The Unreasonable Effectiveness of Random Projections in Computer Science

Robert J. Durrant

University of Waikato,
Department of Statistics

bobd@waikato.ac.nz

www.stats.waikato.ac.nz/~bobd

2nd IEEE ICDM Workshop on High Dimensional Data Mining,
Sunday 14th December 2014

Outline

1. Background and Preliminaries
2. Johnson-Lindenstrauss Lemma (JLL) and extensions
3. Applications of JLL (1)
    - Approximate Nearest Neighbour Search
    - RP Perceptron
    - Mixtures of Gaussians
4. Compressed Sensing
    - SVM from RP sparse data
5. Applications of JLL (2)
    - RP LLS Regression
6. JLL- and CS-free Approaches
    - RP Fisher’s Linear Discriminant
        - Time Complexity Analysis
        - Experiments
    - RP Estimation of Distribution Algorithm

Outline

1. Background and Preliminaries
2. Johnson-Lindenstrauss Lemma (JLL) and extensions
3. Applications of JLL (1)
4. Compressed Sensing
5. Applications of JLL (2)
6. JLL- and CS-free Approaches

Motivation - Dimensionality Curse

The 'curse of dimensionality': A collection of pervasive, and often counterintuitive, issues associated with working with high-dimensional data.

Two typical problems:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

Motivation - Dimensionality Curse

The ‘curse of dimensionality’: A collection of pervasive, and often counterintuitive, issues associated with working with high-dimensional data.

Two typical problems:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

Motivation - Dimensionality Curse

The ‘curse of dimensionality’: A collection of pervasive, and often counterintuitive, issues associated with working with high-dimensional data.
Two typical problems:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

Curse of Dimensionality

Comment:
What constitutes high-dimensional depends on the problem setting, but data vectors with arity in the thousands very common in practice (e.g. medical images, gene activation arrays, text, time series, ...).

Issues can start to show up when data arity in the tens!

We will simply say that the observations, $\mathcal{T}$, are d-dimensional and there are N of them: $\mathcal{T} = \{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^N$ and we will assume that, for whatever reason, d is too large.

Mitigating the Curse of Dimensionality

An obvious solution: Dimensionality d is too large, so reduce d to $k \ll d$.

How?
Dozens of methods: PCA, Factor Analysis, Projection Pursuit, Random Projection ...

We will be focusing on Random Projection, motivated (at first) by the following important result:

Outline

1 Background and Preliminaries

2 Johnson-Lindenstrauss Lemma (JLL) and extensions

3 Applications of JLL (1)

4 Compressed Sensing

5 Applications of JLL (2)

6 JLL- and CS-free Approaches

Johnson-Lindenstrauss Lemma (JLL)

The JLL is the following rather surprising fact:

Theorem (Johnson and Lindenstrauss, 1984)

Let $\epsilon \in (0,1)$*. Let* $N, k \in \mathbb{N}$ *with* $V \subseteq \mathbb{R}^d$ *a set of* N *points and* $k \in \mathcal{O}\left(\min\{d, \epsilon^{-2}\log N\}\right)$*. Then there exists a mapping* $P : \mathbb{R}^d \to \mathbb{R}^k$*, such that for all* $u, v \in V$*:*

$$(1-\epsilon)\|u-v\|_d^2 \leqslant \|Pu - Pv\|_k^2 \leqslant (1+\epsilon)\|u-v\|_d^2$$

- Dot products are also approximately preserved by P since if JLL holds then: $u^T v - \epsilon \leqslant (Pu)^T Pv \leqslant u^T v + \epsilon$. (Proof: parallelogram law).
- Note that the projection dimension k depends only on ϵ and N.
- For linear mappings scale of k is sharp: *No linear dimensionality reduction can improve on JLL guarantee* for an arbitrary point set. Lower bound $k \in \Omega\left(\min\{d, \epsilon^{-2}\log N\}\right)$ [LN14].

Johnson-Lindenstrauss Lemma (JLL)

The JLL is the following rather surprising fact:

Theorem (Johnson and Lindenstrauss, 1984)

Let $\epsilon \in (0, 1)$. *Let* $N, k \in \mathbb{N}$ *with* $V \subseteq \mathbb{R}^d$ *a set of* N *points and* $k \in \mathcal{O}\left(\min\{d, \epsilon^{-2} \log N\}\right)$. *Then there exists a mapping* $P : \mathbb{R}^d \to \mathbb{R}^k$, *such that for all* $u, v \in V$:

$$(1 - \epsilon)\|u - v\|_d^2 \leqslant \|Pu - Pv\|_k^2 \leqslant (1 + \epsilon)\|u - v\|_d^2$$

- Dot products are also approximately preserved by P since if JLL holds then: $u^T v - \epsilon \leqslant (Pu)^T Pv \leqslant u^T v + \epsilon$. (Proof: parallelogram law).
- Note that the projection dimension k depends only on ϵ and N.
- For linear mappings scale of k is sharp: *No linear dimensionality reduction can improve on JLL guarantee* for an arbitrary point set. Lower bound $k \in \Omega\left(\min\{d, \epsilon^{-2} \log N\}\right)$ [LN14].

Johnson-Lindenstrauss Lemma (JLL)

The JLL is the following rather surprising fact:

Theorem (Johnson and Lindenstrauss, 1984)

Let $\epsilon \in (0, 1)$. *Let* $N, k \in \mathbb{N}$ *with* $V \subseteq \mathbb{R}^d$ *a set of* N *points and* $k \in \mathcal{O}\left(\min\{d, \epsilon^{-2} \log N\}\right)$. *Then there exists a mapping* $P : \mathbb{R}^d \to \mathbb{R}^k$, *such that for all* $u, v \in V$*:*

$$(1 - \epsilon)\|u - v\|_d^2 \leqslant \|Pu - Pv\|_k^2 \leqslant (1 + \epsilon)\|u - v\|_d^2$$

- Dot products are also approximately preserved by P since if JLL holds then: $u^T v - \epsilon \leqslant (Pu)^T Pv \leqslant u^T v + \epsilon$. (Proof: parallelogram law).
- Note that the projection dimension k depends only on ϵ and N.
- For linear mappings scale of k is sharp: *No linear dimensionality reduction can improve on JLL guarantee* for an arbitrary point set. Lower bound $k \in \Omega\left(\min\{d, \epsilon^{-2} \log N\}\right)$ [LN14].

Johnson-Lindenstrauss Lemma (JLL)

The JLL is the following rather surprising fact:

Theorem (Johnson and Lindenstrauss, 1984)

Let $\epsilon \in (0,1)$*. Let* $N, k \in \mathbb{N}$ *with* $V \subseteq \mathbb{R}^d$ *a set of* N *points and* $k \in \mathcal{O}\left(\min\{d, \epsilon^{-2}\log N\}\right)$*. Then there exists a mapping* $P : \mathbb{R}^d \to \mathbb{R}^k$*, such that for all* $u, v \in V$*:*

$$(1-\epsilon)\|u-v\|_d^2 \leqslant \|Pu-Pv\|_k^2 \leqslant (1+\epsilon)\|u-v\|_d^2$$

- Dot products are also approximately preserved by P since if JLL holds then: $u^Tv - \epsilon \leqslant (Pu)^T Pv \leqslant u^Tv + \epsilon$. (Proof: parallelogram law).
- Note that the projection dimension k depends only on ϵ and N.
- For linear mappings scale of k is sharp: *No linear dimensionality reduction can improve on JLL guarantee* for an arbitrary point set. Lower bound $k \in \Omega\left(\min\{d, \epsilon^{-2}\log N\}\right)$ [LN14].

Distributional JLL

Theorem (Distributional JLL [DG02, Ach03])

Let $\epsilon, \delta \in (0,1)$. Let $k \in \mathbb{N}$ such that $k \in \mathcal{O}\left(\epsilon^{-2} \log \delta^{-1}\right)$. Then there is a ***random linear mapping*** *$R : \mathbb{R}^d \to \mathbb{R}^k$ such that for any vector $x \in \mathbb{R}^d$, with probability at least $1 - \delta$ it holds that:*

$$(1-\epsilon)\|x\|_d^2 \leqslant \|Rx\|_k^2 \leqslant (1+\epsilon)\|x\|_d^2$$

- Note that the projection dimension k depends only on ϵ and δ.
- For random linear mappings scale of k is sharp: Lower bound $k \in \Omega(\epsilon^{-2} \log \delta^{-1})$ sharp for randomized dimensionality reduction [KMN11].
- Taking $\delta^{-1} \in \mathcal{O}\binom{N}{2} \simeq N^2$ obtain same order of k as canonical JLL.
- We call the $k \times d$ matrix R a 'random projection'. As far as geometry preservation goes, RP is (w.h.p, or on average) essentially as good as best possible *deterministic* linear scheme.

Distributional JLL

Theorem (Distributional JLL [DG02, Ach03])

Let $\epsilon, \delta \in (0,1)$. Let $k \in \mathbb{N}$ such that $k \in \mathcal{O}\left(\epsilon^{-2} \log \delta^{-1}\right)$. Then there is a ***random linear mapping*** *$R : \mathbb{R}^d \to \mathbb{R}^k$ such that for any vector $x \in \mathbb{R}^d$, with probability at least $1 - \delta$ it holds that:*

$$(1-\epsilon)\|x\|_d^2 \leqslant \|Rx\|_k^2 \leqslant (1+\epsilon)\|x\|_d^2$$

- Note that the projection dimension k depends only on ϵ and δ.
- For random linear mappings scale of k is sharp: Lower bound $k \in \Omega(\epsilon^{-2} \log \delta^{-1})$ sharp for randomized dimensionality reduction [KMN11].
- Taking $\delta^{-1} \in \mathcal{O}\binom{N}{2} \simeq N^2$ obtain same order of k as canonical JLL.
- We call the $k \times d$ matrix R a 'random projection'. As far as geometry preservation goes, RP is (w.h.p, or on average) essentially as good as best possible *deterministic* linear scheme.

Distributional JLL

Theorem (Distributional JLL [DG02, Ach03])

Let $\epsilon, \delta \in (0,1)$. Let $k \in \mathbb{N}$ such that $k \in \mathcal{O}\left(\epsilon^{-2} \log \delta^{-1}\right)$. Then there is a ***random linear mapping*** *$R : \mathbb{R}^d \to \mathbb{R}^k$ such that for any vector $x \in \mathbb{R}^d$, with probability at least $1 - \delta$ it holds that:*

$$(1-\epsilon)\|x\|_d^2 \leqslant \|Rx\|_k^2 \leqslant (1+\epsilon)\|x\|_d^2$$

- Note that the projection dimension k depends only on ϵ and δ.
- For random linear mappings scale of k is sharp: Lower bound $k \in \Omega(\epsilon^{-2} \log \delta^{-1})$ sharp for randomized dimensionality reduction [KMN11].
- Taking $\delta^{-1} \in \mathcal{O}\binom{N}{2} \simeq N^2$ obtain same order of k as canonical JLL.
- We call the $k \times d$ matrix R a 'random projection'. As far as geometry preservation goes, RP is (w.h.p, or on average) essentially as good as best possible *deterministic* linear scheme.

Distributional JLL

Theorem (Distributional JLL [DG02, Ach03])

Let $\epsilon, \delta \in (0, 1)$*. Let* $k \in \mathbb{N}$ *such that* $k \in \mathcal{O}\left(\epsilon^{-2} \log \delta^{-1}\right)$*. Then there is a* ***random linear mapping*** $R : \mathbb{R}^d \to \mathbb{R}^k$ *such that for any vector* $x \in \mathbb{R}^d$*, with probability at least* $1 - \delta$ *it holds that:*

$$(1 - \epsilon)\|x\|_d^2 \leqslant \|Rx\|_k^2 \leqslant (1 + \epsilon)\|x\|_d^2$$

- Note that the projection dimension k depends only on ϵ and δ.
- For random linear mappings scale of k is sharp: Lower bound $k \in \Omega(\epsilon^{-2} \log \delta^{-1})$ sharp for randomized dimensionality reduction [KMN11].
- Taking $\delta^{-1} \in \mathcal{O}\binom{N}{2} \simeq N^2$ obtain same order of k as canonical JLL.
- We call the $k \times d$ matrix R a ‘random projection’. As far as geometry preservation goes, RP is (w.h.p, or on average) essentially as good as best possible *deterministic* linear scheme.

Distributional JLL

Theorem (Distributional JLL [DG02, Ach03])

Let $\epsilon, \delta \in (0, 1)$. Let $k \in \mathbb{N}$ such that $k \in \mathcal{O}\left(\epsilon^{-2} \log \delta^{-1}\right)$. Then there is a ***random linear mapping*** *$R : \mathbb{R}^d \to \mathbb{R}^k$ such that for any vector $x \in \mathbb{R}^d$, with probability at least $1 - \delta$ it holds that:*

$$(1 - \epsilon)\|x\|_d^2 \leqslant \|Rx\|_k^2 \leqslant (1 + \epsilon)\|x\|_d^2$$

- Note that the projection dimension k depends only on ϵ and δ.
- For random linear mappings scale of k is sharp: Lower bound $k \in \Omega(\epsilon^{-2} \log \delta^{-1})$ sharp for randomized dimensionality reduction [KMN11].
- Taking $\delta^{-1} \in \mathcal{O}\binom{N}{2} \simeq N^2$ obtain same order of k as canonical JLL.
- We call the $k \times d$ matrix R a ‘random projection’. As far as geometry preservation goes, RP is (w.h.p, or on average) essentially as good as best possible *deterministic* linear scheme.

Intuition

Geometry of data gets perturbed by random projection, but not too much:

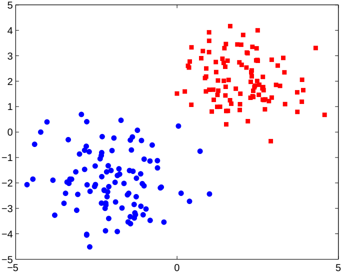


Figure: Original data

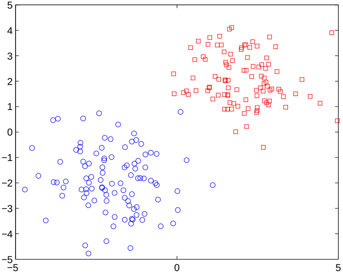


Figure: RP data (schematic)

Intuition

Geometry of data gets perturbed by random projection, but not too much:

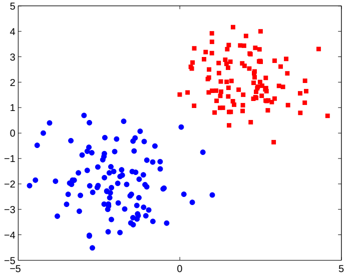


Figure: Original data

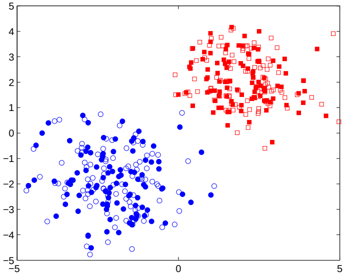


Figure: RP data & Original data

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], . . .

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], . . .

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], . . .

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], . . .

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], . . .

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], ...

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], …

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], . . .

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], . . .

Applications

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK10, DK14]
- Regression. e.g. [MM09, HWB07, BD09, Kab14]
- Clustering and Density estimation. e.g. [IM98, AC06, FB03, Das99, KMV12, AV09]
- Other related applications: structure-adaptive kd-trees [DF08], low-rank matrix approximation [Rec11, Sar06], sparse signal reconstruction (compressed sensing) [Don06, CT06], data stream computations [AMS96], real-valued optimization [KBD13], . . .

Distributional JLL - Proof Ideas

- Construct a $k \times d$ matrix R with entries $R_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ and fix some arbitrary vector $x \in \mathbb{R}^d$.
- Show that the expected squared Euclidean norm of the mapped vector Rx is $\mathrm{E}[\|Rx\|^2] = k\|x\|^2/d\sigma^2$.
- Show that with high probability $\|Rx\|^2$ is close to its expectation.
- For an N point set $\mathcal{T} := \{z_1, z_2, \ldots, z_N | z_i \in \mathbb{R}^d\}$ instantiate x as the vector $x_{ij} := z_i - z_j$, $i < j$. There are $N(N-1)/2$ such pairs.
- Obtain guarantee that all interpoint distances are approximately preserved w.p. at least $1 - \delta$ by applying union bound – i.e. Let A_{ij} be the event that the projection of x_{ij} has greater than ϵ distortion and use $\Pr(\bigvee_{i<j} A_{ij}) \leqslant \sum_{i<j} Pr(A_{ij}) =: \delta$.
- With probability at least $1 - \delta$, $\sqrt{d\sigma^2/k}R$ is a JLL mapping.

Note that proof of distributional JLL is *constructive* - it tells us how to construct a JLL embedding w.h.p. using a random matrix.

Distributional JLL - Proof Ideas

- Construct a $k \times d$ matrix R with entries $R_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ and fix some arbitrary vector $x \in \mathbb{R}^d$.
- Show that the expected squared Euclidean norm of the mapped vector Rx is $\text{E}[\|Rx\|^2] = k\|x\|^2/d\sigma^2$.
- Show that with high probability $\|Rx\|^2$ is close to its expectation.
- For an N point set $\mathcal{T} := \{z_1, z_2, \ldots, z_N | z_i \in \mathbb{R}^d\}$ instantiate x as the vector $x_{ij} := z_i - z_j$, $i < j$. There are $N(N-1)/2$ such pairs.
- Obtain guarantee that all interpoint distances are approximately preserved w.p. at least $1 - \delta$ by applying union bound – i.e. Let A_{ij} be the event that the projection of x_{ij} has greater than ϵ distortion and use $\Pr(\bigvee_{i<j} A_{ij}) \leqslant \sum_{i<j} Pr(A_{ij}) =: \delta$.
- With probability at least $1 - \delta$, $\sqrt{d\sigma^2/k}R$ is a JLL mapping.

Note that proof of distributional JLL is *constructive* - it tells us how to construct a JLL embedding w.h.p. using a random matrix.

Distributional JLL - Proof Ideas

- Construct a $k \times d$ matrix R with entries $R_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ and fix some arbitrary vector $x \in \mathbb{R}^d$.
- Show that the expected squared Euclidean norm of the mapped vector Rx is $\mathrm{E}[\|Rx\|^2] = k\|x\|^2/d\sigma^2$.
- Show that with high probability $\|Rx\|^2$ is close to its expectation.
- For an N point set $\mathcal{T} := \{z_1, z_2, \ldots, z_N | z_i \in \mathbb{R}^d\}$ instantiate x as the vector $x_{ij} := z_i - z_j$, $i < j$. There are $N(N-1)/2$ such pairs.
- Obtain guarantee that all interpoint distances are approximately preserved w.p. at least $1 - \delta$ by applying union bound – i.e. Let A_{ij} be the event that the projection of x_{ij} has greater than ϵ distortion and use $\Pr(\bigvee_{i<j} A_{ij}) \leqslant \sum_{i<j} Pr(A_{ij}) =: \delta$.
- With probability at least $1 - \delta$, $\sqrt{d\sigma^2/k}R$ is a JLL mapping.

Note that proof of distributional JLL is *constructive* - it tells us how to construct a JLL embedding w.h.p. using a random matrix.

Distributional JLL - Proof Ideas

- Construct a $k \times d$ matrix R with entries $R_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ and fix some arbitrary vector $x \in \mathbb{R}^d$.
- Show that the expected squared Euclidean norm of the mapped vector Rx is $\mathrm{E}[\|Rx\|^2] = k\|x\|^2/d\sigma^2$.
- Show that with high probability $\|Rx\|^2$ is close to its expectation.
- For an N point set $\mathcal{T} := \{z_1, z_2, \ldots, z_N | z_i \in \mathbb{R}^d\}$ instantiate x as the vector $x_{ij} := z_i - z_j$, $i < j$. There are $N(N-1)/2$ such pairs.
- Obtain guarantee that all interpoint distances are approximately preserved w.p. at least $1 - \delta$ by applying union bound – i.e. Let A_{ij} be the event that the projection of x_{ij} has greater than ϵ distortion and use $\Pr(\bigvee_{i<j} A_{ij}) \leqslant \sum_{i<j} Pr(A_{ij}) =: \delta$.
- With probability at least $1 - \delta$, $\sqrt{d\sigma^2/k}R$ is a JLL mapping.

Note that proof of distributional JLL is *constructive* - it tells us how to construct a JLL embedding w.h.p. using a random matrix.

Distributional JLL - Proof Ideas

- Construct a $k \times d$ matrix R with entries $R_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ and fix some arbitrary vector $x \in \mathbb{R}^d$.
- Show that the expected squared Euclidean norm of the mapped vector Rx is $\mathrm{E}[\|Rx\|^2] = k\|x\|^2/d\sigma^2$.
- Show that with high probability $\|Rx\|^2$ is close to its expectation.
- For an N point set $\mathcal{T} := \{z_1, z_2, \ldots, z_N | z_i \in \mathbb{R}^d\}$ instantiate x as the vector $x_{ij} := z_i - z_j$, $i < j$. There are $N(N-1)/2$ such pairs.
- Obtain guarantee that all interpoint distances are approximately preserved w.p. at least $1 - \delta$ by applying union bound – i.e. Let A_{ij} be the event that the projection of x_{ij} has greater than ϵ distortion and use $\Pr(\bigvee_{i<j} A_{ij}) \leqslant \sum_{i<j} Pr(A_{ij}) =: \delta$.
- With probability at least $1 - \delta$, $\sqrt{d\sigma^2/k}R$ is a JLL mapping.

Note that proof of distributional JLL is *constructive* - it tells us how to construct a JLL embedding w.h.p. using a random matrix.

Distributional JLL - Proof Ideas

- Construct a $k \times d$ matrix R with entries $R_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ and fix some arbitrary vector $x \in \mathbb{R}^d$.
- Show that the expected squared Euclidean norm of the mapped vector Rx is $\mathrm{E}[\|Rx\|^2] = k\|x\|^2/d\sigma^2$.
- Show that with high probability $\|Rx\|^2$ is close to its expectation.
- For an N point set $\mathcal{T} := \{z_1, z_2, \ldots, z_N | z_i \in \mathbb{R}^d\}$ instantiate x as the vector $x_{ij} := z_i - z_j$, $i < j$. There are $N(N-1)/2$ such pairs.
- Obtain guarantee that all interpoint distances are approximately preserved w.p. at least $1 - \delta$ by applying union bound – i.e. Let A_{ij} be the event that the projection of x_{ij} has greater than ϵ distortion and use $\Pr(\bigvee_{i<j} A_{ij}) \leqslant \sum_{i<j} Pr(A_{ij}) =: \delta$.
- With probability at least $1 - \delta$, $\sqrt{d\sigma^2/k}R$ is a JLL mapping.

Note that proof of distributional JLL is *constructive* - it tells us how to construct a JLL embedding w.h.p. using a random matrix.

Distributional JLL - Proof Ideas

- Construct a $k \times d$ matrix R with entries $R_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ and fix some arbitrary vector $x \in \mathbb{R}^d$.
- Show that the expected squared Euclidean norm of the mapped vector Rx is $\mathrm{E}[\|Rx\|^2] = k\|x\|^2/d\sigma^2$.
- Show that with high probability $\|Rx\|^2$ is close to its expectation.
- For an N point set $\mathcal{T} := \{z_1, z_2, \ldots, z_N | z_i \in \mathbb{R}^d\}$ instantiate x as the vector $x_{ij} := z_i - z_j$, $i < j$. There are $N(N-1)/2$ such pairs.
- Obtain guarantee that all interpoint distances are approximately preserved w.p. at least $1 - \delta$ by applying union bound – i.e. Let A_{ij} be the event that the projection of x_{ij} has greater than ϵ distortion and use $\mathrm{Pr}(\bigvee_{i<j} A_{ij}) \leqslant \sum_{i<j} Pr(A_{ij}) =: \delta$.
- With probability at least $1 - \delta$, $\sqrt{d\sigma^2/k}R$ is a JLL mapping.

Note that proof of distributional JLL is *constructive* - it tells us how to construct a JLL embedding w.h.p. using a random matrix.

Distributional JLL - Proof Ideas

- Construct a $k \times d$ matrix R with entries $R_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ and fix some arbitrary vector $x \in \mathbb{R}^d$.
- Show that the expected squared Euclidean norm of the mapped vector Rx is $\mathrm{E}[\|Rx\|^2] = k\|x\|^2/d\sigma^2$.
- Show that with high probability $\|Rx\|^2$ is close to its expectation.
- For an N point set $\mathcal{T} := \{z_1, z_2, \ldots, z_N | z_i \in \mathbb{R}^d\}$ instantiate x as the vector $x_{ij} := z_i - z_j$, $i < j$. There are $N(N-1)/2$ such pairs.
- Obtain guarantee that all interpoint distances are approximately preserved w.p. at least $1 - \delta$ by applying union bound – i.e. Let A_{ij} be the event that the projection of x_{ij} has greater than ϵ distortion and use $\Pr(\bigvee_{i<j} A_{ij}) \leqslant \sum_{i<j} Pr(A_{ij}) =: \delta$.
- With probability at least $1 - \delta$, $\sqrt{d\sigma^2/k}R$ is a JLL mapping.

Note that proof of distributional JLL is *constructive* - it tells us how to construct a JLL embedding w.h.p. using a random matrix.

What is Random Projection? (1)

Canonical RP:

- Construct a (wide, flat) matrix $R \in \mathcal{M}_{k \times d}$ by picking the entries i.i.d from a zero-mean Gaussian $r_{ij} \sim \mathcal{N}(0, \sigma^2)$.
- Orthonormalize the rows of R, e.g. set $R' = (RR^T)^{-1/2}R$.
- To project a point $v \in \mathbb{R}^d$, pre-multiply the vector v with RP matrix R'. Then $v \mapsto R'v \in R'(\mathbb{R}^d) \equiv \mathbb{R}^k$ is the projection of the d-dimensional data into a random k-dimensional projection space.

R' is like 'half a projection matrix' in the sense that if $P = (R')^T R'$, then P is a projection matrix in the standard sense.

What is Random Projection? (1)

Canonical RP:

- Construct a (wide, flat) matrix $R \in \mathcal{M}_{k \times d}$ by picking the entries i.i.d from a zero-mean Gaussian $r_{ij} \sim \mathcal{N}(0, \sigma^2)$.
- Orthonormalize the rows of R, e.g. set $R' = (RR^T)^{-1/2}R$.
- To project a point $v \in \mathbb{R}^d$, pre-multiply the vector v with RP matrix R'. Then $v \mapsto R'v \in R'(\mathbb{R}^d) \equiv \mathbb{R}^k$ is the projection of the d-dimensional data into a random k-dimensional projection space.

R' is like 'half a projection matrix' in the sense that if $P = (R')^T R'$, then P is a projection matrix in the standard sense.

What is Random Projection? (1)

Canonical RP:

- Construct a (wide, flat) matrix $R \in \mathcal{M}_{k \times d}$ by picking the entries i.i.d from a zero-mean Gaussian $r_{ij} \sim \mathcal{N}(0, \sigma^2)$.
- Orthonormalize the rows of R, e.g. set $R' = (RR^T)^{-1/2}R$.
- To project a point $v \in \mathbb{R}^d$, pre-multiply the vector v with RP matrix R'. Then $v \mapsto R'v \in R'(\mathbb{R}^d) \equiv \mathbb{R}^k$ is the projection of the d-dimensional data into a random k-dimensional projection space.

R' is like 'half a projection matrix' in the sense that if $P = (R')^T R'$, then P is a projection matrix in the standard sense.

What is Random Projection? (1)

Canonical RP:

- Construct a (wide, flat) matrix $R \in \mathcal{M}_{k \times d}$ by picking the entries i.i.d from a zero-mean Gaussian $r_{ij} \sim \mathcal{N}(0, \sigma^2)$.
- Orthonormalize the rows of R, e.g. set $R' = (RR^T)^{-1/2}R$.
- To project a point $v \in \mathbb{R}^d$, pre-multiply the vector v with RP matrix R'. Then $v \mapsto R'v \in R'(\mathbb{R}^d) \equiv \mathbb{R}^k$ is the projection of the d-dimensional data into a random k-dimensional projection space.

R' is like 'half a projection matrix' in the sense that if $P = (R')^T R'$, then P is a projection matrix in the standard sense.

What is Random Projection? (1)

Canonical RP:

- Construct a (wide, flat) matrix $R \in \mathcal{M}_{k \times d}$ by picking the entries i.i.d from a zero-mean Gaussian $r_{ij} \sim \mathcal{N}(0, \sigma^2)$.
- Orthonormalize the rows of R, e.g. set $R' = (RR^T)^{-1/2}R$.
- To project a point $v \in \mathbb{R}^d$, pre-multiply the vector v with RP matrix R'. Then $v \mapsto R'v \in R'(\mathbb{R}^d) \equiv \mathbb{R}^k$ is the projection of the d-dimensional data into a random k-dimensional projection space.

R' is like 'half a projection matrix' in the sense that if $P = (R')^T R'$, then P is a projection matrix in the standard sense.

Comment (1)

If d is very large we can drop the orthonormalization in practice - the rows of R will be nearly orthogonal to each other and all nearly the same length.

For example, for Gaussian ($\mathcal{N}(0, \sigma^2)$) R we have [DK12]:

$$\Pr\left\{(1-\epsilon)d\sigma^2 \leqslant \|R_i\|^2 \leqslant (1+\epsilon)d\sigma^2\right\} \geqslant 1-\delta,\ \forall\epsilon \in (0,1]$$

where R_i denotes the i-th row of R and $\delta = \exp(-(\sqrt{1+\epsilon}-1)^2 d/2) + \exp(-(\sqrt{1-\epsilon}-1)^2 d/2)$.

Similarly [Led01]:

$$\Pr\{|R_i^T R_j|/d\sigma^2 \leqslant \epsilon\} \geqslant 1 - 2\exp(-\epsilon^2 d/2),\ \forall i \neq j.$$

Comment (1)

If d is very large we can drop the orthonormalization in practice - the rows of R will be nearly orthogonal to each other and all nearly the same length.

For example, for Gaussian ($\mathcal{N}(0, \sigma^2)$) R we have [DK12]:

$$\Pr\left\{(1-\epsilon)d\sigma^2 \leqslant \|R_i\|^2 \leqslant (1+\epsilon)d\sigma^2\right\} \geqslant 1-\delta,\ \forall\epsilon \in (0,1]$$

where R_i denotes the i-th row of R and
$\delta = \exp(-(\sqrt{1+\epsilon}-1)^2 d/2) + \exp(-(\sqrt{1-\epsilon}-1)^2 d/2)$.

Similarly [Led01]:

$$\Pr\{|R_i^T R_j|/d\sigma^2 \leqslant \epsilon\} \geqslant 1 - 2\exp(-\epsilon^2 d/2),\ \forall i \neq j.$$

Comment (1)

If d is very large we can drop the orthonormalization in practice - the rows of R will be nearly orthogonal to each other and all nearly the same length.

For example, for Gaussian ($\mathcal{N}(0, \sigma^2)$) R we have [DK12]:

$$\Pr\left\{(1-\epsilon)d\sigma^2 \leqslant \|R_i\|^2 \leqslant (1+\epsilon)d\sigma^2\right\} \geqslant 1-\delta,\ \forall \epsilon \in (0,1]$$

where R_i denotes the i-th row of R and
$\delta = \exp(-(\sqrt{1+\epsilon}-1)^2 d/2) + \exp(-(\sqrt{1-\epsilon}-1)^2 d/2)$.

Similarly [Led01]:

$$\Pr\{|R_i^T R_j|/d\sigma^2 \leqslant \epsilon\} \geqslant 1 - 2\exp(-\epsilon^2 d/2),\ \forall i \neq j.$$

Concentration of norms in rows of R

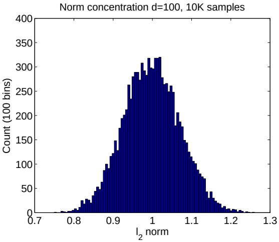


Figure: $d = 100$ norm concentration

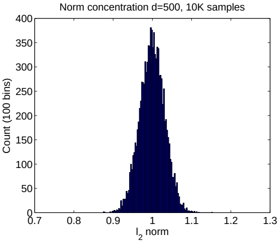


Figure: $d = 500$ norm concentration

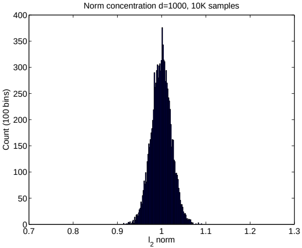


Figure: $d = 1000$ norm concentration

Near-orthogonality of rows of *R*

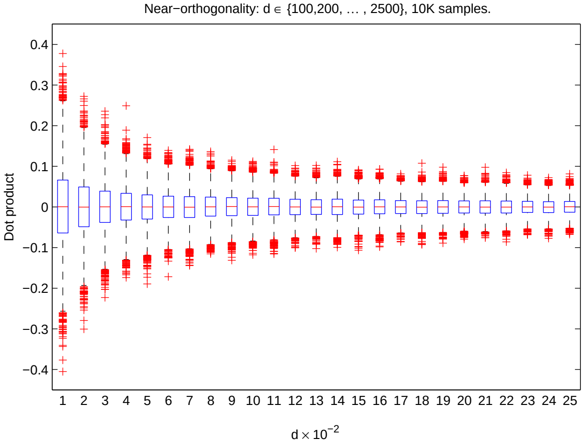


Figure: Normalized dot product is concentrated about zero, $d \in \{100, 200, \ldots, 2500\}$

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$, RP$\in \mathcal{O}\left(k^2d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2}\log N\right)$ – JLL.
  - Any fixed, finite point set w.h.p
  - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- Tractable to analysis.

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$, RP$\in \mathcal{O}\left(k^2d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2}\log N\right)$ – JLL.
  - Any fixed, finite point set w.h.p
  - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- Tractable to analysis.

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$, RP$\in \mathcal{O}\left(k^2d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2}\log N\right)$ – JLL.
  - Any fixed, finite point set w.h.p
  - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- Tractable to analysis.

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$, RP$\in \mathcal{O}\left(k^2d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2} \log N\right)$ – JLL.
  - Any fixed, finite point set w.h.p
  - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- Tractable to analysis.

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$, RP$\in \mathcal{O}\left(k^2d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2} \log N\right)$ – JLL.
  - Any fixed, finite point set w.h.p
  - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- Tractable to analysis.

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2 N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$, RP$\in \mathcal{O}\left(k^2 d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2} \log N\right)$ – JLL.
  - Any fixed, finite point set w.h.p
  - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- Tractable to analysis.

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$, RP$\in \mathcal{O}\left(k^2d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2}\log N\right)$ – JLL.
  - Any fixed, finite point set w.h.p
  - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- Tractable to analysis.

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$,
  RP$\in \mathcal{O}\left(k^2d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2}\log N\right)$ – JLL.
  - Any fixed, finite point set w.h.p
  - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- Tractable to analysis.

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$, RP$\in \mathcal{O}\left(k^2d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2}\log N\right)$ – JLL.
    - Any fixed, finite point set w.h.p
    - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- Tractable to analysis.

Why Random Projection?

Various motivations including:

- Linear.
- Cheap: E.g. PCA$\in \mathcal{O}\left(d^2N\right) + \mathcal{O}\left(d^3\right) + \mathcal{O}\left(Nkd\right)$, RP$\in \mathcal{O}\left(k^2d\right) + \mathcal{O}\left(Nkd\right)$.
- Easy to implement.
- Approximates an isometry/uniform scaling when $k \in \mathcal{O}\left(\epsilon^{-2} \log N\right)$ – JLL.
    - Any fixed, finite point set w.h.p.
    - Projection dimension independent of data dimensionality.
- JLL geometry preservation guarantee optimal for linear mappings.
- Oblivious to data distribution.
- **Tractable to analysis.**

Comment (2)

Random matrices with symmetric zero-mean *sub-Gaussian* entries also have similar properties.

Sub-Gaussians are those distributions whose tails decay no slower than a Gaussian, e.g. practically all bounded distributions have this property.

We obtain similar guarantees (i.e. up to small multiplicative constants) for sub-Gaussian RP matrices too!

This allows us to get around issue of *dense matrix multiplication* in dimensionality-reduction step.

Comment (2)

Random matrices with symmetric zero-mean *sub-Gaussian* entries also have similar properties.

Sub-Gaussians are those distributions whose tails decay no slower than a Gaussian, e.g. practically all bounded distributions have this property.

We obtain similar guarantees (i.e. up to small multiplicative constants) for sub-Gaussian RP matrices too!

This allows us to get around issue of *dense matrix multiplication* in dimensionality-reduction step.

Comment (2)

Random matrices with symmetric zero-mean *sub-Gaussian* entries also have similar properties.

Sub-Gaussians are those distributions whose tails decay no slower than a Gaussian, e.g. practically all bounded distributions have this property.

We obtain similar guarantees (i.e. up to small multiplicative constants) for sub-Gaussian RP matrices too!

This allows us to get around issue of *dense matrix multiplication* in dimensionality-reduction step.

Comment (2)

Random matrices with symmetric zero-mean *sub-Gaussian* entries also have similar properties.

Sub-Gaussians are those distributions whose tails decay no slower than a Gaussian, e.g. practically all bounded distributions have this property.

We obtain similar guarantees (i.e. up to small multiplicative constants) for sub-Gaussian RP matrices too!

This allows us to get around issue of *dense matrix multiplication* in dimensionality-reduction step.

What is Random Projection? (2)

Different types of RP matrix easy to construct - take entries i.i.d from *nearly any* zero-mean subgaussian distribution. All behave in much the same way.

What is Random Projection? (2)

Different types of RP matrix easy to construct - take entries i.i.d from *nearly any* zero-mean subgaussian distribution. All behave in much the same way.

Popular variations [Ach03, AC06, Mat08]: The entries R_{ij} can be:

What is Random Projection? (2)

Different types of RP matrix easy to construct - take entries i.i.d from *nearly any* zero-mean subgaussian distribution. All behave in much the same way.
Popular variations [Ach03, AC06, Mat08]: The entries R_{ij} can be:

$$R_{ij} = \begin{cases} +1 & \text{w.p. } 1/2, \\ -1 & \text{w.p. } 1/2. \end{cases}$$

What is Random Projection? (2)

Different types of RP matrix easy to construct - take entries i.i.d from *nearly any* zero-mean subgaussian distribution. All behave in much the same way.
Popular variations [Ach03, AC06, Mat08]: The entries R_{ij} can be:

$$R_{ij} = \begin{cases} +1 & \text{w.p. } 1/2, \\ -1 & \text{w.p. } 1/2. \end{cases}$$

$$R_{ij} = \begin{cases} +1 & \text{w.p. } 1/6, \\ -1 & \text{w.p. } 1/6, \\ 0 & \text{w.p. } 2/3. \end{cases}$$

What is Random Projection? (2)

Different types of RP matrix easy to construct - take entries i.i.d from *nearly any* zero-mean subgaussian distribution. All behave in much the same way.
Popular variations [Ach03, AC06, Mat08]: The entries R_{ij} can be:

$$R_{ij} = \begin{cases} +1 & \text{w.p. } 1/2, \\ -1 & \text{w.p. } 1/2. \end{cases} \qquad R_{ij} = \begin{cases} \mathcal{N}(0, 1/q) & \text{w.p. } q, \\ 0 & \text{w.p. } 1-q. \end{cases}$$

$$R_{ij} = \begin{cases} +1 & \text{w.p. } 1/6, \\ -1 & \text{w.p. } 1/6, \\ 0 & \text{w.p. } 2/3. \end{cases}$$

What is Random Projection? (2)

Different types of RP matrix easy to construct - take entries i.i.d from *nearly any* zero-mean subgaussian distribution. All behave in much the same way.
Popular variations [Ach03, AC06, Mat08]: The entries R_{ij} can be:

$$R_{ij} = \begin{cases} +1 & \text{w.p. } 1/2, \\ -1 & \text{w.p. } 1/2. \end{cases} \qquad R_{ij} = \begin{cases} \mathcal{N}(0, 1/q) & \text{w.p. } q, \\ 0 & \text{w.p. } 1-q. \end{cases}$$

$$R_{ij} = \begin{cases} +1 & \text{w.p. } 1/6, \\ -1 & \text{w.p. } 1/6, \\ 0 & \text{w.p. } 2/3. \end{cases} \qquad R_{ij} = \begin{cases} +1 & \text{w.p. } q, \\ -1 & \text{w.p. } q, \\ 0 & \text{w.p. } 1-2q. \end{cases}$$

What is Random Projection? (2)

Different types of RP matrix easy to construct - take entries i.i.d from *nearly any* zero-mean subgaussian distribution. All behave in much the same way.
Popular variations [Ach03, AC06, Mat08]: The entries R_{ij} can be:

$$R_{ij} = \begin{cases} +1 & \text{w.p. } 1/2, \\ -1 & \text{w.p. } 1/2. \end{cases} \qquad R_{ij} = \begin{cases} \mathcal{N}(0, 1/q) & \text{w.p. } q, \\ 0 & \text{w.p. } 1-q. \end{cases}$$

$$R_{ij} = \begin{cases} +1 & \text{w.p. } 1/6, \\ -1 & \text{w.p. } 1/6, \\ 0 & \text{w.p. } 2/3. \end{cases} \qquad R_{ij} = \begin{cases} +1 & \text{w.p. } q, \\ -1 & \text{w.p. } q, \\ 0 & \text{w.p. } 1-2q. \end{cases}$$

For the RH examples, taking q too small gives high distortion of sparse vectors [Mat08]. [AC06] get around this by using a randomized orthogonal (normalized Hadamard) matrix to ensure w.h.p all data vectors are dense.

Fast, sparse variants

Achlioptas ‘01 [Ach03]: $R_{ij} = 0$ w.p. $2/3$

Ailon-Chazelle ‘06 [AC06]: Use $x \longmapsto PHDx$, P random and sparse, $R_{ij} \sim \mathcal{N}(0, 1/q)$ w.p $1/q$, H normalized Hadamard (orthogonal) matrix, $D = \text{diag}(\pm 1)$ random. Mapping takes $\mathcal{O}\left(d \log d + qd\epsilon^{-2} \log N\right)$.

Ailon-Liberty ‘09 [AL09]: Similar construction to [AC06]. $\mathcal{O}\left(d \log k + k^2\right)$.

Dasgupta-Kumar-Sarlós ‘10 [DKS10]: Use sequence of (dependent) random hash functions. $\mathcal{O}\left(\epsilon^{-1} \log^2(k/\delta) \log \delta^{-1}\right)$ for $k \in \mathcal{O}\left(\epsilon^{-2} \log \delta^{-1}\right)$.

Ailon-Liberty ‘11 [AL11]: Similar construction to [AC06]. $\mathcal{O}\left(d \log d\right)$ provided $k \in \mathcal{O}\left(\epsilon^{-2} \log N \log^4 d\right)$.

Generalizations of JLL to Manifolds

From JLL we obtain high-probability guarantees that, independently of the data dimension, random projection approximately preserves data geometry of a finite point set (for a suitably large k). In particular norms and dot products approximately preserved w.h.p.

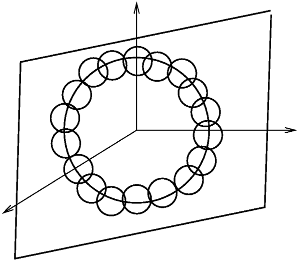
JLL approach can be extended to (smooth, compact) Riemannian manifolds: 'Manifold JLL'

Key idea: Preserve ϵ-covering of smooth manifold under some metric instead of geometry of data points. Replace N with corresponding covering number M and take $k \in \mathcal{O}\left(\epsilon^{-2} \log M\right)$.

Subspace JLL

Let S be an s-dimensional linear subspace. Let $\epsilon > 0$. For $k = \mathcal{O}\left(\epsilon^{-2} s \log(12/\epsilon)\right)$ [BW09] w.h.p. a JLL matrix R satisfies $\forall x, y \in S$: $(1-\epsilon)\|x-y\|_d^2 \leqslant \|Rx - Ry\|_k^2 \leqslant (1+\epsilon)\|x-y\|_d^2$

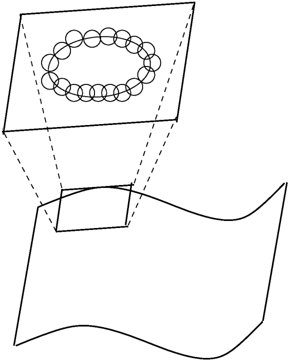
1. R linear, so no loss to take $\|x - y\| = 1$.
2. Cover unit sphere in subspace with $\epsilon/4$-balls. Covering number $M = (12/\epsilon)^s$.
3. Apply JLL to centres of the balls. $k = \mathcal{O}(\log M)$ for this.
4. Extend to entire s-dimensional subspace by approximating any unit vector with one of the centres.



Manifold JLL

Proof idea:

1. Let M be a smooth s-dimensional manifold in $\mathbb{R}^d$ ($\Rightarrow$ M is locally like a linear subspace).
2. Approximate manifold with tangent subspaces.
3. Apply subspace-JLL on each subspace.
4. Union bound over subspaces to preserve large distances.



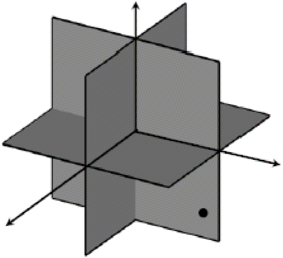
(Same basic approach can be used to preserve geodesic distances.)

JLL for unions of axis-aligned subspaces (‘RIP’)

RIP = Restricted isometry property (more on this later).

Proof idea:

1. Note that s-sparse d-dimensional vectors live on a union of $\binom{d}{s}$ s-dimensional subspaces.
2. Apply subspace-JLL to each s-flat.
3. Apply union bound to all $\binom{d}{s}$ subspaces.



$k = \mathcal{O}\left(\epsilon^{-2} s \log(\frac{12}{\epsilon}\frac{d}{s})\right)$ [BDDW08]

Comment: This is the canonical ‘compressed sensing’ setting assuming the sparse basis is known.

Outline

1 Background and Preliminaries

2 Johnson-Lindenstrauss Lemma (JLL) and extensions

3 Applications of JLL (1)
- Approximate Nearest Neighbour Search
- RP Perceptron
- Mixtures of Gaussians

4 Compressed Sensing

5 Applications of JLL (2)

6 JLL- and CS-free Approaches

Applications of Random Projection (1)

JLL implies that, with a suitable choice of k, we can construct an 'ϵ-approximate' version of *any* algorithm which depends only on the (Euclidean) geometry of the data, but in a much lower-dimensional space. This includes:

- Nearest-neighbour algorithms.
- Clustering algorithms.
- Margin-based classifiers.
- Least-squares regressors.

That is, we trade off some accuracy (perhaps) for reduced algorithmic time and space complexity.

Using one RP...

Diverse motivations for RP in the literature:

- To trade some accuracy in order to reduce computational expense and/or storage overhead (e.g. kNN).
- To create a new theory of cognitive learning (RP Perceptron).
- To replace a heuristic optimizer with a provably correct algorithm with performance guarantees (e.g. Mixture learning).
- To bypass the collection of lots of data then throwing away most of it at preprocessing (Compressed sensing).

Solution in all cases: Work with random projections of the data.

JLL-based approaches

Recall our two initial problem settings:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

For this part of the talk we focus mainly on the first of these problems, i.e. reducing the complexity of large problems.

JLL-based approaches

Recall our two initial problem settings:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

For this part of the talk we focus mainly on the first of these problems, i.e. reducing the complexity of large problems.

JLL-based approaches

Recall our two initial problem settings:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

For this part of the talk we focus mainly on the first of these problems, i.e. reducing the complexity of large problems.

JLL-based approaches

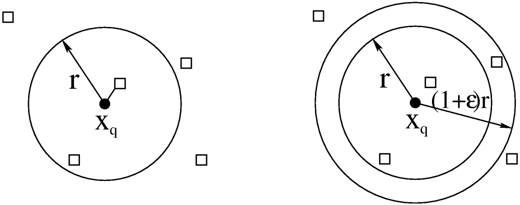
Recall our two initial problem settings:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

For this part of the talk we focus mainly on the first of these problems, i.e. reducing the complexity of large problems.

Approximate Nearest Neighbour Search

- Kept theoreticians busy for over 40 years.
- Many applications: Machine Learning kNN rule; Database retrieval; Data compression (vector quantization).
- Exact Nearest Neighbour Search: Given a point set $\mathcal{T} = \{x_1, ..., x_N\}$ in $\mathbb{R}^d$, find the closest point to a query point x_q.
- Approximate NNS: Find $x \in \mathcal{T}$ that is ϵ-close to x_q. That is, such that $\forall x' \in \mathcal{T}, \|x - x_q\| \leqslant (1+\epsilon)\|x' - x_q\|$.



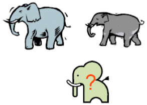
- The problem: Space or time complexity exponential in d even for sophisticated approximate NNS. [Kle97, HP01, AI06].

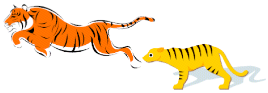
Nearest Neighbour Search

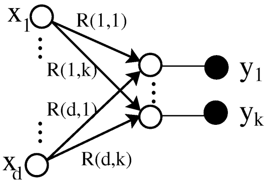
- The first known approximate NNS algorithm with space and time complexity polynomial in d is due to Indyk & Motwani '98[IM98], and uses the JLL.
  - Have an algorithm with query time $\mathcal{O}\exp(d)$.
  - Apply JLL, so take $k \in \mathcal{O}(\epsilon^{-2} \log N)$ and randomly project data.
  - This yields an algorithm that has query time $\mathcal{O}(N^{C\epsilon^{-2}})$.
- Since this important advance, there have been many further results on approximate NNS (including other uses of random projections! e.g. [Cha02]).

Neuronal RP Perceptron Learning

- Motivation: How does the brain learn concepts from a handful of examples when each example contains many features?
- Large margin ⇒ ‘robustness’ of concept.
- Idea:
  1. When the target concept robust, random projection of examples to a low-dimensional subspace preserves the concept.
  2. In the low-dimensional space, the number of examples and time required to learn concepts are comparatively small.







Definition. For any real number $\ell > 0$, a concept in conjunction with a distribution $\mathcal{D}$ on $\mathbb{R}^d \times \{-1, 1\}$, is said to be *robust*, if $\Pr\{x | \exists x' : label(x) \neq label(x'), \text{ and } \|x - x'\| \leqslant \ell\} = 0$.
Given $\mathcal{T} = \{(x_1, y_1), ..., (x_N, y_N)\} \sim \mathcal{D}^N$ labelled training set, $R \in \mathcal{M}_{k \times d}$ a random matrix with zero-mean sub-Gaussian entries.
Suppose $\mathcal{T}$ is a sample from a robust concept, i.e. $\exists h \in \mathbb{R}^d, \|h\| = 1$ s.t. $\forall n \in \{1, ..., N\}, y_n \cdot h^T x_n \geqslant \ell$.

Algorithm

1. Project $\mathcal{T}$ to $\mathcal{T}' = \{(Rx_1, y_1), ..., (Rx_N, y_N)\} \subseteq \mathbb{R}^k$.
2. Learn a perceptron $\hat{h}_R$ in $\mathbb{R}^k$ from $\mathcal{T}'$ (i.e. by minimizing training error).
3. Output R and $\hat{h}_R$.

For a query point x_q predict $sign(\hat{h}_R^T R x_q)$.

PAC Guarantees for RP Perceptron

Using known results on generalization for halfspaces [KV94], and on the running time of Perceptron [MP69] in $\mathbb{R}^k$, we use JLL to ensure their preconditions hold w.h.p.

The results employed in [AV06] require classes to be separated by a margin ℓ in the dataspace and, for the randomly-projected perceptron, they then show that the classes remain $\ell/2$-separated w.h.p and that all norms and dot products are also approximately preserved w.h.p.

Taking L to be the squared diameter of the data and N the number of training examples, a PAC guarantee is obtained provided that:

$$k = \mathcal{O}\left(\frac{L}{\ell^2} \cdot \log(12N/\delta)\right) \quad (1)$$

i.e. take k large enough to ensure all of the above conditions hold w.h.p.

Comment: In [AV06] the authors obtain $k = \mathcal{O}\left(\frac{1}{\ell^2} \cdot \log(\frac{1}{\epsilon\ell\delta})\right)$ by taking the allowed misclassification rate as $\epsilon \geqslant \frac{1}{12N\ell}$. This somewhat obscures effect on upper bound on gen. error of # training points N.

Provably Learning Mixtures of Gaussians

- Mixtures of Gaussians (MoG) are among the most fundamental and widely used statistical models. $p(x) = \sum_{y=1}^{K} \pi_y \mathcal{N}(x|\mu_y, \Sigma_y)$, where $\mathcal{N}(x|\mu_y, \Sigma_y) = \frac{1}{(2\pi)^{d/2}|\Sigma_y|^{1/2}} \exp(-\frac{1}{2}(x - \mu_y)^T \Sigma_y^{-1}(x - \mu_y))$.
- Given a set of unlabelled data points drawn from a MoG, the goal is to estimate the mean μ_y and covariance Σ_y for each source.
- Greedy heuristics (such as Expectation-Maximization) widely used for this purpose do not guarantee correct recovery of mixture parameters (can get stuck in local optima of the likelihood function).
- The first provably correct algorithm to learn a MoG from data is based on random projections.

Algorithm

Inputs: Sample S: set of N data points in $\mathbb{R}^d$; m = number of mixture components; ϵ,δ: resp. accuracy and confidence params. π_{min}: smallest mixture weight to be considered.
(Values for other params derived from these via the theoretical analysis of the algorithm.)

1. Randomly project the data onto a k-dimensional subspace of the original space $\mathbb{R}^d$. Takes time $\mathcal{O}(Nkd)$.
2. In the projected space:
    - For $x \in S$, let r_x be the smallest radius such that there are $\geqslant p$ points within distance r_x of x.
    - Start with $S' = S$.
    - For $y = 1, ..., m$:
        - Let estimate $\hat{\mu}_y^*$ be the point x with the lowest r_x
        - Find the q closest points to this estimated center.
        - Remove these points from S'.
    - For each y, let S_y denote the l points in S which are closest to $\hat{\mu}_y^*$.
3. Let the (high-dimensional) estimate $\hat{\mu}_y$ be the mean of S_y in $\mathbb{R}^d$.

Definition

Two Gaussians $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$ in $\mathbb{R}^d$ are said to be *c-separated* if $\|\mu_1 - \mu_2\| \geqslant c\sqrt{d \cdot \max\{\lambda_{\max}(\Sigma_1), \lambda_{\max}(\Sigma_2)\}}$. A mixture of Gaussians is c-separated if its components are c-separated.

Theorem

Let $\delta, \epsilon \in (0, 1)$. Suppose the data is drawn from a mixture of m Gaussians in $\mathbb{R}^d$ which is c-separated, for $c > 1/2$, has (unknown) common covariance matrix Σ with condition number $\kappa = \lambda_{\max}(\Sigma)/\lambda_{\min}(\Sigma)$, and $\min_y \pi_y = \Omega(1/m)$. Then,

- w.p. $\geqslant 1 - \delta$, the centre estimates returned by the algorithm are accurate within ℓ_2 distance $\epsilon\sqrt{d\lambda_{\max}}$;
- if $\sqrt{\kappa} \leqslant \mathcal{O}(d^{1/2}/\log(m/(\epsilon\delta)))$, then the reduced dimension required is $k = \mathcal{O}(\log m/(\epsilon\delta))$, and the number of data points needed is $N = m^{\mathcal{O}(\log^2(1/(\epsilon\delta)))}$. The algorithm runs in time $\mathcal{O}(N^2k + Nkd)$.

The proof is lengthy but it starts from the following observations:

- A c-separated mixture becomes a $(c \cdot \sqrt{1-\epsilon})$-separated mixture w.p. $1-\delta$ after RP. This is because
  - JLL ensures that the distances between centers are preserved
  - $\lambda_{\max}(R\Sigma R^T) \leqslant \lambda_{\max}(\Sigma)$
- RP makes covariances more spherical (i.e. condition number decreases).

It is worth mentioning that the latest theoretical advances [KMV12] on learning of high dimensional mixture distributions under general conditions (i.e. without the c-separability condition) in polynomial time also use RP.
At present this is a theoretical construction only - no concrete implementation.

10 years later...

Outline

1. Background and Preliminaries
2. Johnson-Lindenstrauss Lemma (JLL) and extensions
3. Applications of JLL (1)
4. Compressed Sensing
   - SVM from RP sparse data
5. Applications of JLL (2)
6. JLL- and CS-free Approaches

Compressed Sensing (1)

Often high-dimensional data is *sparse* in the following sense: There is some representation of the data in a linear basis such that most of the coefficients of the data vectors are (nearly) zero in this basis. For example, image and audio data in e.g. DCT basis.
Sparsity implies compressibility e.g. discarding small DCT coefficients gives us lossy compression techniques such as jpeg and mp3.

Idea: Instead of collecting sparse data and then compressing it to (say) 10% of its former size, what if we just captured 10% of the data in the first place?
In particular, what if we just captured 10% of the data at random? Could we reconstruct the original data?

Compressed (or Compressive) Sensing [Don06, CT06].

Compressed Sensing (2)

Problem: Want to reconstruct sparse d-dimensional signal x, with s non-zero coeffs. in sparse basis, given only k random measurements. i.e. we observe:

$$y = Rx, \; y \in \mathbb{R}^k, \; R \in \mathcal{M}_{k \times d}, \; x \in \mathbb{R}^d, \; k \ll d.$$

and we want to find x given y. Since R is rank $k \ll d$ no unique solution in general.
However we also know that x is s-sparse...

Compressed Sensing (3)

Basis Pursuit Theorem (Candès-Tao 2004)

Let R be a $k \times d$ matrix and s an integer such that:

- $y = Rx$ admits an s-sparse solution $\hat{x}$, i.e. such that $\|\hat{x}\|_0 \leqslant s$.
- R satisfies the restricted isometry property (RIP) of order $(2s, \delta_{2s})$ with $\delta_{2s} \leqslant 2/(3 + \sqrt{7/4}) \simeq 0.4627$

Then:

$$\hat{x} = \underset{x}{\arg\min} \{\|x\|_1 : y = Rx\}$$

- If R and x satisfy the conditions on the BPT, then we can reconstruct x perfectly from its compressed representation, using efficient ℓ_1 minimization methods.
- We know x needs to be s-sparse. Which matrices R then satisfy the RIP?

Restricted Isometry Property

Restricted Isometry Property

Let R be a $k \times d$ matrix and s an integer. The matrix R satisfies the RIP of order (s, δ) provided that, for all s-sparse vectors $x \in \mathbb{R}^d$:

$$(1-\delta)\|x\|_2^2 \leqslant \|Rx\|_2^2 \leqslant (1+\delta)\|x\|_2^2$$

One can show that random projection matrices satisfying the JLL w.h.p also satisfy the RIP w.h.p provided that $k \in \mathcal{O}(s \log d)$. [BDDW08] does this using JLL combined with a covering argument in the projected space, finally union bound over all possible $\binom{d}{s}$ s-dimensional subspaces.

N.B. For signal reconstruction, data must be sparse: no perfect reconstruction guarantee from random projection matrices if $s > d/2$.

Compressed Learning

Intuition: If the data are s-sparse then one can perfectly reconstruct the data w.h.p from its randomly projected representation, provided that $k \in \mathcal{O}(s \log d)$.

It follows that w.h.p no information was lost by carrying out the random projection.

Therefore w.h.p one should be able to construct a classifier (or regressor) from the RP data which generalizes as well as the classifier (or regressor) learned from the original (non-RP) data.

Fast learning of SVM from sparse data

Theorem (Calderbank et al. [CJS09])

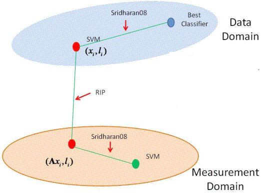
Let R be a $k \times d$ random matrix which satisfies the RIP. Let $RS = \{(Rx_1, y_1), ..., (Rx_N, y_N)\} \sim \mathcal{D}^N$.
Let $\hat{z}_{RS}$ be the soft-margin SVM trained on RS.
Let w_0 be the best linear classifier in the data domain with low hinge loss and large margin (hence small $\|w_0\|$).
Then, w.p. $1 - 2\delta$ (over RS):

$$H_{\mathcal{D}}(\hat{z}_{RS}) \leqslant H_{\mathcal{D}}(w_0) + \mathcal{O}\left(\sqrt{\|w_0\|^2\left(L^2\epsilon + \frac{\log(1/\delta)}{N}\right)}\right) \tag{2}$$

where $H_{\mathcal{D}}(w) = E_{(x,y)\sim\mathcal{D}}[1 - yw^T x]$ is the true hinge loss of the classifier in its argument, and $L = max_n \|x_n\|$.

The proof idea is somewhat analogous to that in Arriaga & Vempala, with several differences:

- Major:
  - Data is assumed to be sparse. This allows using RIP instead of JLL and eliminates the dependence of the required k on the sample size N. Instead it will now depend (linearly) on s.
- Minor:
  - Different classifier
  - The best classifier is not assumed to have zero error



Proof sketch:

- Risk bound in [SSSS08] bounds the true SVM hinge loss of a classifier learned from data from that of the best classifier. Used twice: once in the data space, and again in the projection space.
- By definition (of best classifier), the true error of the best classifier in projected space is smaller than that of the projection of the best classifier in the data space.
- From RIP, derive the preservation of dot-products (similarly as previously in the case of JLL) which is then used to connect between the two spaces.

Outline

1. Background and Preliminaries
2. Johnson-Lindenstrauss Lemma (JLL) and extensions
3. Applications of JLL (1)
4. Compressed Sensing
5. Applications of JLL (2)
   - RP LLS Regression
6. JLL- and CS-free Approaches

Compressive Linear Least Squares Regression

Given $\mathcal{T} = \{(x_1, y_1), ..., (x_N, y_N)\}$ with $x_n \in \mathbb{R}^d$, $y_n \in \mathbb{R}$.
Algorithm.

1. Let R a $k \times d$ RP matrix with entries $R_{ij} \sim \mathcal{N}(0, 1)$, let $P := R/\sqrt{k}$, and project the data: XP^T to $\mathbb{R}^k$.
2. Run a regression method in $\mathbb{R}_k$.

Result. Using JLL in conjunction with bounds on the excess risk of regression estimators with least squares loss, the gap between the true loss of the obtained estimator in the projected space and that of the optimal predictor in the data space can be bounded with high probability, provided that $k \in \mathcal{O}(\frac{8}{\epsilon^2} \log(8N/\delta))$.
For full details see [MM09].
Here we outline the special case of ordinary least squares regression (OLS).

Denote by X the design matrix with x_i its rows, and Y a column vector with elements y_i.
Assume X is fixed (not random), and we want to learn an estimator $\hat{\beta}$ so that $X\hat{\beta}$ approximates $\mathrm{E}[Y|X]$.
Definitions in $\mathbb{R}^d$.
Squared loss: $L(w) = \frac{1}{N}\mathrm{E}_Y[\|Y - Xw\|^2]$ (where E_Y denotes $\mathrm{E}_{Y|X}$).
Optimal predictor: $\beta = \underset{w}{\arg\min} L(w)$.
Excess risk of an estimator: $R(\hat{\beta}) = L(\hat{\beta}) - L(\beta)$.
For linear regression this is: $(\hat{\beta} - \beta)^T\Sigma(\hat{\beta} - \beta)$ where $\Sigma = X^TX/N$.
OLS estimator: $\hat{\beta} := \underset{w}{\arg\min}\frac{1}{N}\|Y - Xw\|^2$

Proposition: OLS. If $\mathrm{Var}(Y_i) \leqslant 1$ then $\mathrm{E}_Y[R(\hat{\beta})] \leqslant \frac{d}{N}$.

Definitions in $\mathbb{R}^k$.
Square loss: $L_P(w) = \frac{1}{N}\mathrm{E}_Y[\|Y - (XP^T)w\|^2]$ (where E_Y denotes $\mathrm{E}_{Y|X}$).
Optimal predictor: $\beta_P = \arg\min_w L_P(w)$.
RP-OLS estimator: $\hat{\beta}_P := \arg\min_w \frac{1}{N}\|Y - (XP^T)w\|^2$

Proposition: Risk bound for RP-OLS
Assume $\mathrm{Var}(Y_i) \leqslant 1$, and let P as defined earlier. Then, for $k = \mathcal{O}(\log(8N/\delta)/\epsilon^2$ and any $\epsilon, \delta > 0$, w.p. $1 - \delta$ we have:

$$\mathrm{E}_Y[L_P(\hat{\beta}_P)] - L(\beta) \leqslant \frac{k}{N} + \|\beta\|^2\|\Sigma\|_{\text{trace}}\epsilon^2 \qquad (3)$$

Comment: The argument in [MM09] uses JLL to obtain the excess risk guarantee. Value of k obtained is suboptimal - log N term can be removed (for an extended class of random matrices - including non-RIP/non-JLL matrices) using a more careful treatment - see [Kab14] for details.

Outline

1. Background and Preliminaries
2. Johnson-Lindenstrauss Lemma (JLL) and extensions
3. Applications of JLL (1)
4. Compressed Sensing
5. Applications of JLL (2)
6. JLL- and CS-free Approaches
    - RP Fisher's Linear Discriminant
        - Time Complexity Analysis
        - Experiments
    - RP Estimation of Distribution Algorithm

Further approaches not leveraging JLL or CS

Recall our two initial problem settings:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

For the remainder we will focus on the second problem, i.e. good inference from next to no data.

Further approaches not leveraging JLL or CS

Recall our two initial problem settings:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

For the remainder we will focus on the second problem, i.e. good inference from next to no data.

Further approaches not leveraging JLL or CS

Recall our two initial problem settings:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

For the remainder we will focus on the second problem, i.e. good inference from next to no data.

Further approaches not leveraging JLL or CS

Recall our two initial problem settings:

- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and very many observations ($N \in \mathcal{O}(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in \mathcal{O}(1000)$) and hardly any observations ($N \in \mathcal{O}(10)$): Inference a hard problem. Bogus interactions between features.

For the remainder we will focus on the second problem, i.e. good inference from next to no data.

Some intuition and an idea

- Consider JLL again: If we have hardly any observations, $N \in \mathcal{O}(\log d)$ say then JLL would only require $k \in \mathcal{O}(\log \log d)$ . . .
- . . . so then $N \in \mathcal{O}(\exp(k))$! Q: Could better parameter estimates in the projected space compensate for the distortion introduced by RP?
- Somewhat appealing but potentially a *serious problem*, at least from a statistical perspective, is that we now seem to estimate the parameters for the wrong distribution.

Let's run with it for a while anyhow. . . ☺

Some intuition and an idea

- Consider JLL again: If we have hardly any observations, $N \in \mathcal{O}(\log d)$ say then JLL would only require $k \in \mathcal{O}(\log \log d) \ldots$
- $\ldots$ so then $N \in \mathcal{O}(\exp(k))$! Q: Could better parameter estimates in the projected space compensate for the distortion introduced by RP?
- Somewhat appealing but potentially a *serious problem*, at least from a statistical perspective, is that we now seem to estimate the parameters for the wrong distribution.

Let's run with it for a while anyhow… ☺

Some intuition and an idea

- Consider JLL again: If we have hardly any observations, $N \in \mathcal{O}(\log d)$ say then JLL would only require $k \in \mathcal{O}(\log \log d) \ldots$
- ... so then $N \in \mathcal{O}(\exp(k))$! Q: Could better parameter estimates in the projected space compensate for the distortion introduced by RP?
- Somewhat appealing but potentially a *serious problem*, at least from a statistical perspective, is that we now seem to estimate the parameters for the wrong distribution.

Let's run with it for a while anyhow... ☺

Some intuition and an idea

- Consider JLL again: If we have hardly any observations, $N \in \mathcal{O}(\log d)$ say then JLL would only require $k \in \mathcal{O}(\log \log d) \ldots$
- $\ldots$ so then $N \in \mathcal{O}(\exp(k))$! Q: Could better parameter estimates in the projected space compensate for the distortion introduced by RP?
- Somewhat appealing but potentially a *serious problem*, at least from a statistical perspective, is that we now seem to estimate the parameters for the wrong distribution.

Let's run with it for a while anyhow. . . ☺

Some intuition and an idea

- Consider JLL again: If we have hardly any observations, $N \in \mathcal{O}(\log d)$ say then JLL would only require $k \in \mathcal{O}(\log \log d) \ldots$
- $\ldots$ so then $N \in \mathcal{O}(\exp(k))$! Q: Could better parameter estimates in the projected space compensate for the distortion introduced by RP?
- Somewhat appealing but potentially a *serious problem*, at least from a statistical perspective, is that we now seem to estimate the parameters for the wrong distribution.

Let's run with it for a while anyhow. . . ☺

Guarantees without JLL

We can obtain guarantees for randomly-projected learning algorithms *without* directly applying the JLL, by using measure concentration and random matrix theoretic-based approaches. One application of such an approach is the work on OLS by [Kab14]; here we will consider classification and real-valued optimization.

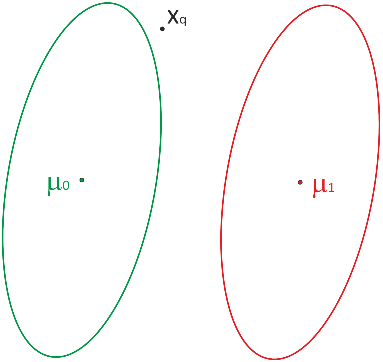
Key Intuition: Projecting from a high-dimensional to a low dimensional setting turns an ill-posed problem (i.e. with no unique solution) into a well-posed one (there is a unique solution).

Therefore we can view random projection *as a way of regularizing the original problem*. But what kind of regularization? Can we say more? Most importantly, can we interpret the problem solved in the projected space in terms of the original problem?

Applications (1)
Classification with Fisher's Linear Discriminant

Fisher's Linear Discriminant (FLD)

- Supervised learning approach.
- Simple and popular linear classifier, in widespread application.
- Classes are modelled as MVN distributions with same covariance.
- Assign class label to query point according to Mahalanobis distance from class means.
- FLD decision rule:



$$\hat{h}(\mathbf{x}_q) = \mathbf{1}\left\{(\hat{\mu}_1 - \hat{\mu}_0)^T \hat{\Sigma}^{-1}\left(\mathbf{x}_q - \frac{(\hat{\mu}_0 + \hat{\mu}_1)}{2}\right) > 0\right\}$$

Randomly-projected Fisher's Linear Discriminant

'RP-FLD': Learn the classifier and carry out the classification in the RP space.

Things we don't need to worry about:

- Important points lying in the null space of R: Happens with probability 0.
- Problems mapping means, covariances in data space to RP space: All well-defined due to linearity of R and $\mathrm{E}[\cdot]$.

RP-FLD decision rule:

$$\hat{h}_R(\mathbf{x}_q) = \mathbf{1}\left\{(\hat{\mu}_1 - \hat{\mu}_0)^T R^T \left(R\hat{\Sigma}R^T\right)^{-1} R\left(\mathbf{x}_q - \frac{(\hat{\mu}_0 + \hat{\mu}_1)}{2}\right) > 0\right\}$$

Randomly-projected Fisher's Linear Discriminant

'RP-FLD': Learn the classifier and carry out the classification in the RP space.

Things we don't need to worry about:

- Important points lying in the null space of R: Happens with probability 0.
- Problems mapping means, covariances in data space to RP space: All well-defined due to linearity of R and $\mathrm{E}[\cdot]$.

RP-FLD decision rule:

$$\hat{h}_R(\mathbf{x}_q) = \mathbf{1}\left\{(\hat{\mu}_1 - \hat{\mu}_0)^T R^T \left(R\hat{\Sigma}R^T\right)^{-1} R\left(\mathbf{x}_q - \frac{(\hat{\mu}_0 + \hat{\mu}_1)}{2}\right) > 0\right\}$$

Randomly-projected Fisher's Linear Discriminant

'RP-FLD': Learn the classifier and carry out the classification in the RP space.

Things we don't need to worry about:

- Important points lying in the null space of R: Happens with probability 0.
- Problems mapping means, covariances in data space to RP space: All well-defined due to linearity of R and $\mathrm{E}[\cdot]$.

RP-FLD decision rule:

$$\hat{h}_R(\mathbf{x}_q) = \mathbf{1}\left\{(\hat{\mu}_1 - \hat{\mu}_0)^T R^T \left(R\hat{\Sigma}R^T\right)^{-1} R\left(\mathbf{x}_q - \frac{(\hat{\mu}_0 + \hat{\mu}_1)}{2}\right) > 0\right\}$$

Randomly-projected Fisher's Linear Discriminant

'RP-FLD': Learn the classifier and carry out the classification in the RP space.

Things we don't need to worry about:

- Important points lying in the null space of R: Happens with probability 0.
- Problems mapping means, covariances in data space to RP space: All well-defined due to linearity of R and $\mathrm{E}[\cdot]$.

RP-FLD decision rule:

$$\hat{h}_R(\mathbf{x}_q) = \mathbf{1}\left\{(\hat{\mu}_1 - \hat{\mu}_0)^T R^T \left(R\hat{\Sigma}R^T\right)^{-1} R\left(\mathbf{x}_q - \frac{(\hat{\mu}_0 + \hat{\mu}_1)}{2}\right) > 0\right\}$$

Randomly-projected Fisher's Linear Discriminant

'RP-FLD': Learn the classifier and carry out the classification in the RP space.

Things we don't need to worry about:

- Important points lying in the null space of R: Happens with probability 0.
- Problems mapping means, covariances in data space to RP space: All well-defined due to linearity of R and $\mathrm{E}[\cdot]$.

RP-FLD decision rule:

$$\hat{h}_R(\mathbf{x}_q) = \mathbf{1}\left\{(\hat{\mu}_1 - \hat{\mu}_0)^T R^T \left(R\hat{\Sigma}R^T\right)^{-1} R\left(\mathbf{x}_q - \frac{(\hat{\mu}_0 + \hat{\mu}_1)}{2}\right) > 0\right\}$$

The Problem

Assume a two-class classification problem, with N real-valued d-dimensional training observations:
$\mathcal{T} = \{(\mathbf{x}_i, y_i) : (\mathbf{x}, y) \in \mathbb{R}^d \times \{0, 1\}\}_{i=1}^N$.

Furthermore assume that $N \ll d$, which is a common situation in practice (e.g. medical imaging, genomics, proteomics, face recognition, etc.), and WLOG that the unknown data distribution is full rank i.e. rank$(\Sigma) = d$. (Can relax to rank$(\Sigma) >$ rank$(\hat{\Sigma})$.)

Challenges (1)

Problems:
Issues: N is too small (for good estimation of model parameters) w.r.t $d \iff d$ is too large w.r.t N.
$\hat{\Sigma}$ (but not Σ) is singular.
$\hat{\Sigma}$ must be inverted to construct FLD classifier.

Solution: Compress data by random projection to $\mathbb{R}^k$, $k \leqslant \text{rank}\hat{\Sigma} =: \rho$.

Challenges (2)

It is known that, for a single randomly-projected (linear) classifier, expected misclassification error (w.r.t R) grows nearly exponentially as $k \searrow 1$ [DK13].

Solution:
Recover performance using an ensemble of RP FLD classifiers [DK14].

Ensembles that use some form of randomization in the design of the base classifiers have a long and successful history in machine learning: E.g. bagging [Bre96]; random subspaces [Ho98]; random forests [Bre01]; random projection ensembles [FB03, GBN05].

Comment: We also obtain substantial computational savings with our approach – details later.

Our Questions

- Can we recover (or improve on) the level of classification performance achieved in the data space, using our RP FLD ensemble?
- Can we understand how the RP FLD ensemble acts to improve performance?
- Can we overfit the data with too large an RP-FLD ensemble?
- Can we interpret the RP ensemble classifier parameters in terms of data space parameters?

Our Questions

- Can we recover (or improve on) the level of classification performance achieved in the data space, using our RP FLD ensemble?
- Can we understand how the RP FLD ensemble acts to improve performance?
- Can we overfit the data with too large an RP-FLD ensemble?
- Can we interpret the RP ensemble classifier parameters in terms of data space parameters?

Our Questions

- Can we recover (or improve on) the level of classification performance achieved in the data space, using our RP FLD ensemble?
- Can we understand how the RP FLD ensemble acts to improve performance?
- Can we overfit the data with too large an RP-FLD ensemble?
- Can we interpret the RP ensemble classifier parameters in terms of data space parameters?

Our Questions

- Can we recover (or improve on) the level of classification performance achieved in the data space, using our RP FLD ensemble?
- Can we understand how the RP FLD ensemble acts to improve performance?
- Can we overfit the data with too large an RP-FLD ensemble?
- Can we interpret the RP ensemble classifier parameters in terms of data space parameters?

Our Questions

- Can we recover (or improve on) the level of classification performance achieved in the data space, using our RP FLD ensemble?
- Can we understand how the RP FLD ensemble acts to improve performance?
- Can we overfit the data with too large an RP-FLD ensemble?
- Can we interpret the RP ensemble classifier parameters in terms of data space parameters?

RP FLD Classifier Ensemble

For a single RP FLD classifier, the decision rule is given by:

$$\mathbf{1}\left\{(\hat{\mu}_1 - \hat{\mu}_0)^T R^T \left(R\hat{\Sigma}R^T\right)^{-1} R \left(\mathbf{x}_q - \frac{\hat{\mu}_1 + \hat{\mu}_0}{2}\right) > 0\right\}$$

which is the randomly projected analogue of the FLD decision rule. For the ensemble we use an equally weighted linear combination of RP FLD classifiers:

$$\mathbf{1}\left\{\frac{1}{M}\sum_{i=1}^{M}(\hat{\mu}_1 - \hat{\mu}_0)^T R_i^T \left(R_i\hat{\Sigma}R_i^T\right)^{-1} R_i \left(\mathbf{x}_q - \frac{\hat{\mu}_1 + \hat{\mu}_0}{2}\right) > 0\right\} \quad (4)$$

Linear combination rules are a common choice for ensembles. This rule works well in practice and it is also tractable to analysis.

Observation

We can rewrite decision rule as:

$$\mathbf{1}\left\{(\hat{\mu}_1 - \hat{\mu}_0)^T \frac{1}{M}\sum_{i=1}^{M} R_i^T \left(R_i \hat{\Sigma} R_i^T\right)^{-1} R_i \left(\mathbf{x}_q - \frac{\hat{\mu}_1 + \hat{\mu}_0}{2}\right) > 0\right\}$$

Then, for average case analysis with a *fixed* training set, it is enough to consider:

$$\lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M} R_i^T \left(R_i \hat{\Sigma} R_i^T\right)^{-1} R_i \; = \; \mathrm{E}\left[R^T \left(R \hat{\Sigma} R^T\right)^{-1} R\right]$$

(Provided this limit exists).

Theory(1):Regularization

For a fixed training set ρ, d are constant and $1 \leqslant k \leqslant \rho$ is the integer regularization parameter (which we can choose).

Our analysis reveals that the (implicit) regularization scheme implemented by the ensemble has a particularly pleasing form. In particular our ensemble implements:

- Shrinkage regularization [LW04] in range of $\hat{\Sigma}$.
- Ridge regularization [HTF01] in null space of $\hat{\Sigma}$.

As $k \nearrow \rho - 1$ we regularize less, and approach the performance of pseudo-inverted FLD (i.e. we start to overfit).
As $k \searrow 1$ we regularize more – very small choices of k typically underfit.
Sweet spot seems to be around $k = \rho/2$.

Theory(2):Exact Error of Converged Ensemble

Theorem (Ensemble error with Gaussian classes)

Let $\mathbf{x}_q \sim \sum_{y=0}^{1} \pi_y \mathcal{N}(\mu_y, \Sigma)$*, where* $\Sigma \in \mathcal{M}_{d\times d}$ *is a full rank covariance matrix. Let* $R \in \mathcal{M}_{k\times d}$ *be a random projection matrix with i.i.d. Gaussian entries and denote* $S_R^{-1} := E_R\left[R^T\left(R\hat{\Sigma}R^T\right)^{-1}R\right]$*. Then the exact error of the randomly projected ensemble classifier* (4)*, conditioned on the training set, is given by:*

$$\sum_{y=0}^{1} \pi_y \Phi\left(-\frac{1}{2}\frac{(\hat{\mu}_{\neg y}-\hat{\mu}_y)^T S_R^{-1}(\hat{\mu}_0+\hat{\mu}_1-2\mu_y)}{\sqrt{(\hat{\mu}_1-\hat{\mu}_0)^T S_R^{-1}\Sigma S_R^{-1}(\hat{\mu}_1-\hat{\mu}_0)}}\right)$$

Comment: Generalization error is monotonic increasing in $\kappa(S_R^{-1/2}\Sigma S_R^{-1/2})$ - this condition number is bounded a.s. for a large enough ensemble, but not for pseudo-inverted FLD – cf.[BL04].

Theory (3): Finite sample guarantee for ensemble

Theorem (Generalization Error)

Let $\mathcal{T}_N = \{(x_i, y_i)\}_{i=1}^{N}$ be a set of training data of size $N = N_0 + N_1$, subject to $N < d$ and $N_y > 1$ $\forall y$. Let $\mathbf{x}_q$ be a query point with Gaussian class-conditionals $\mathbf{x}_q | y_q \sim \mathcal{N}(\mu_y, \Sigma)$, and let $\Pr\{y_q = y\} = \pi_y$. *Let ρ be the rank of the maximum likelihood estimate of the covariance matrix and let $k < \rho - 1$ be a positive integer. Then for any $\delta \in (0, 1)$ and any training set of size N, the generalization error of the converged ensemble is upper-bounded w.p. at least $1 - \delta$ by:*

$$\Pr_{\mathbf{x}_q, y_q}(\hat{h}_{ens}(\mathbf{x}_q) \neq y_q) \leqslant \sum_{y=0}^{1} \pi_y \Phi\left(-\left[g\left(\bar{\kappa}\left(\sqrt{2\log\frac{5}{\delta}}\right)\right) \times \ldots\right.\right.$$

$$\left.\left.\ldots \left[\sqrt{\|\Sigma^{-\frac{1}{2}}(\mu_1 - \mu_0)\|^2 + \frac{dN}{N_0 N_1}} - \sqrt{\frac{2N}{N_0 N_1}\log\frac{5}{\delta}}\right]_+ - \sqrt{\frac{d}{N_y}}\left(1 + \sqrt{\frac{2}{d}\log\frac{5}{\delta}}\right)\right]\right)$$

where $\bar{\kappa}(\epsilon)$ is a high probability (w.r.t draws of $\mathcal{T}_N$) upper bound on the condition number of $\Sigma\hat{S}^{-1}$ and $g(\cdot)$ is the function $g(a) := \frac{\sqrt{a}}{1+a}$.

Comment

The principal terms in this bound are:

1. The function $g : [1, \infty) \to (0, \frac{1}{2}]$ which is a decreasing function of its argument and here captures the effect of the mismatch between the estimated model covariance matrix $\hat{S}^{-1}$ and the true class-conditional covariance Σ, via a high-probability upper bound on the condition number of $\hat{S}^{-1}\Sigma$;
2. The Mahalanobis distance between the two class centres which captures the fact that the better separated the classes are the smaller the generalization error should be;
3. Antagonistic terms involving the sample size (N) and the number of training examples in each class (N_0, N_1), capturing the effect of class imbalance - the more balanced the classes the tighter the bound.

Time Complexity Analysis

Time Complexity Analysis (1)

Learning:
In the data space, the cost of learning the FLD classifier is dominated by inverting $\hat{\Sigma}$, which is $\mathcal{O}\left(d^3\right)$ using Gauss-Jordan or $\mathcal{O}\left(d^{2.807}\right)$ using Strassen's algorithm.

For the RP ensemble the main cost comes from projecting the data (M $k \times d$ matrix multiplications) plus M matrix inversions each $\mathcal{O}\left(k^3\right)$.
Taking $M \in \mathcal{O}\left(\left\lceil \frac{d}{k} \right\rceil\right)$ so that the ensemble covariance is full rank w.p. 1, the time complexity *on a single core* is then:

$$\mathcal{O}\left(MkdN + Mk^3\right) = \mathcal{O}\left(d^2N + dk^2\right) \ll \mathcal{O}\left(d^3\right) \text{ when } N, k \ll d$$

N.B. Parallel implementation of the ensemble straightforward, sparse RP matrices improve hidden constants considerably.
Comment: Pseudo-inverse FLD is $\mathcal{O}\left(Nd^2\right)$ time complexity, diagonal FLD $\mathcal{O}\left(d\right)$. Theory and experiments show classification performance of these approaches can be very poor.

Time Complexity Analysis (2)

Classification:

Can avoid randomly projecting a query point M times by averaging the individual classifier decision rules. Bracket the argument to the ensemble decision rule as:

$$\left((\hat{\mu}_1 - \hat{\mu}_0)^T \frac{1}{M} \sum_{i=1}^{M} R_i^T \left(R_i \hat{\Sigma} R_i^T \right)^{-1} R_i \right) \left(\mathbf{x}_q - \frac{\hat{\mu}_1 + \hat{\mu}_0}{2} \right)$$

to obtain a single linear classifier of the form $\hat{h} = w + b$, $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, where:

$$w := \frac{1}{M} \sum_{i=1}^{M} (\hat{\mu}_1 - \hat{\mu}_0)^T R_i^T \left(R_i \hat{\Sigma} R_i^T \right)^{-1} R_i = \frac{1}{M} \sum_{i=1}^{M} w_i$$

and

$$b := -\frac{1}{M} \sum_{i=1}^{M} (\hat{\mu}_1 - \hat{\mu}_0)^T R_i^T \left(R_i \hat{\Sigma} R_i^T \right)^{-1} R_i \left(\frac{\hat{\mu}_1 + \hat{\mu}_0}{2} \right) = \frac{1}{M} \sum_{i=1}^{M} b_i$$

Complexity of classification then $\mathcal{O}(d)$ - same as data space FLD.

Experiments

Experiments: Datasets

Table: Datasets

Name	Source	Sample size	# features
colon	Alon et al. [ABN+99]	62	2000
leukaemia	Golub et al. [GST+99]	72	3571
leukaemia lge	Golub et al. [GST+99]	72	7129
prostate	Singh et al. [SFR+02]	102	6033
duke	West et al. [WBD+01]	44	7129
dorothea	NIPS 2003 [GGBHD03]	800	100000

First five datasets are real-valued, Dorothea is binary and very sparse.

Experiments: Protocol

- Standardized features to have mean 0 and variance 1. For gene arrays ran experiments on 100 independent splits, for Dorothea used single (competition) split.
- For gene arrays, in each split took 12 points for testing, rest for training. For Dorothea 800 points for training, 350 for testing.
- For data space experiments on colon and leukemia used ridge-regularized FLD and fitted regularization parameter using 5-fold CV independently on each split, search in $\{2^{-11}, 2^{-10}, \ldots, 2\}$.
- For other gene array datasets we used diagonal FLD in the data space (size, no sig. diff. in error on colon, leuk.). For Dorothea used Bernoulli NB without preprocessing the binary data.
- Compared performance with SVM with linear kernel and parameter C fitted by 5-fold CV, search in $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$, also compared with ℓ_1-regularized SVM.
- RP base learners: FLDs with full covariance and no regularization.

Experiments: Protocol

- Standardized features to have mean 0 and variance 1. For gene arrays ran experiments on 100 independent splits, for Dorothea used single (competition) split.
- For gene arrays, in each split took 12 points for testing, rest for training. For Dorothea 800 points for training, 350 for testing.
- For data space experiments on colon and leukemia used ridge-regularized FLD and fitted regularization parameter using 5-fold CV independently on each split, search in $\{2^{-11}, 2^{-10}, \ldots, 2\}$.
- For other gene array datasets we used diagonal FLD in the data space (size, no sig. diff. in error on colon, leuk.). For Dorothea used Bernoulli NB without preprocessing the binary data.
- Compared performance with SVM with linear kernel and parameter C fitted by 5-fold CV, search in $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$, also compared with ℓ_1-regularized SVM.
- RP base learners: FLDs with full covariance and no regularization.

Experiments: Protocol

- Standardized features to have mean 0 and variance 1. For gene arrays ran experiments on 100 independent splits, for Dorothea used single (competition) split.
- For gene arrays, in each split took 12 points for testing, rest for training. For Dorothea 800 points for training, 350 for testing.
- For data space experiments on colon and leukemia used ridge-regularized FLD and fitted regularization parameter using 5-fold CV independently on each split, search in $\{2^{-11}, 2^{-10}, \ldots, 2\}$.
- For other gene array datasets we used diagonal FLD in the data space (size, no sig. diff. in error on colon, leuk.). For Dorothea used Bernoulli NB without preprocessing the binary data.
- Compared performance with SVM with linear kernel and parameter C fitted by 5-fold CV, search in $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$, also compared with ℓ_1-regularized SVM.
- RP base learners: FLDs with full covariance and no regularization.

Experiments: Protocol

- Standardized features to have mean 0 and variance 1. For gene arrays ran experiments on 100 independent splits, for Dorothea used single (competition) split.
- For gene arrays, in each split took 12 points for testing, rest for training. For Dorothea 800 points for training, 350 for testing.
- For data space experiments on colon and leukemia used ridge-regularized FLD and fitted regularization parameter using 5-fold CV independently on each split, search in $\{2^{-11}, 2^{-10}, \ldots, 2\}$.
- For other gene array datasets we used diagonal FLD in the data space (size, no sig. diff. in error on colon, leuk.). For Dorothea used Bernoulli NB without preprocessing the binary data.
- Compared performance with SVM with linear kernel and parameter C fitted by 5-fold CV, search in $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$, also compared with ℓ_1-regularized SVM.
- RP base learners: FLDs with full covariance and no regularization.

Experiments: Protocol

- Standardized features to have mean 0 and variance 1. For gene arrays ran experiments on 100 independent splits, for Dorothea used single (competition) split.
- For gene arrays, in each split took 12 points for testing, rest for training. For Dorothea 800 points for training, 350 for testing.
- For data space experiments on colon and leukemia used ridge-regularized FLD and fitted regularization parameter using 5-fold CV independently on each split, search in $\{2^{-11}, 2^{-10}, \ldots, 2\}$.
- For other gene array datasets we used diagonal FLD in the data space (size, no sig. diff. in error on colon, leuk.). For Dorothea used Bernoulli NB without preprocessing the binary data.
- Compared performance with SVM with linear kernel and parameter C fitted by 5-fold CV, search in $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$, also compared with ℓ_1-regularized SVM.
- RP base learners: FLDs with full covariance and no regularization.

Experiments: Protocol

- Standardized features to have mean 0 and variance 1. For gene arrays ran experiments on 100 independent splits, for Dorothea used single (competition) split.
- For gene arrays, in each split took 12 points for testing, rest for training. For Dorothea 800 points for training, 350 for testing.
- For data space experiments on colon and leukemia used ridge-regularized FLD and fitted regularization parameter using 5-fold CV independently on each split, search in $\{2^{-11}, 2^{-10}, \ldots, 2\}$.
- For other gene array datasets we used diagonal FLD in the data space (size, no sig. diff. in error on colon, leuk.). For Dorothea used Bernoulli NB without preprocessing the binary data.
- Compared performance with SVM with linear kernel and parameter C fitted by 5-fold CV, search in $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$, also compared with ℓ_1-regularized SVM.
- RP base learners: FLDs with full covariance and no regularization.

Experiments: Protocol

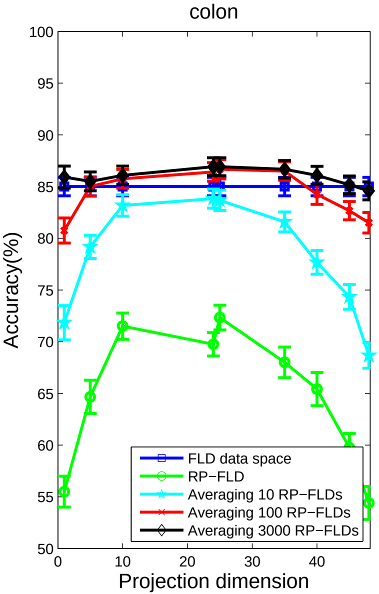
- Standardized features to have mean 0 and variance 1. For gene arrays ran experiments on 100 independent splits, for Dorothea used single (competition) split.
- For gene arrays, in each split took 12 points for testing, rest for training. For Dorothea 800 points for training, 350 for testing.
- For data space experiments on colon and leukemia used ridge-regularized FLD and fitted regularization parameter using 5-fold CV independently on each split, search in $\{2^{-11}, 2^{-10}, \ldots, 2\}$.
- For other gene array datasets we used diagonal FLD in the data space (size, no sig. diff. in error on colon, leuk.). For Dorothea used Bernoulli NB without preprocessing the binary data.
- Compared performance with SVM with linear kernel and parameter C fitted by 5-fold CV, search in $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$, also compared with ℓ_1-regularized SVM.
- RP base learners: FLDs with full covariance and no regularization.

Experiments: Results for $k = \rho/2$

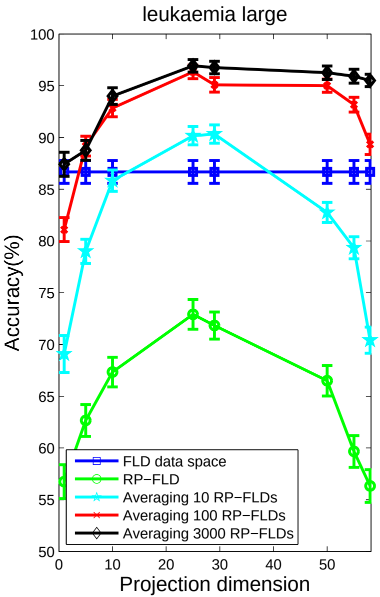
Table: Mean test error rates ± 1 standard error, estimated from 100 independent splits with $k = \rho/2$. (Dorothea: single data split, error rates from 10 independent RP-FLD ensembles).

Dataset	$\rho/2$	100 RP-FLD	1000 RP-FLD	SVM
colon	24	13.58 ± 0.89	13.08 ± 0.86	16.58 ± 0.95
leuk.	29	1.83 ± 0.36	1.83 ± 0.37	1.67 ± 0.36
leuk.lg.	29	4.91 ± 0.70	3.25 ± 0.60	3.50 ± 0.46
prost.	44	8.00 ± 0.76	8.00 ± 0.72	8.00 ± 0.72
duke	15	17.41 ± 1.27	16.58 ± 1.27	13.50 ± 1.10
dorothea	399	8.66 ± 0.044	8.80 ± 0.038	86.86 (33.43 NB)

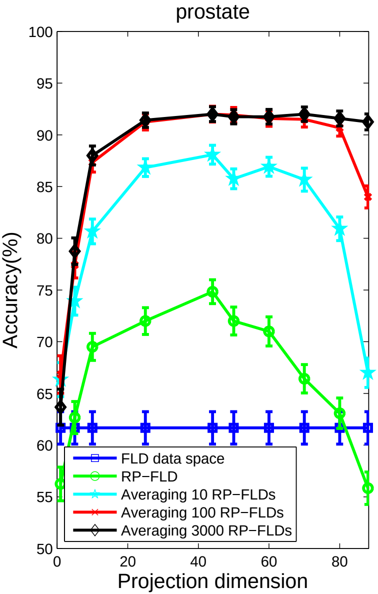
Experiments: Colon Cancer, $R_{ij} \sim \mathcal{N}(0, 1)$



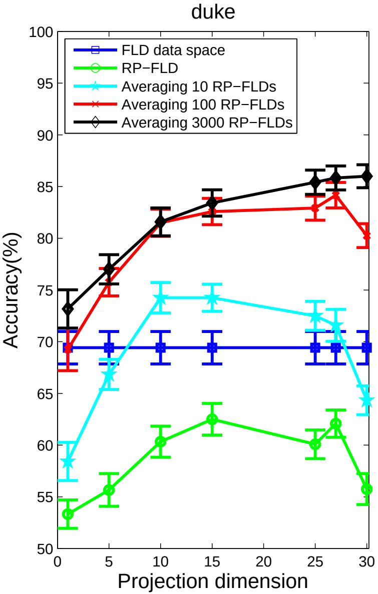
Effect of k: Leukemia Large, $R_{ij} \sim \mathcal{N}(0,1)$



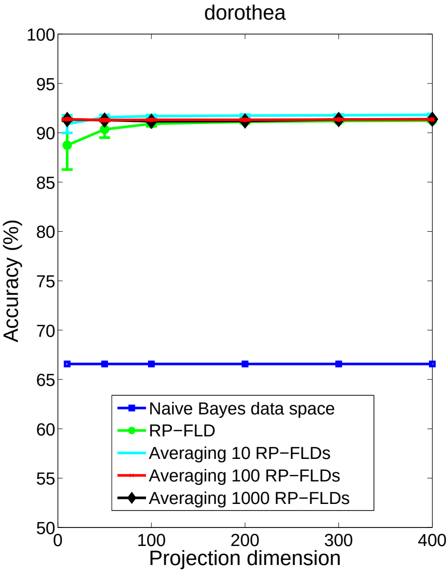
Experiments: Prostate Cancer, $R_{ij} \sim \mathcal{N}(0,1)$



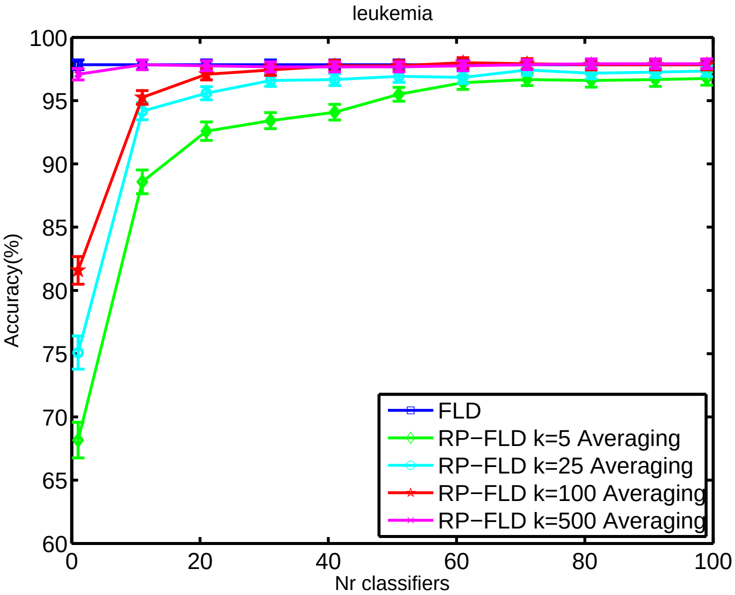
Experiments: Duke Breast Cancer, $R_{ij} \sim \mathcal{N}(0,1)$



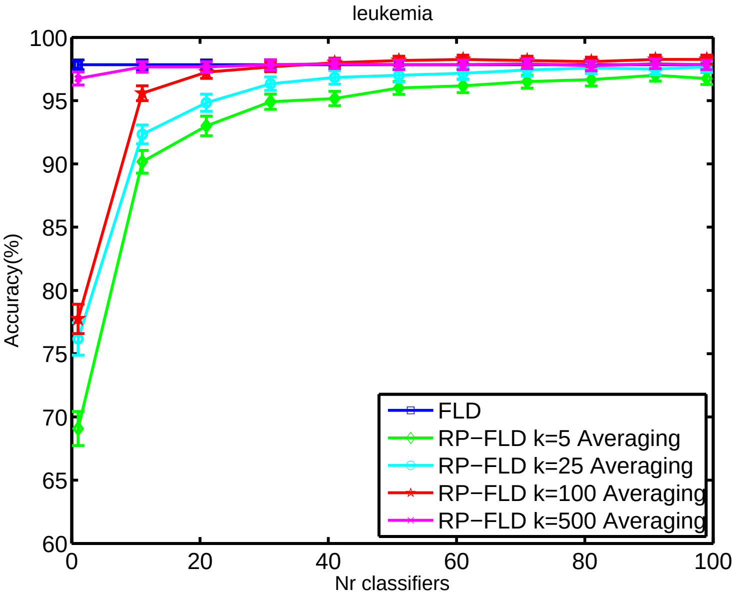
Effect of k: Dorothea, $R_{ij} \sim \mathcal{N}(0, 1)$



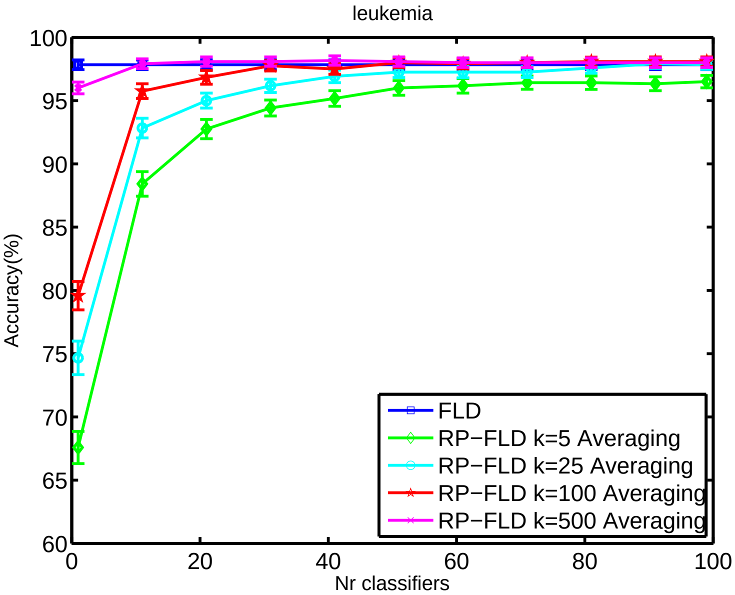
Effect of M: Leukemia, $R_{ij} \sim \mathcal{N}(0,1)$



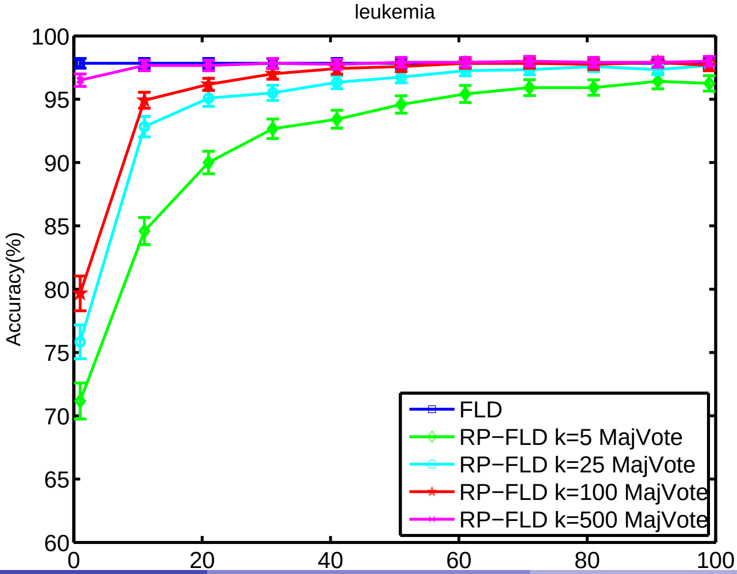
Bernoulli RP Matrix: Leukemia, $R_{ij} \sim \{-1, +1\}$



Sparse RP Matrix: Leukemia, $R_{ij} \sim \{-1, 0, +1\}$



Comparison with Majority Vote: Leukemia, $R_{ij} \sim \mathcal{N}(0,1)$



Answers from Ensembles of RP-FLD

- Can we recover (or improve on) level of classification performance in data space, using the RP FLD ensemble? **YES**
- Can we understand how the RP FLD ensemble acts to improve performance? **YES**
- Can we overfit the data with the RP FLD ensemble? **NO (with appropriate choice of *k*)**
- Can we interpret the ensemble classifier parameters in terms of data space parameters? **YES**

Answers from Ensembles of RP-FLD

- Can we recover (or improve on) level of classification performance in data space, using the RP FLD ensemble? **YES**
- Can we understand how the RP FLD ensemble acts to improve performance? **YES**
- Can we overfit the data with the RP FLD ensemble? **NO (with appropriate choice of k)**
- Can we interpret the ensemble classifier parameters in terms of data space parameters? **YES**

Answers from Ensembles of RP-FLD

- Can we recover (or improve on) level of classification performance in data space, using the RP FLD ensemble? **YES**
- Can we understand how the RP FLD ensemble acts to improve performance? **YES**
- Can we overfit the data with the RP FLD ensemble? **NO (with appropriate choice of k)**
- Can we interpret the ensemble classifier parameters in terms of data space parameters? **YES**

Answers from Ensembles of RP-FLD

- Can we recover (or improve on) level of classification performance in data space, using the RP FLD ensemble? **YES**
- Can we understand how the RP FLD ensemble acts to improve performance? **YES**
- Can we overfit the data with the RP FLD ensemble? **NO (with appropriate choice of k)**
- Can we interpret the ensemble classifier parameters in terms of data space parameters? **YES**

Answers from Ensembles of RP-FLD

- Can we recover (or improve on) level of classification performance in data space, using the RP FLD ensemble? **YES**
- Can we understand how the RP FLD ensemble acts to improve performance? **YES**
- Can we overfit the data with the RP FLD ensemble? **NO (with appropriate choice of** k**)**
- Can we interpret the ensemble classifier parameters in terms of data space parameters? **YES**

Applications (2)
Large-scale Unconstrained Continuous Optimization using EDA

Estimation of Distribution Algorithm

EDA is a state-of-the art heuristic optimization scheme for unconstrained real optimization problems (in low-dimensional settings). Evolutionary algorithm-like approach - the main loop runs until stopping criteria are met, and looks like:

1. Evaluate the objective function on a population of N candidate solutions, and select the N' best individuals.
2. Assume these N' best individuals were drawn i.i.d from a multivariate Gaussian, and estimate its mean μ and covariance matrix Σ by $\hat{\mu}$ and $\hat{\Sigma}$ using them.
3. Sample N new candidate solutions from the estimated distribution.
4. Go to 1.

Comment: One can show that the eigenvalues of $\hat{\Sigma}$ act like ‘learning rates’ or ‘temperatures’, and good estimates of them imply (under conditions) an optimal search policy.

Problems when Scaling Up EDA

In practice, objective functions evaluations are typically costly and so one tries to keep the population size $N \geqslant N'$ as small as possible. Then:

- If there are d parameters to optimize, and d is large, it is infeasible to have a full-rank estimate of $\hat{\Sigma}$ (i.e. to have $N' \geqslant d+1$).
- ML estimates of eigenvalues from few samples are very poor - the extrema are typically out by orders of magnitude.
- Sampling from a Gaussian in $\mathbb{R}^d$ is an $\mathcal{O}\left(d^3\right)$ operation, and this is also infeasible if d is large.
- Diagonal constraints on $\hat{\Sigma}$ and other simple regularizers can work poorly, and don't solve the issue of the sampling cost.

We want an approach that will: Keep N small, improve the eigenvalue estimates, NEVER sample in $\mathbb{R}^d$, avoid unjustified constraints on $\hat{\Sigma}$, and still obtain good outcomes from the optimization scheme.

Our Approach

Use random projections and simple averaging to implement a scaleable divide-and-conquer scheme for high-dimensional EDA [KBD13].

We make M different random projections of the N' best points, where the projection dimension is $k \ll N'$.

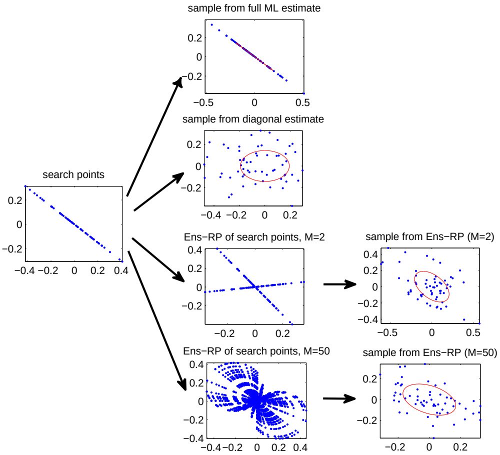
Estimate the Gaussian distribution in each of these M RP spaces, and sample a new population of size N from each one.

Project the new populations back to $\mathbb{R}^d$ using the transpose of the corresponding RP matrix.

Generate a new d-dimensional sample of size N in $\mathbb{R}^d$, by averaging the populations. i.e. for $j = 1, \ldots, N$ set:

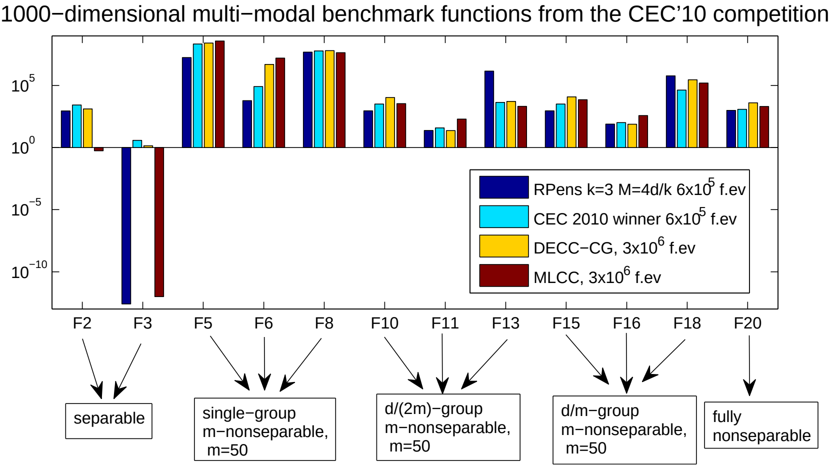
$$x_j = \frac{1}{M} \sum_{i=1}^{M} R_i^T R_i x_j^{(i)}$$

Why might this help?



- Search using ML estimate is constrained to lie in the range of $\hat{\Sigma}$.
- Diagonally-constrained EDA searches whole space, but ignores orientation of fitness density of the parent population.
- Sampling from RP subspaces and averaging respects orientation of the fitness density, while still searching whole space.

Does it work?


1000−dimensional multi−modal benchmark functions from the CEC’10 competition

separable

single−group m−nonseparable, m=50

d/(2m)−group m−nonseparable, m=50

d/m−group m−nonseparable, m=50

fully nonseparable

(Small is good here.)

Take me home!

- Random projections have a wide range of *theoretically well-motivated* and *effective* applications in machine learning and data mining.
- In particular, random projections:
  - are easy to implement,
  - can reduce time and space complexity of algorithms for small performance cost, and
  - can be used to construct parallel implementations of existing algorithms.
- Because random projection is *independent of data distribution*, theoretical analysis possible unlike many deterministic approaches.
- In particular, can avoid worst-case guarantees for random projections but not (usually) for approaches such as PCA.
- In high dimensions RP matrices act like approximate isometries, preserving geometric properties of data well but in a lower dimensional space.

Take me home!

- Random projections have a wide range of *theoretically well-motivated* and *effective* applications in machine learning and data mining.
- In particular, random projections:
  - are easy to implement,
  - can reduce time and space complexity of algorithms for small performance cost, and
  - can be used to construct parallel implementations of existing algorithms.
- Because random projection is *independent of data distribution*, theoretical analysis possible unlike many deterministic approaches.
- In particular, can avoid worst-case guarantees for random projections but not (usually) for approaches such as PCA.
- In high dimensions RP matrices act like approximate isometries, preserving geometric properties of data well but in a lower dimensional space.

Take me home!

- Random projections have a wide range of *theoretically well-motivated* and *effective* applications in machine learning and data mining.
- In particular, random projections:
  - are easy to implement,
  - can reduce time and space complexity of algorithms for small performance cost, and
  - can be used to construct parallel implementations of existing algorithms.
- Because random projection is *independent of data distribution*, theoretical analysis possible unlike many deterministic approaches.
- In particular, can avoid worst-case guarantees for random projections but not (usually) for approaches such as PCA.
- In high dimensions RP matrices act like approximate isometries, preserving geometric properties of data well but in a lower dimensional space.

Take me home!

- Random projections have a wide range of *theoretically well-motivated* and *effective* applications in machine learning and data mining.
- In particular, random projections:
  - are easy to implement,
  - can reduce time and space complexity of algorithms for small performance cost, and
  - can be used to construct parallel implementations of existing algorithms.
- Because random projection is *independent of data distribution*, theoretical analysis possible unlike many deterministic approaches.
- In particular, can avoid worst-case guarantees for random projections but not (usually) for approaches such as PCA.
- In high dimensions RP matrices act like approximate isometries, preserving geometric properties of data well but in a lower dimensional space.

Take me home!

- Random projections have a wide range of *theoretically well-motivated* and *effective* applications in machine learning and data mining.
- In particular, random projections:
  - are easy to implement,
  - can reduce time and space complexity of algorithms for small performance cost, and
  - can be used to construct parallel implementations of existing algorithms.
- Because random projection is *independent of data distribution*, theoretical analysis possible unlike many deterministic approaches.
- In particular, can avoid worst-case guarantees for random projections but not (usually) for approaches such as PCA.
- In high dimensions RP matrices act like approximate isometries, preserving geometric properties of data well but in a lower dimensional space.

Take me home!

- Random projections have a wide range of *theoretically well-motivated* and *effective* applications in machine learning and data mining.
- In particular, random projections:
  - are easy to implement,
  - can reduce time and space complexity of algorithms for small performance cost, and
  - can be used to construct parallel implementations of existing algorithms.
- Because random projection is *independent of data distribution*, theoretical analysis possible unlike many deterministic approaches.
- In particular, can avoid worst-case guarantees for random projections but not (usually) for approaches such as PCA.
- In high dimensions RP matrices act like approximate isometries, preserving geometric properties of data well but in a lower dimensional space.

Take me home!

- Random projections have a wide range of *theoretically well-motivated* and *effective* applications in machine learning and data mining.
- In particular, random projections:
  - are easy to implement,
  - can reduce time and space complexity of algorithms for small performance cost, and
  - can be used to construct parallel implementations of existing algorithms.
- Because random projection is *independent of data distribution*, theoretical analysis possible unlike many deterministic approaches.
- In particular, can avoid worst-case guarantees for random projections but not (usually) for approaches such as PCA.
- In high dimensions RP matrices act like approximate isometries, preserving geometric properties of data well but in a lower dimensional space.

Take me home!

- Random projections have a wide range of *theoretically well-motivated* and *effective* applications in machine learning and data mining.
- In particular, random projections:
  - are easy to implement,
  - can reduce time and space complexity of algorithms for small performance cost, and
  - can be used to construct parallel implementations of existing algorithms.
- Because random projection is *independent of data distribution*, theoretical analysis possible unlike many deterministic approaches.
- In particular, can avoid worst-case guarantees for random projections but not (usually) for approaches such as PCA.
- In high dimensions RP matrices act like approximate isometries, preserving geometric properties of data well but in a lower dimensional space.

Take me home!

- Random projections have a wide range of *theoretically well-motivated* and *effective* applications in machine learning and data mining.
- In particular, random projections:
    - are easy to implement,
    - can reduce time and space complexity of algorithms for small performance cost, and
    - can be used to construct parallel implementations of existing algorithms.
- Because random projection is *independent of data distribution*, theoretical analysis possible unlike many deterministic approaches.
- In particular, can avoid worst-case guarantees for random projections but not (usually) for approaches such as PCA.
- In high dimensions RP matrices act like approximate isometries, preserving geometric properties of data well but in a lower dimensional space.

Thank You!

References I

[ABN^{+}99] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine, *Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays*, Proceedings of the National Academy of Sciences **96** (1999), no. 12, 6745.

[AC06] N. Ailon and B. Chazelle, *Approximate nearest neighbors and the fast johnson-lindenstrauss transform*, Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, ACM, 2006, pp. 557–563.

[Ach03] D. Achlioptas, *Database-friendly random projections: Johnson-Lindenstrauss with binary coins*, Journal of Computer and System Sciences **66** (2003), no. 4, 671–687.

[AI06] A. Andoni and P. Indyk, *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions*, Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on, IEEE, 2006, pp. 459–468.

[AL09] N. Ailon and E. Liberty, *Fast dimension reduction using rademacher series on dual bch codes*, Discrete & Computational Geometry **42** (2009), no. 4, 615–630.

[AL11] ———, *An almost optimal unrestricted fast johnson-lindenstrauss transform*, Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2011, pp. 185–191.

[AMS96] N. Alon, Y. Matias, and M. Szegedy, *The space complexity of approximating the frequency moments*, Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, ACM, 1996, pp. 20–29.

References II

[AV06] R. Arriaga and S. Vempala, *An algorithmic theory of learning*, Machine Learning **63** (2006), no. 2, 161–182.

[AV09] R. Avogadri and G. Valentini, *Fuzzy ensemble clustering based on random projections for dna microarray data analysis*, Artificial Intelligence in Medicine **45** (2009), no. 2, 173–183.

[BD09] C. Boutsidis and P. Drineas, *Random projections for the nonnegative least-squares problem*, Linear Algebra and its Applications **431** (2009), no. 5-7, 760–771.

[BDDW08] R.G. Baraniuk, M. Davenport, R.A. DeVore, and M.B. Wakin, *A Simple Proof of the Restricted Isometry Property for Random Matrices* , Constructive Approximation **28** (2008), no. 3, 253–263.

[BL04] P. Bickel and E. Levina, *Some theory for Fisher's linear discriminant function, 'naïve Bayes', and some alternatives when there are many more variables than observations*, Bernoulli **10** (2004), no. 6, 989–1010.

[BM01] E. Bingham and H. Mannila, *Random projection in dimensionality reduction: applications to image and text data.*, Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001) (F. Provost and R. Srikant, ed.), 2001, pp. 245–250.

[Bre96] L. Breiman, *Bagging predictors*, Machine learning **24** (1996), no. 2, 123–140.

[Bre01] ______, *Random forests*, Machine learning **45** (2001), no. 1, 5–32.

References III

[BW09] R.G. Baraniuk and M.B. Wakin, *Random projections of smooth manifolds*, Foundations of Computational Mathematics **9** (2009), no. 1, 51–77.

[Cha02] M.S. Charikar, *Similarity estimation techniques from rounding algorithms*, Proceedings of the thirty-fourth annual ACM symposium on Theory of computing, ACM, 2002, pp. 380–388.

[CJS09] R. Calderbank, S. Jafarpour, and R. Schapire, *Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain*, Tech. report, Rice University, 2009.

[CT06] E.J. Candes and T. Tao, *Near-optimal signal recovery from random projections: Universal encoding strategies?*, Information Theory, IEEE Transactions on **52** (2006), no. 12, 5406–5425.

[Das99] S. Dasgupta, *Learning Mixtures of Gaussians*, Annual Symposium on Foundations of Computer Science, vol. 40, 1999, pp. 634–644.

[DF08] S. Dasgupta and Y. Freund, *Random projection trees and low dimensional manifolds*, Proceedings of the 40th annual ACM symposium on Theory of computing, ACM, 2008, pp. 537–546.

[DG02] S. Dasgupta and A. Gupta, *An Elementary Proof of the Johnson-Lindenstrauss Lemma*, Random Struct. Alg. **22** (2002), 60–65.

References IV

[DK10] R.J. Durrant and A. Kabán, *Compressed Fisher Linear Discriminant Analysis: Classification of Randomly Projected Data*, Proceedings16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2010), 2010.

[DK12] ______, *Error bounds for Kernel Fisher Linear Discriminant in Gaussian Hilbert space*, Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AIStats 2012), 2012.

[DK13] RJ Durrant and A Kabán, *Sharp generalization error bounds for randomly-projected classifiers*, International Conference on Machine Learning (ICML 2013), vol. 16, 2013, p. 21.

[DK14] R.J. Durrant and A. Kabán, *Random Projections as Regularizers: Learning a Linear Discriminant from Fewer Observations than Dimensions*, Machine Learning (In Press) (2014).

[DKS10] A. Dasgupta, R. Kumar, and T. Sarlós, *A sparse johnson-lindenstrauss transform*, Proceedings of the 42nd ACM symposium on Theory of computing, ACM, 2010, pp. 341–350.

[Don06] D.L. Donoho, *Compressed Sensing*, IEEE Trans. Information Theory **52** (2006), no. 4, 1289–1306.

[FB03] X.Z. Fern and C.E. Brodley, *Random projection for high dimensional data clustering: A cluster ensemble approach*, International Conference on Machine Learning, vol. 20, 2003, p. 186.

References V

[FM03] D. Fradkin and D. Madigan, *Experiments with random projections for machine learning*, Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2003, pp. 522–529.

[GBN05] N. Goel, G. Bebis, and A. Nefian, *Face recognition experiments with random projection*, Proceedings of SPIE, vol. 5779, 2005, p. 426.

[GGBHD03] Isabelle Guyon, S Gunn, A Ben-Hur, and G Dror, *NIPS 2003 Workshop on Feature Extraction: Dorothea dataset*, 2003.

[GST$^+$99] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander, *Molecular classification of cancer: class discovery and class prediction by gene expression monitoring*, Science **286** (1999), no. 5439, 531.

[Ho98] T.K. Ho, *The random subspace method for constructing decision forests*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **20** (1998), no. 8, 832–844.

[HP01] S. Har-Peled, *A replacement for voronoi diagrams of near linear size*, Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on, IEEE, 2001, pp. 94–103.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning; data mining, inference, and prediction*, Springer, 2001.

References VI

[HWB07] C. Hegde, M.B. Wakin, and R.G. Baraniuk, *Random projections for manifold learningproofs and analysis*, Neural Information Processing Systems, 2007.

[IM98] P. Indyk and R. Motwani, *Approximate nearest neighbors: towards removing the curse of dimensionality*, Proceedings of the thirtieth annual ACM symposium on Theory of computing, ACM New York, NY, USA, 1998, pp. 604–613.

[Kab14] Ata Kabán, *New bounds on compressive linear least squares regression*, Proceedings of the 17-th International Conference on Artificial Intelligence and Statistics (AISTATS 2014), Journal of Machine Learning Research-Proceedings Track, vol. 33, 2014, pp. 448–456.

[KBD13] Ata Kaban, Jakramate Bootkrajang, and Robert John Durrant, *Towards large scale continuous eda: A random matrix theory perspective*, Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation (New York, NY, USA), GECCO '13, ACM, 2013, pp. 383–390.

[Kle97] J.M. Kleinberg, *Two algorithms for nearest-neighbor search in high dimensions*, Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, ACM, 1997, pp. 599–608.

[KMN11] D. Kane, R. Meka, and J. Nelson, *Almost optimal explicit johnson-lindenstrauss families*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (2011), 628–639.

References VII

[KMV12] A.T. Kalai, A. Moitra, and G. Valiant, *Disentangling gaussians*, Communications of the ACM **55** (2012), no. 2, 113–120.

[KV94] M.J. Kearns and U.V. Vazirani, *An introduction to computational learning theory*, MIT Press, 1994.

[Led01] M. Ledoux, *The concentration of measure phenomenon*, vol. 89, American Mathematical Society, 2001.

[LN14] Kasper Green Larsen and Jelani Nelson, *The Johnson-Lindenstrauss lemma is optimal for linear dimensionality reduction*, arXiv preprint arXiv:1411.2404 (2014).

[LW04] O. Ledoit and M. Wolf, *A well-conditioned estimator for large-dimensional covariance matrices*, Journal of multivariate analysis **88** (2004), no. 2, 365–411.

[Mat08] J. Matoušek, *On variants of the johnson–lindenstrauss lemma*, Random Structures & Algorithms **33** (2008), no. 2, 142–156.

[MM09] O. Maillard and R. Munos, *Compressed Least-Squares Regression*, NIPS, 2009.

[MP69] M. Minsky and S. Papert, *Perceptrons*, MIT press, 1969.

[Rec11] B. Recht, *A simpler approach to matrix completion*, Journal of Machine Learning Research **12** (2011), 3413–3430.

[RR08] A. Rahimi and B. Recht, *Random features for large-scale kernel machines*, Advances in neural information processing systems **20** (2008), 1177–1184.

References VIII

[Sar06] T. Sarlos, *Improved approximation algorithms for large matrices via random projections*, Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on, IEEE, 2006, pp. 143–152.

[SFR$^+$02] D. Singh, P.G. Febbo, K. Ross, D.G. Jackson, J. Manola, C. Ladd, P. Tamayo, A.A. Renshaw, A.V. D'Amico, J.P. Richie, E.S. Lander, M. Loda, P.W. Kantoff, T.R. Golub, and W.S. Sellers, *Gene expression correlates of clinical prostate cancer behavior*, Cancer cell **1** (2002), no. 2, 203–209.

[SR09] A. Schclar and L. Rokach, *Random projection ensemble classifiers*, Enterprise Information Systems (Joaquim Filipe, Jos Cordeiro, Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, and Clemens Szyperski, eds.), Lecture Notes in Business Information Processing, vol. 24, Springer, 2009, pp. 309–316.

[SSSS08] K. Sridharan, N. Srebro, and S. Shalev-Shwartz, *Fast rates for regularized objectives*, Advances in Neural Information Processing Systems 21, 2008, pp. 1545–1552.

[WBD$^+$01] M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J.A. Olson, J.R. Marks, and J.R. Nevins, *Predicting the clinical status of human breast cancer by using gene expression profiles*, Proceedings of the National Academy of Sciences **98** (2001), no. 20, 11462.

Appendix

Proposition JLL for dot products.
Let $x_n, n = \{1 \ldots N\}$ and u be vectors in $\mathbb{R}^d$ s.t. $\|x_n\|, \|u\| \leqslant 1$.
Let R be a $k \times d$ RP matrix with i.i.d. entries $R_{ij} \sim \mathcal{N}(0, 1/\sqrt{k})$ (or with zero-mean sub-Gaussian entries).
Then for any $\epsilon, \delta > 0$, if $k \in \mathcal{O}\left(\frac{8}{\epsilon^2} \log(4N/\delta)\right)$ w.p. at least $1 - \delta$ we have:

$$|x_n^T u - (Rx_n)^T Ru| < \epsilon \tag{5}$$

simultaneously for all $n = \{1 \ldots N\}$.

Proof of JLL (1)

We will prove the following 'distributional' version of the JLL, and then show that this implies the original theorem:

Theorem (Distributional JLL)

Let $\epsilon \in (0, 1)$. *Let* $k \in \mathbb{N}$ *such that* $k \geqslant C\epsilon^{-2} \log \delta^{-1}$, *for a large enough absolute constant* C. *Then there is a random linear mapping* $P : \mathbb{R}^d \to \mathbb{R}^k$, *such that for any unit vector* $x \in \mathbb{R}^d$:

$$Pr\left\{(1 - \epsilon) \leqslant \|Px\|^2 \leqslant (1 + \epsilon)\right\} \geqslant 1 - \delta$$

- No loss to take $\|x\| = 1$, since P is linear.
- Note that the projected dimension k depends only on ϵ and δ.
- Lower bound $k \in \Omega(\epsilon^{-2} \log \delta^{-1})$ sharp for randomized dimensionality reduction [KMN11, LN14].

Proof of JLL (2)

Consider the following simple mapping:

$$Px := \frac{1}{\sqrt{k}} Rx$$

where $R \in \mathcal{M}_{k \times d}$ with entries $R_{ij} \overset{i.i.d}{\sim} \mathcal{N}(0, 1)$.

Let $x \in \mathbb{R}^d$ be an arbitrary unit vector.
We are interested in quantifying:

$$\|Px\|^2 = \left\| \frac{1}{\sqrt{k}} Rx \right\|^2 := \left\| \frac{1}{\sqrt{k}} (Y_1, Y_2, \ldots, Y_k) \right\|^2 = \frac{1}{k} \sum_{i=1}^{k} Y_i^2 =: Z$$

where $Y_i = \sum_{j=1}^{d} R_{ij} x_j$.

Proof of JLL (3)

Recall that if $W_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and the W_i are independent, then $\sum_i W_i \sim \mathcal{N}\left(\sum_i \mu_i, \sum_i \sigma_i^2\right)$. Hence, in our setting, we have:

$$Y_i = \sum_{j=1}^{d} R_{ij} x_j \sim \mathcal{N}\left(\sum_{j=1}^{d} \mathrm{E}[R_{ij} x_j], \sum_{j=1}^{d} \mathrm{Var}(R_{ij} x_j)\right) \equiv \mathcal{N}\left(0, \sum_{j=1}^{d} x_j^2\right)$$

and since $\|x\|^2 = \sum_{j=1}^{d} x_j^2 = 1$ we therefore have:

$$Y_i \sim \mathcal{N}\left(0, 1\right), \ \forall i \in \{1, 2, \ldots, k\}$$

it follows that each of the Y_i are standard normal RVs and therefore $kZ = \sum_{i=1}^{k} Y_i^2$ is χ_k^2 distributed.

Now we complete the proof using a standard Chernoff-bounding approach.

Proof of JLL (4)

$$\begin{aligned}
& \Pr\{Z > 1+\epsilon\} && = && \Pr\{\exp(tkZ) > \exp(tk(1+\epsilon))\},\ \forall t > 0 \\
& \text{Markov ineq.} && \leqslant && \mathrm{E}\left[\exp(tkZ)\right] / \exp(tk(1+\epsilon)), \\
& Y_i \text{ indep.} && = && \prod_{i=1}^{k} \mathrm{E}\left[\exp(tY_i^2)\right] / \exp(tk(1+\epsilon)), \\
& \text{mgf of } \chi^2 && = && \left[\exp(t)\sqrt{1-2t}\right]^{-k} \exp(-kt\epsilon), \forall t < 1/2 \\
& \text{next slide} && \leqslant && \exp\left(kt^2/(1-2t) - kt\epsilon\right), \\
& && \leqslant && e^{-\epsilon^2 k/8}, \text{ taking } t = \epsilon/4 < 1/2.
\end{aligned}$$

$\Pr\{Z < 1-\epsilon\} = \Pr\{-Z > \epsilon - 1\}$ is tackled in a similar way and gives same bound. Taking RHS as $\delta/2$ and applying union bound completes the proof (for a single x).

Estimating $(e^t\sqrt{1-2t})^{-1}$

$$\begin{aligned}
\left(e^t\sqrt{1-2t}\right)^{-1} &= \exp\left(-t-\frac{1}{2}\log(1-2t)\right), \\
\text{Maclaurin S. for } \log(1-x) \quad &= \exp\left(-t-\frac{1}{2}\left(-2t-\frac{(2t)^2}{2}-\ldots\right)\right), \\
&= \exp\left(\frac{(2t)^2}{4}+\frac{(2t)^3}{6}+\ldots\right), \\
&\leqslant \exp\left(t^2\left(1+2t+(2t)^2\ldots\right)\right), \\
&= \exp\left(t^2/\left(1-2t\right)\right) \text{ since } 0 < 2t < 1
\end{aligned}$$

Randomized JLL implies Deterministic JLL

- Solving $\delta = 2\exp(-\epsilon^2 k/8)$ for k we obtain $k = 8/\epsilon^2 \log \delta^{-1} + \log 2$. i.e. $k \in \mathcal{O}\left(\epsilon^{-2}\log \delta^{-1}\right)$.
- Let $V = \{x_1, x_2, \ldots, x_N\}$ an arbitrary set of N points in $\mathbb{R}^d$ and set $\delta = 1/N^2$, then $k \in \mathcal{O}\left(\epsilon^{-2}\log N\right)$.
- Applying union bound to the randomized JLL proof for all $\binom{N}{2}$ possible interpoint distances, for N points we see a random JLL embedding of V into k dimensions succeeds with probability at least $1 - \binom{N}{2}\frac{1}{N^2} > \frac{1}{2}$.
- We succeed with positive probability for arbitrary V. Hence we conclude that, for any set of N points, there exists linear $P : \mathbb{R}^d \to \mathbb{R}^k$ such that:

$$(1-\epsilon)\|x_i - x_j\|^2 \leqslant \|Px_i - Px_j\|^2 \leqslant (1+\epsilon)\|x_i - x_j\|^2$$

which is the (deterministic) JLL.

Comment (2)

In the proof of the randomized JLL the only properties we used which are specific to the Gaussian distribution were:

1. Closure under additivity.
2. Bounding squared Gaussian RV using mgf of χ^2.

In particular, bounding via the mgf of χ^2 gave us exponentially fast concentration about mean norm.

Can do similar for matrices with zero-mean *sub-Gaussian* entries also: Sub-Gaussians are those distributions whose tails decay no slower than a Gaussian, e.g. practically all bounded distributions have this property.

We obtain similar guarantees (i.e. up to small multiplicative constants) for sub-Gaussian RP matrices too!

This allows us to get around issue of *dense matrix multiplication* in dimensionality-reduction step.

Proof of JLL for dot products

Outline: Fix one n, use parallelogram law and JLL twice, then use union bound.

$$\begin{aligned}
4(Rx_n)^T(Ru) &= \|Rx_n + Ru\|^2 - \|Rx_n - Ru\|^2 && (6)\\
&\geqslant (1-\epsilon)\|x_n + u\|^2 - (1+\epsilon)\|x_n - u\|^2 && (7)\\
&= 4x_n^T u - 2\epsilon(\|x_n\|^2 + \|u\|^2) && (8)\\
&\geqslant 4x_n^T u - 4\epsilon && (9)
\end{aligned}$$

Hence, $(Rx_n)^T(Ru) \geqslant x_n^T u - \epsilon$, and because we used two sides of JLL, this holds except w.p. no more than $2\exp(-k\epsilon^2/8)$.
The other side is similar and gives $(Rx_n)^T(Ru) \leqslant x_n^T u + \epsilon$ except w.p. $2\exp(-k\epsilon^2/8)$.
Put together, $|(Rx_n)^T(Ru) - x_n^T u| \leqslant \epsilon \cdot \frac{\|x\|^2+\|u\|^2}{2} \leqslant \epsilon$ holds except w.p. $4\exp(-k\epsilon^2/8)$.
This holds for a fixed x_n. To ensure that it holds for all x_n together, we take union bound and obtain eq.(5) must hold except w.p. $4N\exp(-k\epsilon^2/8)$. Finally, solving for δ we obtain that $k \geqslant \frac{8}{\epsilon^2}\log(4N/\delta)$.