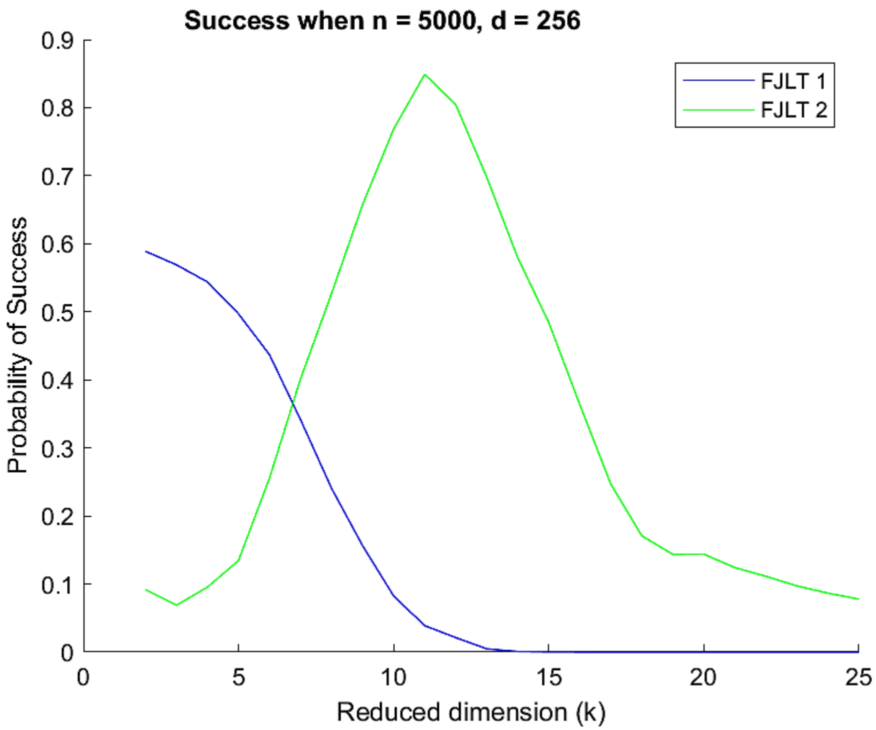
Comparison (1)

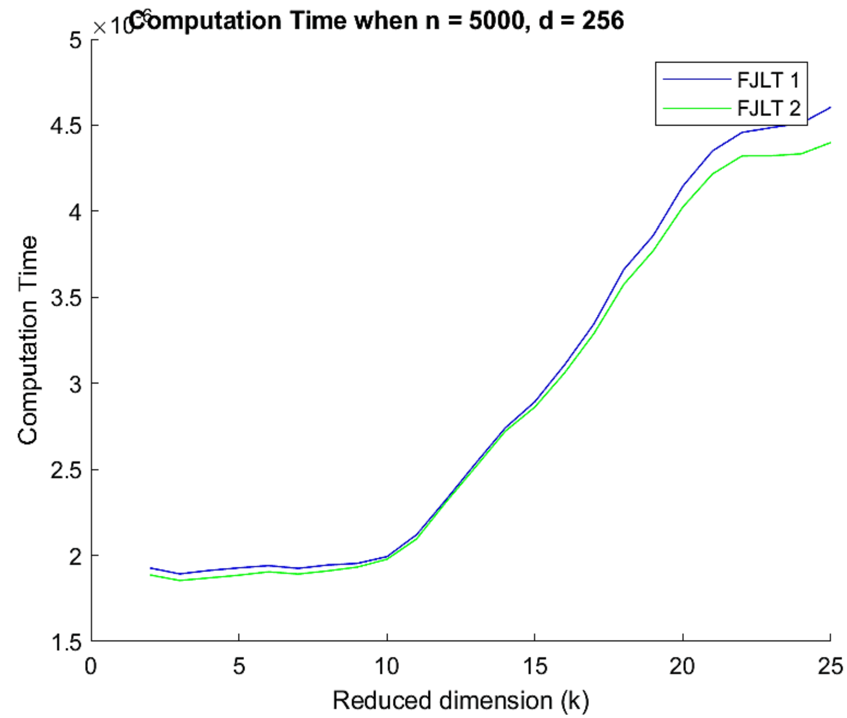




k: reduced dimension; d: original dimension; n: sample size

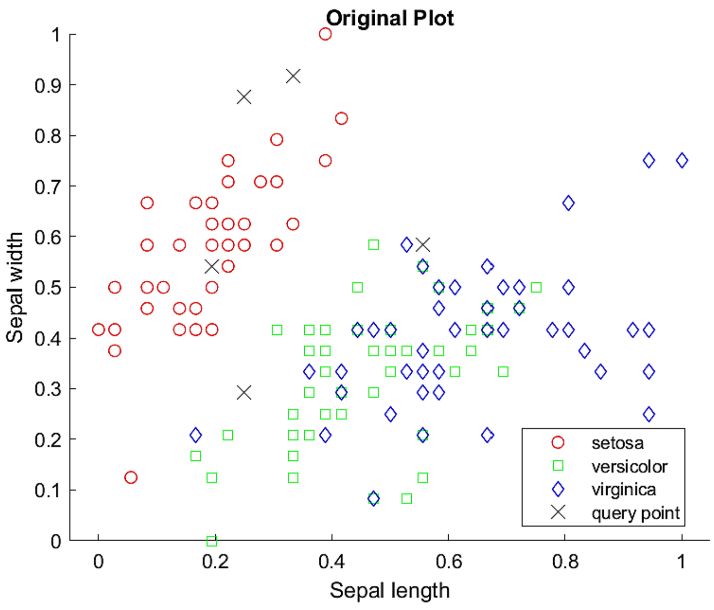
Comparison (2)

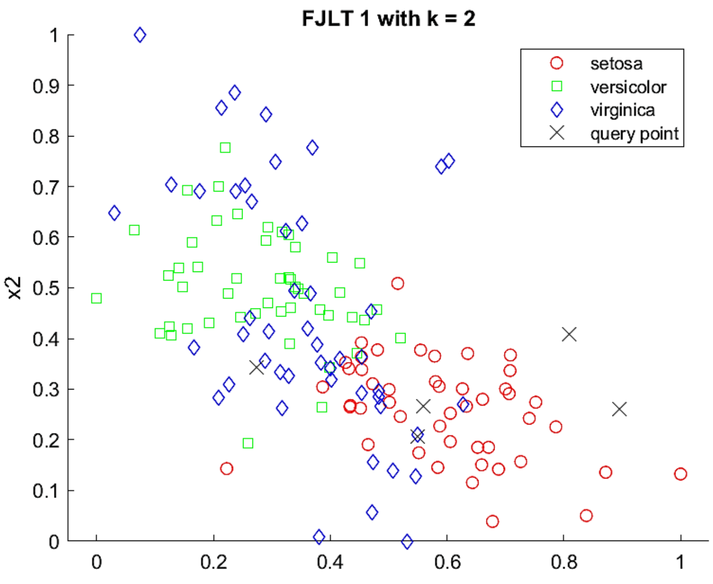


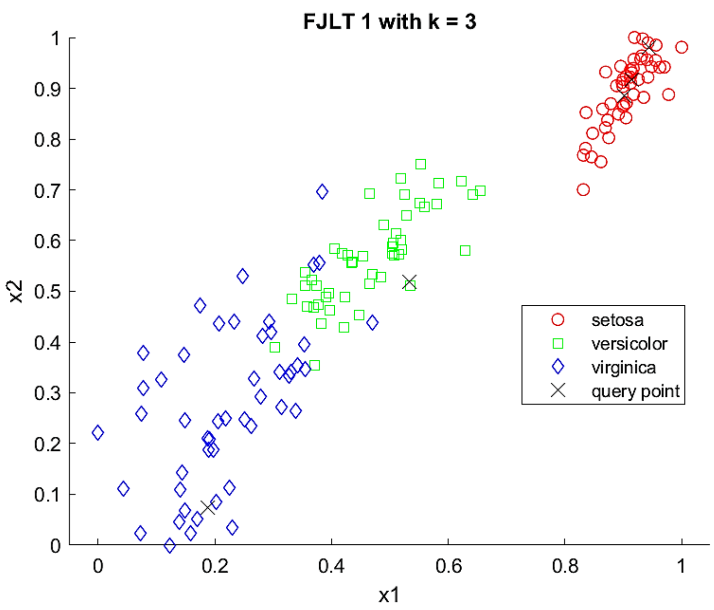


k: reduced dimension; d: original dimension; n: sample size

FJLT 1 on ML



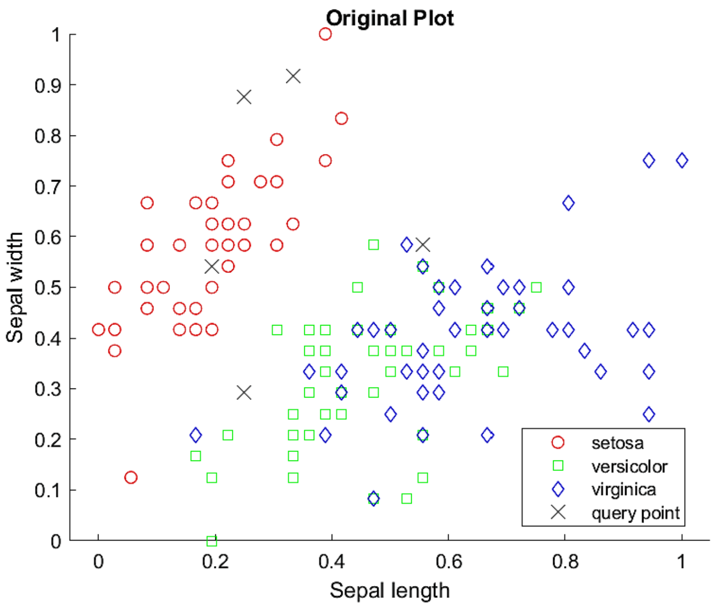


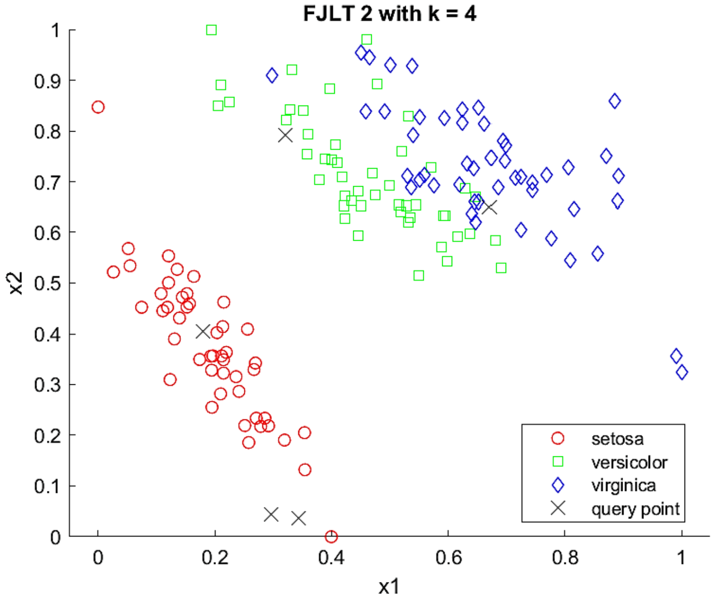


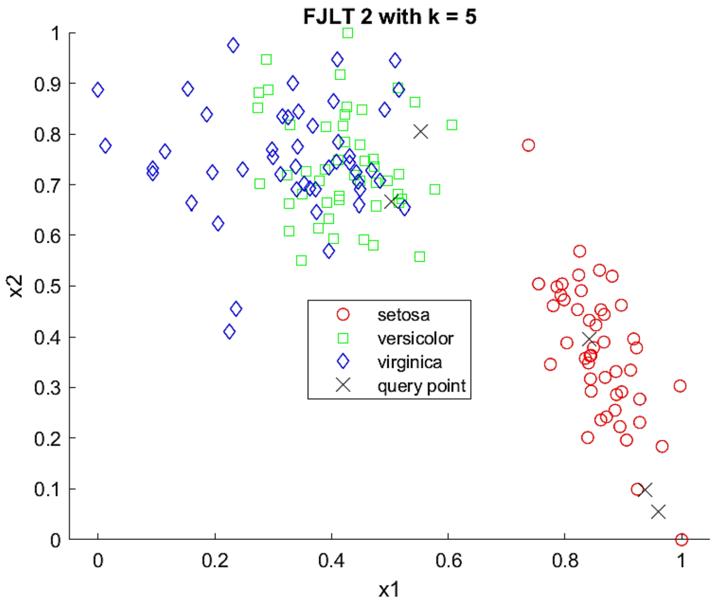
k: reduced dimension; d: original dimension;

Imperial College London

FJLT 2 on ML







k: reduced dimension; d: original dimension;

FJLT on ML

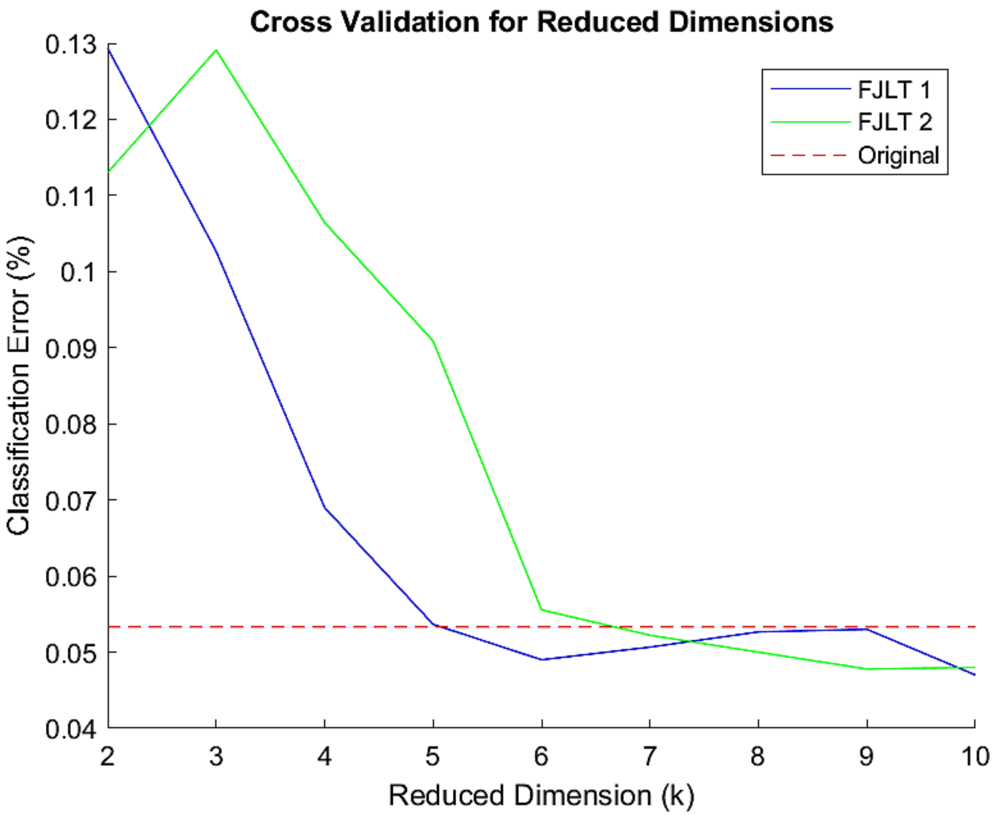
- Cross-validation test

		Actual Class		
		setosa	versicolor	virginica
Predicted Class	setosa	50	0	0
	versicolor	0	48	6
	virginica	0	2	44
	undefined	0	0	0

Table 5.9: Confusion matrix for FJLT 2 projection with k = 4

		Actual Class		
		setosa	versicolor	virginica
Predicted Class	setosa	50	0	0
	versicolor	0	47	2
	virginica	0	3	48
	undefined	0	0	0

Table 5.10: Confusion matrix for FJLT 2 projection with k = 5



k: reduced dimension; d: original dimension; n: sample size

Conclusions

1. FJLT projection method 1
   - Sparse matrices
   - Computation time: $O(d \log d)$; Reduced dimension: $k \leq d^{1/3}$
2. FJLT projection method 2
   - Methods from coding theory
   - Computation time: $O(d \log k)$; Reduced dimension: $k < d^{1/2}$
3. For very complex datasets, the computation time is similar.
   - Better probability of success achieved by FJLT 2
4. Great performance in standard ML techniques

k: reduced dimension; d: original dimension; n: sample size

Future Work

- Simulations of FJLT on large pixel image datasets
  - Emotion/face recognition
- Restriction on reduced dimension
  - Current $k < d^{1/2}$
  - Use of techniques such as Restricted Isometry Property and sparse dimension reduction
- Improving the computation time
  - Better than $O(d \log k)$

k: reduced dimension; d: original dimension; n: sample size

Imperial College London

25th June 2018

Thank you

Devin Nanayakkara

Imperial College London

25th June 2018

APPENDIX

Devin Nanayakkara

FJLT Method 1

- Theorem:

$Given\ a\ fixed\ set\ of\ X\ of\ n\ points\ in\ \mathbb{R}^d, and\ \epsilon < 1,$

$draw\ a\ matrix\ \Phi\ from\ FJLT\ 1.\ With\ probability\ at\ least\ \frac{2}{3}, the\ following$

$2\ events\ occur:$

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

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

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

$$O(d \log d + \min\{d\epsilon^{-2} \log n, \epsilon^{-2} \log^3 n\})$$

$operations.$

k: reduced dimension; d: original dimension; n: sample size

FJLT Method 1

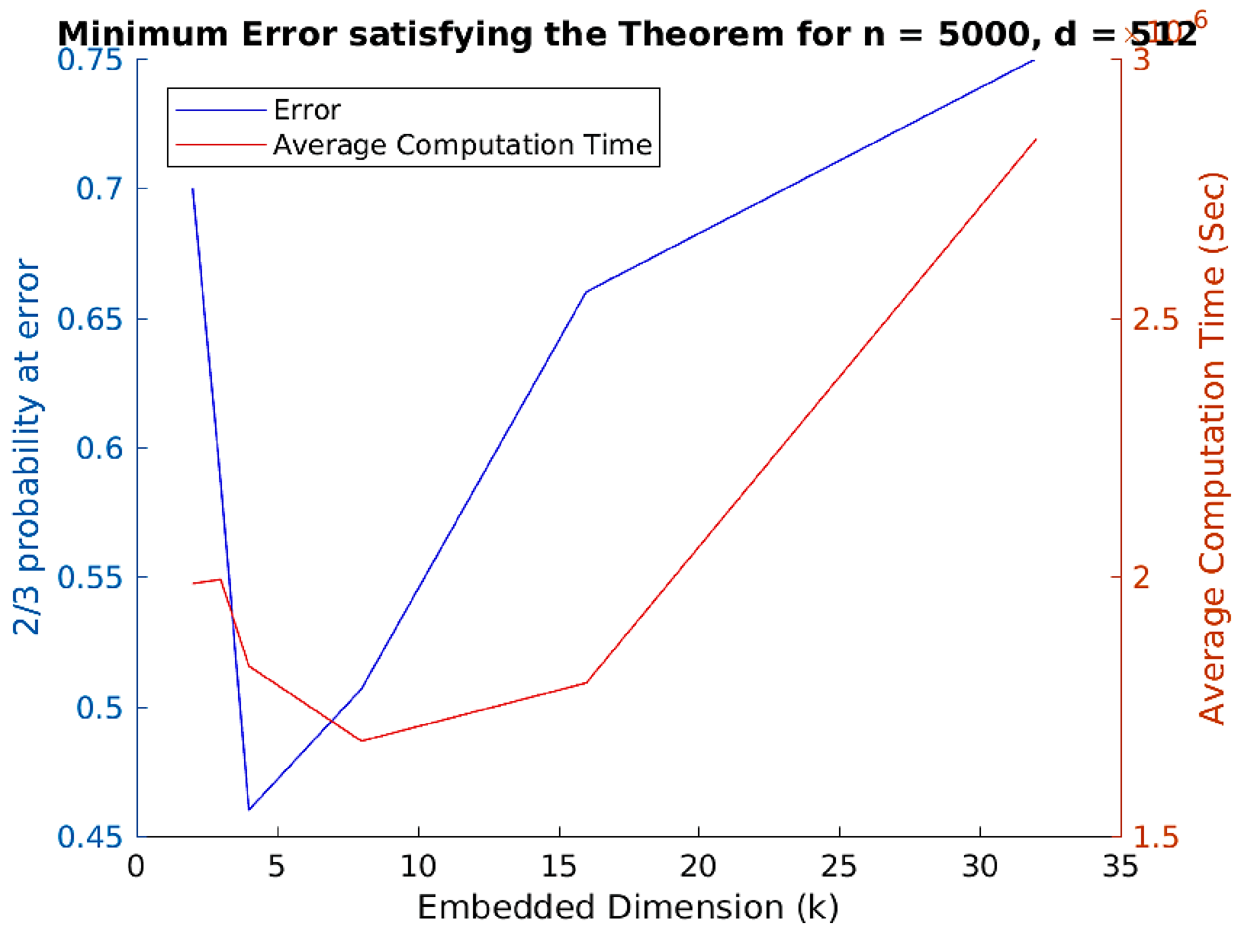
d	$d^{1/3}$	n	k					
			2	**3**	**4**	**8**	**16**	**32**
64	4	1000	0.540	0.540	0.640	0.760	0.880	0.940
		5000	0.460	0.540	0.680	0.820	0.880	0.940
		10000	0.580	0.600	0.720	0.800	0.880	0.940
128	5	1000	0.520	0.420	0.380	0.680	0.800	0.900
		5000	0.500	0.500	0.600	0.740	0.840	0.920
		10000	0.520	0.560	0.620	0.760	0.880	0.920
256	6	1000	0.680	0.520	0.400	0.580	0.720	0.860
		5000	0.600	0.380	0.400	0.640	0.780	0.880
		10000	0.540	0.380	0.440	0.640	0.820	0.900
512	8	1000	NaN	0.740	0.480	0.380	0.620	0.780
		10000	0.780	0.480	0.360	0.520	0.720	0.840
1024	10	10000	NaN	0.840	0.500	0.340	0.600	0.720

Table 5.1: Error at which 2/3 probability is reached for different tests.

k: reduced dimension; d: original dimension; n: sample size

Results

- FJLT 1
  - Test 2
  - $k \leq 8$



k: reduced dimension; d: original dimension; n: sample size

FJLT Method 2

- Theorems:

Theorem 3.2 *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 3.3 *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 $(0 \leq \epsilon \leq 1/2)$, 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 $(p = 2)$ and the Manhattan $(p = 1)$ cases.*

k: reduced dimension; d: original dimension; n: sample size

FJLT Method 2

$$\mathrm{A} = B \,.\, D \,.\, \Phi_d^r$$

$$B_{k\times d} = [B_k \quad B_k \quad B_k \ldots B_k]$$

$$BD = \underbrace{(B_k \quad B_k \quad B_k \ldots B_k)}_{d/\beta \text{ copies of } k\times\beta \text{ blocks}} \underbrace{\begin{pmatrix} D_\beta & 0 & \cdots & 0 \\ 0 & D_\beta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_\beta \end{pmatrix}}_{d/\beta \text{ copies of } \beta\times\beta \text{ blocks}}$$

k: reduced dimension; d: original dimension; n: sample size

FJLT Method 2

$$A = B \,.\, D \,.\, \Phi_d^r$$

$$BD = \underbrace{\begin{pmatrix} B_k & B_k & B_k \ldots B_k \end{pmatrix}}_{d/\beta \text{ copies of } k\times\beta \text{ blocks}} \underbrace{\begin{pmatrix} D_\beta & 0 & \cdots & 0 \\ 0 & D_\beta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_\beta \end{pmatrix}}_{d/\beta \text{ copies of } \beta\times\beta \text{ blocks}}$$

$$HD' = \underbrace{\begin{pmatrix} H_\beta & 0 & \cdots & 0 \\ 0 & H_\beta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H_\beta \end{pmatrix}}_{d/\beta \text{ copies of } \beta\times\beta \text{ blocks}} \underbrace{\begin{pmatrix} D_\beta^{(1)} & 0 & \cdots & 0 \\ 0 & D_\beta^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_\beta^{(r)} \end{pmatrix}}_{d/\beta \text{ copies of } \beta\times\beta \text{ blocks}}$$

- For $k < d^{1/2}$, the runtime is $O(d \log k)$

FJLT Method 2 (1)

Theorem 3.2 *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 3.3 *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 $(0 \leq \epsilon \leq 1/2)$, 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 $(p = 2)$ and the Manhattan $(p = 1)$ cases.*

Lemma 3.1 *There exists a 4-wise independent code matrix of size $k \times f_{BCH}(k)$, where $f_{BCH}(k) = \Theta(k^2)$.*

Lemma 3.2 *Assume B is a $k \times d$ 4-wise independent code matrix.*

k: reduced dimension; d: original dimension; n: sample size

FJLT Method 2 (2)

$$B_{k\times d} = [B_k \quad B_k \quad B_k \ldots B_k]$$

$$D' = D^{(1)}, D^{(2)}, D^{(3)}, \ldots, D^{(r)}$$

$$\Phi_d^{(r)} = HD^{(r)}HD^{(r-1)}\ldots HD^{(1)}$$

$$BD = \underbrace{(B_k \quad B_k \quad B_k \ldots B_k)}_{d/\beta \text{ copies of } k\times\beta \text{ blocks}} \underbrace{\begin{pmatrix} D_\beta & 0 & \cdots & 0 \\ 0 & D_\beta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_\beta \end{pmatrix}}_{d/\beta \text{ copies of } \beta\times\beta \text{ blocks}}$$

$$HD' = \underbrace{\begin{pmatrix} H_\beta & 0 & \cdots & 0 \\ 0 & H_\beta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H_\beta \end{pmatrix}}_{d/\beta \text{ copies of } \beta\times\beta \text{ blocks}} \underbrace{\begin{pmatrix} D_\beta^{(1)} & 0 & \cdots & 0 \\ 0 & D_\beta^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_\beta^{(r)} \end{pmatrix}}_{d/\beta \text{ copies of } \beta\times\beta \text{ blocks}}$$

k: reduced dimension; d: original dimension; n: sample size

FJLT Method 2 (3)

d	$d^{1/2}$	n	k	β	$error$
32	5.7	50	5	16	0.380
64	8	50	6	32	0.350
128	11.3	500	3	8	0.310
		5000	3	8	0.330
256	16	50	8	64	0.340
		500	8	64	0.330
		10000	8	64	0.360
512	22.6	1000	18	256	0.330
		10000	18	256	0.300

Table 5.2: Best performing reduced dimension solution for different tests

k: reduced dimension; d: original dimension; n: sample size

Comparison Tests

		FJLT 1		FJLT 2		
n	d	k	min e	k	beta	min e
50	32	3	0.40	5	16	0.40
	64	4	0.44	6	32	0.38
	128	4	0.40	7	32	0.35
	256	8	0.32	9	64	0.25
	512	16	0.30	16	256	0.25
100	32	2	0.50	5	16	0.38
	64	3	0.40	7	32	0.37
	128	4	0.40	6	32	0.34
	256	8	0.32	10	128	0.34
	512	8	0.36	15	256	0.28
500	32	2	0.52	5	16	0.42
	64	2	0.50	7	32	0.40
	128	3	0.44	7	32	0.37
	256	4	0.40	9	64	0.35
	512	8	0.30	14	256	0.32
1000	32	2	4.80	5	16	0.46
	64	2	0.54	7	32	0.41
	128	4	0.38	7	32	0.38
	256	4	0.40	9	64	0.35
	512	8	0.38	18	256	0.33
5000	32	2	0.58	5	16	0.45
	64	2	0.46	7	32	0.42
	128	3	0.50	7	32	0.39
	256	3	0.38	9	64	0.35
	512	4	0.36	14	256	0.33
10000	32	2	0.50	5	16	0.45
	64	2	0.60	7	32	0.41
	128	2	0.52	7	32	0.39
	256	3	0.38	9	64	0.34
	512	4	0.36	18	256	0.32
	1024	8	0.34	25	512	0.30

Table 1: Optimal reduced dimension and error for different parameters

k: reduced dimension; d: original dimension; n: sample size

FJLT on ML

- Summary of correctness tests

Type	Test Size	Accuracy (as per dataset) %									
		Original	k								
			2	3	4	5	6	7	8	9	10
FJLT 1	5	80.0	78.0	84.0	88.0	86.0	90.0	92.0	90.0	84.0	92.0
FJLT 2	5	80.0	86.0	86.0	84.0	88.0	86.0	90.0	90.0	86.0	86.0
FJLT 1	10	90.0	89.0	88.0	92.0	90.0	91.0	93.0	94.0	92.0	91.0
FJLT 2	10	90.0	92.0	87.0	85.0	87.0	94.0	92.0	93.0	94.0	92.0
FJLT 1	15	100.0	94.0	94.0	100.0	99.3	100.0	100.0	100.0	100.0	100.0
FJLT 2	15	100.0	90.0	95.3	98.0	97.3	100.0	100.0	99.3	99.3	100.0

k: reduced dimension; d: original dimension; n: sample size

l_p-norm

$$||x_p|| = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

ℓ_2 *Regression*

- Least square fit of an overdetermined linear system
- Old method:
  - Solution obtained by down sampling
- Problem:
  - Complex solution
  - Down sampling distribution depends on norms of rows of the left singular vector matrix of the original system
- Solution:
  - Multiply the equation matrix on the left by HD
  - The resulting left singular matrix have almost uniform sampling