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 $\in O(1000)$) and very many observations ($N \in O(1000)$): Computational (time and space complexity) issues.
- Very high dimensional data (arity $\in O(1000)$) and hardly any observations ($N \in O(10)$): 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, $T$, are $d$-dimensional and there are $N$ of them: $T = \{x_i \in \mathbb{R}^d\}_{i=1}^N$ 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. Beyond JLL and Compressed Sensing

**Johnson-Lindenstrauss Lemma**

The JLL is the following rather surprising fact [DG02, Ach03]:

**Theorem (Johnson and Lindenstrauss, 1984)**

Let \( \epsilon \in (0, 1) \). Let \( N, k \in \mathbb{N} \) such that \( k \geq C \epsilon^{-2} \log N \), for a large enough absolute constant \( C \). Let \( V \subseteq \mathbb{R}^d \) be a set of \( N \) points. Then there exists a linear mapping \( R : \mathbb{R}^d \to \mathbb{R}^k \), such that for all \( u, v \in V \):

\[
(1 - \epsilon) \| u - v \|^2 \leq \| Ru - Rv \|^2 \leq (1 + \epsilon) \| u - v \|^2
\]

- Dot products are also approximately preserved by \( R \) since if JLL holds then: \( u^T v - \epsilon \leq (Ru)^T Rv \leq u^T v + \epsilon \). (Proof: parallelogram law - see appendix).
- Scale of \( k \) is essentially sharp: \( \forall N, \exists V \text{ s.t. } k \in \Omega(\epsilon^{-2} \log N/ \log \epsilon^{-1}) \) is required [Alo03].
- We shall prove shortly that with high probability random projection implements a suitable linear \( R \).

**Intuition**

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

- Figure: Original data
- Figure: RP data (schematic)

**Applications**

Random projections have been used for:

- Classification. e.g. [BM01, FM03, GBN05, SR09, CJS09, RR08, DK12b]
- Regression. e.g. [MM09, HWB07, BD09]
- 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].

**What is Random Projection? (1)**

Canonical RP:

- Construct a (wide, flat) matrix \( R \in \mathbb{M}_{d \times k} \) by picking the entries from a univariate Gaussian \( N(0, \sigma^2) \).
- 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 \( v \mapsto R'v \in \mathbb{R}^k \equiv \mathbb{R}^k \) is the projection of the \( d \)-dimensional data into a random \( k \)-dimensional projection space.
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 [DK12a]:

$$\Pr \left\{ (1 - \epsilon)ds^2 \leq \|R_i\|^2 \leq (1 + \epsilon)ds^2 \right\} \geq 1 - \delta, \; \forall \epsilon \in (0, 1]$$

where $R_i$ denotes the $i$-th row of $R$ and

$$\delta = \exp(-\sqrt{T + \epsilon - 1}^2d^2/2) + \exp(-\sqrt{T - \epsilon - 1}^2d/2).$$

Similarly [Led01]:

$$\Pr(||R_i^T R_j||/ds^2 \leq \epsilon) \geq 1 - 2\exp(-\epsilon^2d^2/2), \; \forall i \neq j.$$ 

Why Random Projection?

- Linear.
- Cheap.
- Universal – JLL holds w.h.p for any fixed finite point set.
- Oblivious to data distribution.
- Target dimension doesn’t depend on data dimensionality (for JLL).
- Interpretable - approximates an isometry (when $d$ is large).
- Tractable to analysis.

Concentration in norms of rows of $R$

```
Figure: $d = 100$ norm concentration
Figure: $d = 500$ norm concentration
Figure: $d = 1000$ norm concentration
```

Jargon

- ’With high probability’ (w.h.p) means with a probability as close to 1 as we choose to make it.
- ’Almost surely’ (a.s.) or ’with probability 1’ (w.p. 1) means so likely we can pretend it always happens.
- ’With probability 0’ (w.p. 0) means so unlikely we can pretend it never happens.

Proof of JLL (1)

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

**Theorem**

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

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

- No loss to take $\|x\| = 1$, since $P$ is linear.
- Note that this mapping is universal and the projected dimension $k$ depends only on $\epsilon$ and $\delta$.
- Lower bound [JW11, KMN11] $k \in \Omega(\epsilon^{-2} \log \delta^{-1})$
Proof of JLL (2)

Consider the following simple mapping:

$$PX := \frac{1}{\sqrt{k}R}Rx$$

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

Let $X \in \mathbb{R}^d$ be an arbitrary unit vector.

We are interested in quantifying:

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

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

Proof of JLL (3)

Recall that if $W_i \sim \mathcal{N}(\mu_i, \sigma^2)$ and the $W_i$ are independent, then

$$\sum W_i \sim \mathcal{N}\left(\sum \mu_i, \sum \sigma^2_i\right).$$

Hence, in our setting, we have:

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

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

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

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

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

Proof of JLL (4)

Recall that

$$\Pr(Z > 1 + \epsilon) = \Pr(\exp(tkZ) > \exp(tk(1 + \epsilon)),$$ for $t > 0$

Markov ineq. $\leq \mathbb{E}[\exp(tkZ)] / \exp(tk(1 + \epsilon))$.

Y indep. $= \prod_{i=1}^{k} \left[\exp(\epsilon Y_i^2) / \exp(\epsilon(k + 1)\epsilon)\right].$

mgf of $\chi^2 = \left[\exp(t \sqrt{1 - 2t}) - k^2\epsilon\exp(-kt)\right], \forall t < 1/2$

next slide $\leq \exp \left[\frac{t^2}{(1 - 2t) - kt}\right], k^2$.

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

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

$$\left(e^{\sqrt{1 - 2t}}\right)^{-1} = \exp(-t - \frac{1}{2} \log(1 - 2t))$$

Maclaurin S. for log(1 - x) = $\exp(-t - \frac{1}{2} \log(1 - 2t))$.

Applying union bound to the randomized JLL proof for all $t$ possible point distances, for $N$ points we see a random JLL embedding of $V$ into $k$ dimensions succeeds with probability at least $1 - \left(\frac{1}{2}\right)^N > \frac{1}{2}$.

We succeed with positive probability for arbitrary $V$. Hence we conclude that, for any set of $N$ points, there exists linear $P : \mathbb{R}^d \rightarrow \mathbb{R}^k$ such that:

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

which is the (deterministic) JLL.

Randomized JLL implies Deterministic JLL

- Solving $\delta = 2 \exp(-c_k/8)$ for $k$ we obtain
  $k = 8/c_2 \log \delta^{-1}$.

- 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 $k \in O(\log N)/N^c$.

- Applying union bound to the randomized JLL proof for all $t$ possible point distances, for $N$ points we see a random JLL embedding of $V$ into $k$ dimensions succeeds with probability at least $1 - \left(\frac{1}{2}\right)^N > \frac{1}{2}$.

- We succeed with positive probability for arbitrary $V$. Hence we conclude that, for any set of $N$ points, there exists linear $P : \mathbb{R}^d \rightarrow \mathbb{R}^k$ such that:

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

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:

2. Bounding squared Gaussian RV using mgf of $\chi^2$.

In particular, bounding via the mgf of $\chi^2$ gave us exponential 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, for example all bounded distributions have this property.

Can derive 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.

Popular variations [Ach03, AC06, Mat08]:

The entries \( R_j \) can be:

\[
R_j = \begin{cases}
  +1 & \text{w.p. } 1/2, \\
  -1 & \text{w.p. } 1/2,
\end{cases}
\]

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]: \( R_j = 0 \text{ w.p. } 2/3 \)

Ailon-Chazelle 06 [AC06]: Use \( x \mapsto \text{PHD}_x \), \( P \) random and sparse, \( R_j \sim \mathcal{N}(0,1/q) \) w.p. 1/q, \( H \) normalized Hadamard (orthogonal) matrix, \( D = \text{diag}(\pm 1) \) random. Mapping takes \( O(d \log d + q \delta^{-2} \log N) \).

Ailon-Liberty 09 [AL09]: Similar construction to [AC06].

Ailon-Liberty '09 [AL09]: Similar construction to [AC06].

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

Union bound over subspaces to preserve large distances.

Achlioptas '01 [Ach03]: zero-mean subgaussian distribution. All behave in much the same way.

Manifold JLL

Definition: \( M \subset \mathbb{R}^d \) is an \( s \)-dimensional manifold if \( \forall x \in M \) there is a smooth bijective map between \( \mathbb{R}^s \) and a neighbourhood of \( x \).

Let \( M \) be an \( s \)-dimensional manifold in \( \mathbb{R}^d \) with bounded curvature. (\( M \) is locally like a linear subspace.) For \( k = O(\epsilon^{-2}\log d) \) [BW09] w.h.p a JLL matrix \( R \) satisfies \( \forall x, y \in M: (1-\epsilon)\|x-y\| \leq \|Rx-Ry\| \leq (1+\epsilon)\|x-y\| \).

For this, \( k = O(\epsilon^{-2}\log d) \) [BW09] w.h.p a JLL matrix \( R \) satisfies \( \forall x, y \in M: (1-\epsilon)\|x-y\| \leq \|Rx-Ry\| \leq (1+\epsilon)\|x-y\| \).

Proof idea:

- Approximate manifold with tangent subspaces.
- Apply subspace-JLL on each subspace.
- Union bound over subspaces to preserve large distances.

(Jointly used to preserve geodesic distances.)

Generalizations of JLL to Manifolds

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

JLL approach can be extended to (compact) Riemannian manifolds:

- **Manifold JLL**
  - Key idea: Preserve \( \epsilon \)-covering of smooth manifold under some metric instead of geometry of data points. Replace \( N \) with corresponding covering number \( M \) and take \( k = \mathcal{O}(\epsilon^{-2}\log M) \).
Outline

- Background and Preliminaries
- Johnson-Lindenstrauss Lemma (JLL) and extensions
- Applications of JLL (1)
  - Approximate Nearest Neighbour Search
  - RP Perceptron
  - Mixtures of Gaussians
  - Random Features
- Compressed Sensing
- Applications of JLL (2)
- Beyond JLL and Compressed Sensing

Applications of Random Projection (1)

We have seen, via JLL, that with a suitable choice of k we can construct an ‘ϵ-approximate’ version of any algorithm which depends only on the 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 bypass the collection of lots of data then throwing away most of it at preprocessing (Compressed sensing).
- 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).

Solution: Work with random projections of the data.

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 \( T = \{x_1, ..., x_N\} \) in \( \mathbb{R}^d \), find the closest point to a query point \( x_q \).
- Approximate NNS: Find \( x \in T \) that is \( ϵ \)-close to \( x_q \). That is, such that \( ∀x' \in T, \|x - x_q\| ≤ (1 + ϵ)\|x' - x_q\| \).

The problem: Space or time complexity exponential in \( d \) even for sophisticated approximate NNS. [Kle97, HP01, A106].

Nearest Neighbour Search

- The first known approximate NNS algorithm with space and time complexity polynomial in \( d \) is due to Indyk & Motwani ’98 [IM98].
- It is based on the idea of locality sensitive hashing, and using the Johnson Lindenstrauss Lemma (JLL).
- Have an algorithm with query time \( O(d \log N) \) random projections.
- This yields an algorithm that has query time \( O(N^{-\frac{1}{2}}) \).

Since this important advance, there have been many further results on approximate NNS (including other uses of random projection) 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:
  - When the target concept robust, random projection of examples to a low-dimensional subspace preserves the concept.
  - 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 \( D \) on \( \mathbb{R}^d \), is said to be robust, if
\[
Pr\{x \in X' \mid \text{label}(x) \neq \text{label}(x') \} < \ell,
\]
and \( |x - x'| \leq \ell \).

Given \( T = \{(x_1, y_1), \ldots, (x_N, y_N)\} \) a \( D^N \) labelled training set, \( R \in M_{k,d} \) a random matrix with zero-mean sub-Gaussian entries.

Suppose \( T \) is a sample from a robust concept, i.e. \( \exists h \in \mathbb{R}^d, ||h|| = 1 \)
s.t. \( \forall x \in \{1, \ldots, N\}, y_k = h^T x_k > \ell \).

Algorithm

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

For a query point \( x_q \) predict \( h_R^T Rx_q \).

We now want to obtain a PAC learning guarantee on the generalization error and guarantee on running time of this algorithm.

Denote \( L = \max_{x=1, \ldots, N} ||x||^2 \). We apply JLL to preserve all these \( N \) dot-products as well as all the lengths \( ||x||^2 \) and \( ||h||^2 \), with the choice \( \ell/\sqrt{k} \) for the preservation tolerance parameter (i.e. in place of \( \ell \) in JLL).

To have (1)-(2) except w.p. \( \delta/2 \), we need:
\[
k = O\left(\frac{L \log 12N/\delta}{\ell^2} \right) \tag{3}
\]
where \( L \) is the (squared) diameter of the data. We can take \( L = 1 \) w.l.o.g.

Therefore \( k \) of this order is needed to get the generalization bound via Kearns & Vazirani’s theorem.

Comment: In [AV06] the authors obtain \( k = O\left(\frac{1}{\ell^2} \log 12N/\delta \right) \) by taking the allowed misclassification rate as \( \ell > 1/N \). This somewhat obscures the logarithmic increase in the upper bound on generalization error with the number of training points \( N \).

Approach: Use known results on generalization [KV94] for halfspaces, and on the running time of Perceptron [MP69] in \( \mathbb{R}^d \), and use JLL to ensure their preconditions hold w.h.p. Here we focus on the former (the latter goes similarly).

Theorem (Kearns & Vazirani ‘94) [for halfspaces, i.e. \( VCdim = k + 1 \)].

Let \( h \) be the concept class of robust halfspaces. Let \( \epsilon, \delta \in (0, 1) \), and let \( h \in h \) be a concept that is consistent with \( N \) i.i.d. labelled examples \( T \sim D \).

Then, \( \exists h \in h \) (w.r.t. the random draws of the training set \( T \), \( h \) correctly classifies at least \( 1 - \epsilon \) fraction of \( D \) with probability at least \( 1 - \delta \) provided that \( N > \frac{\epsilon}{\delta} \log \frac{2k}{\delta} + \frac{\epsilon}{\delta} \log 2 \).

Provably Learning Mixtures of Gaussians

- Mixtures of Gaussians (MoG) are among the most fundamental and widely used statistical models, \( p(x) = \sum_{j=1}^K \pi_j N(x|\mu_j, \Sigma_j) \), where \( N(x|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1} (x-\mu_j)\right) \).
- Given a set of unlabelled data points drawn from a MoG, the goal is to estimate the mean \( \mu_j \) and covariance \( \Sigma_j \) 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 \( S \) of \( N \) data points in \( \mathbb{R}^d \); \( m \) number of mixture components; \( r, \delta \) resp. accuracy and confidence parameters. \( \pi_{\text{min}} \): smallest mixture weight to be considered.

1. Randomly project the data onto a \( k \)-dimensional subspace of the original space \( \mathbb{R}^d \). Takes time \( O(nkd) \).
2. In the projected space:
   - For \( x \in S \), let \( r_x \) be the smallest radius such that there are \( \geq p \) points within distance \( r_x \) of \( x \).
   - Start with \( S' = S \).
   - For \( y = 1, \ldots, m \) let \( r_y \) be the point with the lowest \( r_y \).
   - Find the \( q \) closest points to this estimated center.
   - Remove these points from \( S' \).
   - For each \( y \), let \( S_y \) denote the points in \( S' \) which are closest to \( r_y \).
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 $||\mu_1 - \mu_2|| \geq c\sqrt{d} \cdot \max\{\lambda_{\max}(\Sigma_1), \lambda_{\max}(\Sigma_2)\}$. A mixture of Gaussians is c-separated if its components are c-separated.

**Theorem**

Let $\delta, c \in (0, 1)$. 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 $\Sigma$ with condition number $\kappa = \lambda_{\max}(\Sigma)/\lambda_{\min}(\Sigma)$, and $\min_i \gamma_i = \Omega(1/m)$. Then:

- w.p. $1 - \delta$, the centre estimates returned by the algorithm are accurate within $c\sqrt{d}\lambda_{\max}$;
- if $\sqrt{\gamma} \leq \mathcal{O}(d^{1/2}/\log(m/(\epsilon\delta)))$, then the reduced dimension required is $k = \mathcal{O}(\log m/(\epsilon\delta))$, and the number of data points needed is $N = \mathcal{O}(\log^2(n)/\epsilon^2)$. The algorithm runs in time $\mathcal{O}(N^3k + Nkd)$.

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

- A c-separated mixture becomes a $(c \cdot \sqrt{1-\gamma})$-separated mixture w.p. $1 - \delta$ after RP. This is because
  - JLL proves that the distances between centers are preserved
  - $\lambda_{\max}(\Sigma_i/R) \leq \lambda_{\max}(\Sigma)$
- 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. overlap is allowed) in polynomial time also use RP.

**Random features as an alternative to the kernel trick**

Kernel trick: $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, $i, j \in \{1, \ldots, N\}$.

Problem: When $N$ is large, storage & computational cost is large. Evaluating a test point $x_q$ requires computing $k(x_q) = \sum_{i=1}^N q_i k(x_i, x_q)$. This is $\mathcal{O}(Nd)$.

Idea: [RR08] (also earlier in [Kon07]; later in work on Hash kernels [WDL+09]).

Construct $z : \mathbb{R}^d \rightarrow \mathbb{R}^k$ s.t. $\|k(x_i, x_j) - z(x_i) \cdot z(x_j)\| < \epsilon$, $\forall x_i, x_j \in M$ where $M \subset \mathbb{R}^d$ a compact domain.

For shift-invariant kernels this can be done for $k = \mathcal{O}(d r^2 \log(1/\epsilon^2))$.

Shift-invariant kernel: has the form $k(x_i, x_j) = Q(x_i - x_j)$, where $Q$ is a positive definite function.

**Construction of $z(\cdot)$ can be done using Bochner’s theorem.** This theorem says that every p.d. function on $\mathbb{R}^d$ can be written as the Fourier transform of a probability measure times a positive constant. So we have:

$$k(x_i, x_j) = Q(x_i - x_j) = E_{w \sim p}(a \exp(-iw^T(x_i - x_j)))$$

for some $p$ and some $a > 0$.

Now, since $Q(x_i - x_j)$ is a real value, we can rewrite the above as $E_{w \sim p}(a \cos(w^T(x_i - x_j)))$.

Since $\cos$ is in $[-1,1]$ using Hoeffding inequality we can approximate this expectation to within $\epsilon$ with a finite average:

$$\frac{1}{M} \sum_{m=1}^M \alpha \cos(w_m^T(x_i - x_j))$$  

where $w_1, \ldots, w_M \sim \mathcal{N}(0, \Sigma)$.

Rewriting this via trig identities we get:

$$\frac{1}{M} \sum_{m=1}^M \alpha \cos(w_m^T \Sigma(\cos(w_m^T x_i) \cos(w_m^T x_j) + \sin(w_m^T x_i) \sin(w_m^T x_j)) = z(x_i)^T z(x_j)$$

where $z(x_i) := \sqrt{m}(\cos(w_m^T x_i), \sin(w_m^T x_i), \ldots, \cos(w_m^T x_i), \sin(w_m^T x_i))$ and $w_1, \ldots, w_M$ are iid random draws from $p$.

Example: for Gaussian kernel $p$ is also Gaussian.

We proved approximation of the kernel value for a fixed pair $(x_i, x_j)$. To get this uniformly over all such pairs (not only in the training set) we cover $M \times M$, apply the above to each centre in the cover and take union bound. Finally extend to the whole space $M \times M$ using the fact that $cos$ is a smooth function.

Code and examples are available at: berkeley.intel-research.net/arahimi/random-features/

Other ways of constructing $z(\cdot)$ include hash kernels, for example [WDL+09].

10 years later...
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 $k \in O(s \log d)$. It follows that w.h.p no information was lost by carrying out the random projection. Therefore 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.

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:

$$y = Rx, \ \ y \in \mathbb{R}^k, \ \ R \in M_{k \times d}, \ x \in \mathbb{R}^d, \ k \ll d.$$ 

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 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 $k \in O(s \log d)$. It follows that w.h.p no information was lost by carrying out the random projection. Therefore 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.

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 $\|\hat{x}\|_0 \leq s$.
- $R$ satisfies the restricted isometry property (RIP) of order $(2s, \delta_{2s})$ with $\delta_{2s} < 2/(3 + \sqrt{7/3}) \approx 0.4627$.

Then:

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

- 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

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

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

One can show that random projection matrices satisfying the JLL w.h.p also satisfy the RIP w.h.p provided that $k \in O(s \log d)$. [BDDW08] does this using JLL combined with a covering argument in the projected space, finally union bound over all possible $s$-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 $k \in O(s \log d)$. It follows that w.h.p no information was lost by carrying out the random projection. Therefore 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.

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 $k \in O(s \log d)$. It follows that w.h.p no information was lost by carrying out the random projection. Therefore 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 $RS = \{(Rx_i, y_i), \ldots, (Rx_N, y_N)\} \sim \mathcal{D}_R$.

Let $w_0$ be the best linear classifier in the data domain with low hinge loss and large margin (i.e., small $\|w_0\|$).

Then, w.p. $1 - \frac{1}{\delta}$ (over $R$):

$$
H_0(z_{RS}) \leq H_0(w_0) + O \left( \sqrt{\left\| w_0 \right\|^2 \left( L^2 + \frac{\log(1/\delta)}{N} \right)} \right)
$$

where $H_0(w) = E_{(x,y) \sim \mathcal{D}} [1 - yw^T x]$ is the true hinge loss of the classifier in its argument, and $L = \max_x \|x\|$.

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 of Sridharan et al. [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 (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**

- Background and Preliminaries
- Johnson-Lindenstrauss Lemma (JLL) and extensions
- Applications of JLL (1)
- Compressed Sensing
- Applications of JLL (2)
  - RP LLS Regression
  - Randomized low-rank matrix approximation
  - Randomized approximate SVM solver
- Beyond JLL and Compressed Sensing

**Compressive Linear Least Squares Regression**

Given $T = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ with $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$.

**Algorithm.**

1. Let $R$ be a $k \times d$ random matrix with entries $R_{ij} \sim \mathcal{N}(0,1)$, let $P := R/\sqrt{k}$, and project the data: $X^T P \sim \mathbb{R}^k$.
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. For full details see [MM09].

Here we detail the special case of ordinary least squares regression (OLS).

**Proof sketch:**

- Denote by $X$ the design matrix having $x_i$ in 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 $E[Y|X]$.

**Definitions** in $\mathbb{R}^d$.

- Squared loss: $L(w) = \frac{1}{2}E_{\mathbb{E}_R} \left( Y - Xw \right)^2$ (where $E_R$ denotes $E_{\mathbb{E}_R}$).
- Optimal predictor: $\beta = \arg \min L(w)$.
- Excess risk of an estimator: $R(\hat{\beta}) = L(\hat{\beta}) - L(\beta)$.

For linear regression this is: $\left( \beta^* - \beta \right)^T \Sigma \left( \beta^* - \beta \right)$ where $\Sigma = XX^T/N$.

**OLS estimator:** $\hat{\beta} := \arg \min_w \left\{ \frac{1}{N} \| Y - Xw \|_2^2 \right\}$

**Proposition:** OLS. If $\text{Var}(Y) \leq 1$ then $\text{E}_R \left( \frac{1}{N} \right) \leq \frac{1}{N}$. 

- R.J. Durrant & A. Kabán (U.Birmingham)
**Definitions in \( \mathbb{R}^k \).**
Square loss: \( L_p(w) = \frac{1}{N} E_Y [ \| Y - (X^T) w \|_2^2 ] \) (where \( E_Y \) denotes \( E_X \)). Optimal predictor: \( \hat{w} = \arg \min_w L_p(w) \).
RP-OLS estimator: \( \hat{\beta}_p = \arg \min_{\beta} \| Y - (X^T) \beta \|_2^2 \).

**Proposition: Risk bound for RP-OLS**
Assume \( \text{Var}(Y) \leq 1 \), and let \( P \) as defined earlier. Then, for \( k = O(\log(8N/\delta))/\epsilon^2 \) and any \( \epsilon, \delta > 0 \), w.p. \( 1 - \delta \) we have:
\[
E_Y[L_P(\hat{\beta}_p)] - L(\beta) \leq \frac{k}{N} + \| \beta \|_2^2 \| E \|_{\text{trace}} \epsilon^2^2
\]  

*Proof.* Applying Proposition OLS in \( \mathbb{R}^k \), we get:
\[
E_Y[R(\hat{\beta}_p)] \leq \frac{k}{N} \tag{6}
\]
Using definition of \( R(\cdot) \), \( E_Y[R(\hat{\beta}_p)] = E_Y[L_p(\hat{\beta}_p)] - L_p(\beta) \). By definition of optimal predictor, \( L_p(\beta) = \min L_p(\beta). \) We rewrite and bound RHS using JLL for \( N \) dot products:
\[
L_P(P\beta) = \frac{1}{N} E_Y[ \| Y - X^T P^T \beta \|_2^2 ] \tag{7}
\]
\[
= \frac{1}{N} E_Y[ \| Y - X \hat{\beta} \|_2^2 ] + \frac{1}{N} \| X \hat{\beta} - X^T P \beta \|_2^2 \tag{8}
\]
\[
\leq L(\hat{\beta}) + \frac{1}{N} \sum_{i=1}^{N} (x_0^T P \beta - x_0 \beta)^2 \tag{9}
\]
\[
\leq L(\hat{\beta}) + \frac{1}{N} \sum_{i=1}^{N} \| \beta \|_2^2 \| x_0 \|_2^2 \tag{10}
\]
Noting that \( \frac{1}{N} \sum_{i=1}^{N} \| x_0 \|_2^2 = \| E \|_{\text{trace}} \epsilon^2 \), and combining with eq. (6) gives the result.

---

**Low-rank Matrix Approximation**

**Problem definition:** Given a \( d \times n \) matrix \( A \), and an integer \( s = k + p \), where \( k = \text{rank}(A) \), \( p \) is an oversampling factor, find a \( d \times s \) orthonormal matrix \( B \) s.t. \( A \approx BB^T A \).

**Algorithm:**
1. Generate random matrix \( R \) of size \( n \times s \), by drawing i.i.d. entries from \( N(0,1) \).
2. \( B := \text{orth}(AR) \) columns of \( B \) form an orthonormal basis for the range of \( A \). (Can be done by Gram-Schmidt or QR decomposition)

**Theorem**
For \( B \) constructed as above, we have:
\[
E_p[ \| A - BB^T A \|_F ] \leq \left( 1 + \frac{k}{p} \right)^{1/2} \left( \sum_{j=1}^{s} \sigma_j^2 \right)^{1/2} \tag{11}
\]
By Eckart-Young theorem, last term on RHS is minimal Frobenius norm error for rank \( k \) approximation of \( A \).
More results in [HMT11, Mah11].

**Improved algorithm when spectrum decays slowly**

1. Generate random matrix \( R \) of size \( n \times s \), e.g. by drawing i.i.d. entries from \( N(0,1) \).
2. \( B := \text{orth}(AA^T)^{kT} \) \( A \) has the same eigenvectors as \( A \) but power iteration increases rate of SV decay.

**Application to SVD**

1. Construct \( B^2 \) as above, i.e. so that \( A \approx BB^T A \).
2. Let \( C := B^T A \)
3. Compute SVD of \( C \) \( C = \tilde{U} \Sigma V^T \)
4. Set \( U := \tilde{B} \).

**Extensions of Compressive Regression**

- Compressive reinforcement learning [GLMM10].
- Compressive regression for sparse data, \( k = O(s \log d) \) [FGPP12].

**Randomized approximate SVM solver**

In the (near-)separable case the dual formulation of SVM optimization problem is equivalent to minimizing distance between (reduced) convex hulls of classes \([BB00, KBH08]\). Minimizing distance between convex hulls belongs to class of abstract LP-type optimization problems - for fixed \( d \) there are linear time randomized algorithms e.g. [Sei91] for solving these.

---

**Figure:** Example application to music similarity prediction from \( d = 10^5 \), very sparse; \( N = 2000 \). Taken from [FGPP12]
Abstract LP-type Problems

**Definition [MS03]**
An abstract LP-type problem is a pair \((H, w)\) where:
- \(H\) a finite set of constraints.
- \(w : H^2 \rightarrow \mathbb{R} \cup (-\infty, \infty)\) an **objective function** to be minimized which satisfies, for any \(h \in H\) and any \(F \subseteq G \subseteq H\):
  - **Monotonicity:** \(w(F) \leq w(G) \leq w(H)\).
  - **Locality:** If \(w(F) = w(G) = w(F \cup h)\) then \(w(F) = w(G \cup h)\).

**Interpretation:** \(w(G)\) is the minimum value of a solution satisfying all constraints on \(G\).

Basis and Combinatorial Dimension

**Definitions: Basis, Combinatorial Dimension**

\(L = (H, w)\) abstract LP-type problem then:
- A **basis** for \(F \subseteq H\) is a minimal set of constraints \(B \subseteq F\) such that \(w(B) = w(F)\).
- The **combinatorial dimension** of \(L\) is the size of the largest basis. Combinatorial dimension examples for problems in \(\mathbb{R}^d\):
  - Smallest enclosing ball, \(d + 1\)
  - Linear program, \(d + 1\)
  - Distance between hyperplanes, \(d + 2\)

Solving LP-type problems

Let \(B\) be a basis for \(F \subseteq H\) and \(h \notin F\) a constraint. We need two primitive operations:
- Test for violation: Is \(w(B \cup h) > w(B)\)?
- Basis update: Set \(B' = \text{basis}(B \cup h)\).

Random sampling for LP-type problems

Input: \((H, w)\), Output: A basis \(B\) for \(H\) such that \(w^* = w(B)\).

**LP-type\((C, G)\)** \(G \subseteq H\), \(C\) some basis.

1. If \(G = C\)
   - return \(C\)
2. Else
   - \(S\) = random subset of \(H\) of size \(M\)
   - \(B = \text{basis}(I, S)\)
   - \(V = \text{violators}(G - S, B)\)
   - While \((|V| > 0)\)
     - \(R\) = a subset of size \(M - |B|\) chosen randomly from \(V\).
     - \(B' = \text{basis}(B, R)\)
     - \(V = \text{violators}(G - R, B')\)
   - End while
   - return \(B\)

How to leverage JLL to solve SVM?

Recall that SVM problem is to minimize \(\|z_1 - z_2\|\) subject to:
- \(z_1 \in \text{Conv}\{x_i, y_i = 1\}\) and \(z_2 \in \text{Conv}\{x_i : y_i = -1\}\)

How do we utilise random projections to solve this efficiently?

Sub-exponential algorithm for LP-type problems

**Matousek, Sharir and Welzl 1996 [MSW96]**
The expected running time of the algorithm LP-type is \(O\left(N^2 \cdot \sqrt{\log b}\right)\), where \(N = |\text{basis}(H)|\).

This gives a linear time algorithm for LP in fixed dimension. What about for SVM? Fast SVM solver of Suresh et al., NIPS 2007 [KBH08].
Running Time for Randomized SVM Solver

- Standard SVM solver has time complexity between $\mathcal{O}(N^2)$ (lower bound) and $\mathcal{O}(N^3)$ [BEW05].
- For data in $\mathbb{R}^d$ expected time complexity of the randomized solver is $\mathcal{O}(N \cdot \exp(2\sqrt{d\log k}))$.
- Taking $k \in \mathcal{O}(\frac{1}{\epsilon} \log N)$ we can preserve the margin, w.h.p, apart from a scaling factor of $1 - \epsilon$.
- Then our approximate SVM solver has expected time complexity: $\mathcal{O}(N \cdot \exp(2\sqrt{d\log k})) = \mathcal{O}(N \cdot \exp(2\sqrt{\log N \log \log N})) = \mathcal{O}(N^2)$. Note no explicit random projection of the data, instead choose $k$ large enough to guarantee w.h.p the solver can find a set of support vectors giving near-optimal margin.

Approaches not leveraging JLL or CS (1)

Recall our two initial problem settings:

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

Approaches not leveraging JLL or CS (2)

- What if we have many, many observations, $N \in \mathcal{O}(\exp(d))$ for example? Inefficient to use all of the data, and JLL guarantees now require $k \in \mathcal{O}(d)$ so no help there. Can we quantify the generalization error cost of RP without appealing to JLL or sparsity? RP-FLD or ‘Compressed FLD’

Guarantees without JLL

We can obtain guarantees for randomly-projected classification algorithms without directly applying the JLL, by using measure concentration and random matrix theoretic-based approaches. Tackling the problem in this way removes the dependency on the number of observations, but at the cost of a dependency on the data dimensionality or a related quantity.

We shall consider two specific examples; a simple linear classifier and classifier ensemble, namely:

- RP Fisher’s Linear Discriminant [DK10a, DK10b].
- An ensemble of RP-FLD classifiers [DK12b]

In these settings we can quantify the price we pay in exchange for lower algorithmic complexity, and our bounds will tighten in a natural way with increasing sample size.
Guarantee on Compressed FLD

**Theorem (Bound on Average Misclassification Error)**

Let \( \hat{h} \) be the FLD classifier learned from a fixed training set. Let \( x_i \sim \sum_{j=1}^{m} p_j \mathcal{N}(\mu_j, \Sigma) \), where \( \Sigma \in \mathcal{M}_{d \times d} \) is a full rank covariance matrix. Let \( R \in \mathcal{M}_{d \times m} \) be a random projection matrix with entries drawn i.i.d from the univariate Gaussian \( \mathcal{N}(0,1) \). Then the estimated misclassification error \( \hat{P}_R(x_i \rightarrow y_i) \) is bounded above by:

\[
\left( 1 + \frac{1}{4} g \left( \Sigma^{-1} \Sigma \right) \right) \left( \frac{1}{d} \| \mu_j - \mu_l \|_2^2 \right)^{1/2}
\]

with \( \mu_j \) the mean of the class from which \( x_i \) was drawn, estimated class means \( \hat{\mu}_0 \) and \( \hat{\mu}_1 \), model covariance \( \Sigma \), and

\[
g(Q) = 4 \cdot \frac{\lambda_{\text{med}}(Q)}{\lambda_{\text{med}}(4)} \cdot \left( 1 + \frac{\lambda_{\text{med}}(Q)}{\lambda_{\text{med}}(4)} \right)^{-2}.
\]

Proof Outline

- Bound expected generalization error in data space (w.r.t. data distribution) - Chernoff bounding via m.g.f of Gaussian gives tractable form for working in the RP space. Bounding via m.g.f admits sub-Gaussian data distributions.
- Optimize bound - this brings in condition on true and estimated class centres.
- Bound expected generalization error in data space (w.r.t. data distribution, picks of random projection matrix \( R \)) - simplify using tricks from matrix analysis, ultimately bound error via m.g.f of \( \chi^2 \). Bounding via m.g.f admits entries in \( R \) with sub-Gaussian distribution.

Compressed Fisher's Linear Discriminant

`Compressed FLD`: Learn the classifier and carry out the classification in the RP space. Interested in quantifying the effect of random projection on the performance of the FLD classifier. In particular average classifier performance, over the random picks of \( R \), when \( k < d \).

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 \( E[\cdot] \).

RP-FLD decision rule:

\[
\hat{h}(x_i) = 1 \left\{ \left( \hat{\mu}_0 - \hat{\mu}_1 \right)^T R^T (E[R^T R])^{-1} R \left( x_i - \left( \hat{\mu}_0 + \hat{\mu}_1 \right) \right) > 0 \right\}
\]
Guarantee on Compressed FLD (2)

Theorem (Bound on Average Misclassification Error [DK11])

Under the same conditions as (4), the estimated misclassification error \( \hat{P}(x_0|y_0) \) is bounded above by:

\[
\exp \left( -\frac{k}{2} \log \left( 1 + \frac{1}{8d} \| \hat{\mu}_1 - \hat{\mu}_0 \|^2 \frac{\lambda_{\min}(\Sigma^{-1})}{\lambda_{\max}(\Sigma^{-1})} \right) \right)
\]

(13)

Comments: Tighter than (4) sometimes (tight when \( \Sigma = \Sigma \), or when \( \Sigma \) or \( \Sigma \) are spherical) but less representative of error behaviour.

Ensembles of RP-based methods

Motivation:
- Reducing variance
- Adding robustness
- Dealing with singularity

Applications of RP Ensembles: Clustering in RP space [FB03]. Face recognition [GBN05]. Covariance estimation when \( N \ll d \) [MTS11]. Classification [SR09, DK12b].

Corollary to (4) - Sufficient Dimensionality

Corollary (Sufficient Projection Dimensionality)

Under the same conditions as (4), for an \( m+1 \)-class problem, in order that the probability of misclassification in the projected space remains below \( \epsilon \) it is sufficient to take:

\[
k \geq 8 \frac{d \lambda_{\max}(\Sigma)}{\min_{i,j} \| \hat{\mu}_i - \hat{\mu}_j \|^2} \cdot \frac{1}{\epsilon} \cdot \log(m/\epsilon)
\]

(14)

Comments:
Compare with [AV99] for 2-class perceptron:

\[
k = O \left( \frac{L}{\epsilon^2} \cdot \log(12N/\epsilon) \right)
\]

(15)

where \( L \) is the (squared) \( \varphi \) of the data \( L = \max_{i,j} N \| x_i - x_j \|_2 \) divided by the margin.

Ensembles of RP-FLD

Assume a two-class classification problem, with \( N \) real-valued \( d \)-dimensional training observations:

\[ T = \{ (x, y) : (x, y) \in \mathbb{R}^d \times \{0, 1\} \}^N \]

Furthermore assume that \( N \ll d \), which is a common situation in practice (e.g. medical imaging, genomics, proteomics, face recognition, etc.), and that the unknown data distribution is full rank i.e. \( \text{rank}(\Sigma) = d \). (Can relax to \( \text{rank}(\Sigma) > N - 2 \).)

Validation - Corollary to Theorem (4)

Experiment confirming theorem (4) and corollary: Error is estimated from 500 random query points, and remains about constant when \( k = 12 \log m, m+1 \) 7-separated unit variance Gaussian classes.

Challenges (1)

Problems:
Inferential issues: \( N \) is too small (for good estimation of model) \( \varphi \) \( d \) \( \iff \) \( d \) is too large \( \varphi \) \( N \).

Computational issues: \( \Sigma \) is singular (and must be inverted to construct classifier).

Solution: Compress data by random projection to \( \mathbb{R}^k, k \ll N \). (Can relax to \( k = d \).)
Challenges (2)
We just saw that for a single RP-FLD classification error grows nearly exponentially as $k \gg 1$.

Solution:
Recover performance using an ensemble of RP FLD classifiers.
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: Potential for substantial computational savings, e.g. inversion of covariance matrix using Gauss-Jordan $O\left(k^2d^2\right)$ -vs- $O\left(d^3\right)$ or Strassen algorithm $O\left(k^2807\right)$ -vs- $O\left(d^2807\right)$ where $k < d$.

Observation
We can rewrite decision rule as:

$$1\left\{ (\hat{\mu}_1 - \hat{\mu}_0)^T R \hat{\Sigma} R^T \right\}^{-1} R (x_q - \hat{\mu}_1 + \hat{\mu}_0) > 0$$

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

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

Our Questions
- Can we recover (or improve on) level of classification performance in data space, using the 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?

Proof Techniques (1)
Rows (and columns) of $R$ drawn from a spherical Gaussian, hence for any orthogonal matrix $U, R \sim RU$. Eigendecomposing $\hat{\Sigma} = U \Lambda U^T$ and using $UU^T = I$ we find that:

$$E \left[ R^T \left( R \hat{\Sigma} R^T \right)^{-1} R \right] = U E \left[ R^T \left( R \Lambda R^T \right)^{-1} R \right] U^T$$  

(17)

Furthermore since a matrix $A$ is diagonal if and only if $VAV^T = A$ for all diagonal orthogonal matrices $V = \text{diag}(\pm 1)$ we can similarly show that the expectation on RHS is diagonal. Now enough to evaluate the diagonal terms on RHS!

Proof Techniques (2)
Define $\rho := \text{rank}(\hat{\Sigma}) = \text{rank}(\hat{\Lambda})$.
Write $R$ as the concatenation of two submatrices: $R = [P|S]$ where $P$ is $k \times \rho$ and $S$ is $k \times d - \rho$. Decompose expectation on RHS of (17) as two diagonal blocks:

$$E \left[ R^T \left( R \Lambda R^T \right)^{-1} R \right] = E\left[ \begin{bmatrix} E[P^T \left( P \Lambda P^T \right)^{-1} P] & 0 \\ 0 & E[S^T \left( S \Lambda S^T \right)^{-1} S] \end{bmatrix} \right]$$

Finally estimate the remaining expectations.

Comment: For $1 \leq k \leq \rho - 2$ this expectation is evaluated exactly in [MTS11] using a complicated procedure. We are more interested in how it relates to characteristics of $\hat{\Sigma}$, so we prefer simply interpretable estimates.
Proof Techniques (3)

Work with positive semidefinite ordering: $A \succeq B \iff A - B$ is positive semidefinite (p.s.d = symmetric with all eigenvalues $\geq 0$).

Upper and lower bound the diagonal matrix expectation (17) in the p.s.d ordering with spherical matrices $\alpha_{\text{max}} \cdot I$, $\alpha_{\text{min}} \cdot I$ to bound its condition number in terms of data space parameters:

$$\alpha_{\text{max}} \cdot I \succeq E \left( R^T (R A R^T)^{-1} R \right) \succeq \alpha_{\text{min}} \cdot I$$

Where $\alpha = \alpha(k, \rho, \lambda_{\text{max}}, \lambda_{\text{min}}, d)$, $k$ is the projected dimensionality, $\rho = \text{rank}(\Sigma) = \text{rank}(\Sigma_{\text{diag}})$, $\lambda_{\text{max}}$ and $\lambda_{\text{min}}$ are respectively the greatest and least non-zero eigenvalues of $\Sigma$.

Experiments: Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>#samples</th>
<th>#features</th>
</tr>
</thead>
<tbody>
<tr>
<td>colon</td>
<td>Alon et al. [ABN+99]</td>
<td>62</td>
<td>2000</td>
</tr>
<tr>
<td>leukemia</td>
<td>Golub et al. [GST+99]</td>
<td>72</td>
<td>3571</td>
</tr>
<tr>
<td>leukemia large</td>
<td>Golub et al. [GST+99]</td>
<td>72</td>
<td>7129</td>
</tr>
<tr>
<td>prostate</td>
<td>Singh et al. [SFR+02]</td>
<td>102</td>
<td>6033</td>
</tr>
<tr>
<td>duke</td>
<td>West et al. [WBD+01]</td>
<td>44</td>
<td>7129</td>
</tr>
</tbody>
</table>

Theory (1): Regularization

For fixed training set $\rho$, $d$ are constant and $1 \leq k \leq d$ is the integer regularization parameter. There are three cases, and each implements a different regularization scheme:

1. $1 \leq k \leq \rho - 2$ Shrinkage regularization [LW04] in range of $\Sigma$. Ridge regularization [HTF01] in null space of $\Sigma$. As $k \gg \rho - 1$ less regularization.

2. $\rho + 2 \leq k \leq d$ Individual matrices in projected ensemble are singular, expectation is not. Pseudoinverting individual classifiers in the ensemble gives: No regularization in range of $\Sigma$. Ridge regularization in null space of $\Sigma$. As $k \gg \rho$ less regularization.

3. $\rho - 1 \leq k \leq \rho + 1$ Shrinkage regularization ($k = \rho - 1$) or no regularization ($k \in [\rho, \rho + 1)$) in range of $\Sigma$. No regularization in null space of $\Sigma$.

Experiments: Protocol

- Standardized features to have mean 0 and variance 1 and ran experiments on 100 independent splits. In each split took 12 points for testing, rest for training.

- For data space experiments on colon and leukemia used ridge-regularized FLD and fitted regularization parameter using 5-fold CV on the first five data splits following [MRW+02].

- For other datasets we used diagonal FLD in the data space (size, no sig. diff. in error on colon, leuk).

- RP base learners: FLDs with full covariance and no regularization when $k \leq \rho$ and pseudoinverted FLD when $k > \rho$.

- Compared performance with SVM with linear kernel as in [FM03].

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>100 RP-FLD</th>
<th>1000 RP-FLD</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>colon</td>
<td>13.58 ± 0.95</td>
<td>13.56 ± 0.88</td>
<td>16.58 ± 0.95</td>
</tr>
<tr>
<td>leuk</td>
<td>1.83 ± 0.36</td>
<td>1.83 ± 0.37</td>
<td>1.67 ± 0.36</td>
</tr>
<tr>
<td>leuk-lg</td>
<td>4.91 ± 0.70</td>
<td>3.25 ± 0.60</td>
<td>3.50 ± 0.46</td>
</tr>
<tr>
<td>prost</td>
<td>4.00 ± 0.76</td>
<td>8.00 ± 0.72</td>
<td>8.00 ± 0.72</td>
</tr>
<tr>
<td>duke</td>
<td>17.41 ± 1.27</td>
<td>16.58 ± 1.27</td>
<td>13.50 ± 1.10</td>
</tr>
</tbody>
</table>

Theory (2): Exact Error of the Ensemble

Theorem (Ensemble error with Gaussian classes)

Let $x^t \sim \sum_{y=0}^{\pi} \pi_y N(\mu_y, \Sigma)$, where $\Sigma \in M_{d \times d}$ is a full rank covariance matrix. Let $R \in M_{k \times d}$ be a random projection matrix with i.i.d. Gaussian entries and denote $S = R \Sigma R^T$. Then the exact error of the randomly projected ensemble classifier (16), conditioned on the training set, is given by:

$$\sum_{y=0}^{\pi} \sum_{i=0}^{d} \Phi \left( -\frac{1}{2} (\mu_y - \mu)^T S^{-1} (\mu_y - \mu) \right)$$

For the converged ensemble, substitute the expectation (17) for $S^{-1}$ above.

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>100 RP-FLD</th>
<th>1000 RP-FLD</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>colon</td>
<td>13.58 ± 0.95</td>
<td>13.56 ± 0.88</td>
<td>16.58 ± 0.95</td>
</tr>
<tr>
<td>leuk</td>
<td>1.83 ± 0.36</td>
<td>1.83 ± 0.37</td>
<td>1.67 ± 0.36</td>
</tr>
<tr>
<td>leuk-lg</td>
<td>4.91 ± 0.70</td>
<td>3.25 ± 0.60</td>
<td>3.50 ± 0.46</td>
</tr>
<tr>
<td>prost</td>
<td>4.00 ± 0.76</td>
<td>8.00 ± 0.72</td>
<td>8.00 ± 0.72</td>
</tr>
<tr>
<td>duke</td>
<td>17.41 ± 1.27</td>
<td>16.58 ± 1.27</td>
<td>13.50 ± 1.10</td>
</tr>
</tbody>
</table>
References II


References III


References IV


Appendix

Proposition JLL for dot products.

Let \( x_n, n = \{1 \ldots N\} \) and \( u \) be vectors in \( \mathbb{R}^d \) s.t. \( \|x_n\|,\|u\| \leq 1 \).

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 \( k \in O(8\epsilon^2 \log(4N/\delta)) \) w.p. at least \( 1 - \delta \) we have:

\[
\left| x_n^T u - (Rx_n)^T Ru \right| < \epsilon \quad (18)
\]
simultaneously for all \( n = \{1 \ldots N\} \).

Proof of JLL for dot products

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

\[
4(Rx_n)^T (Ru) = \|Rx_n + Ru\|^2 - \|Rx_n - Ru\|^2 \quad (19)
\]
\[
\geq (1 - \epsilon)\|x_n + u\|^2 - (1 + \epsilon)\|x_n - u\|^2 \quad (20)
\]
\[
= 4x_n^T u - 2(\|x_n\|^2 + \|u\|^2) \quad (21)
\]
\[
\geq 4x_n^T u - 4\epsilon \quad (22)
\]

Hence, \( (Rx_n)^T (Ru) > x_n^T u - \epsilon \) 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 \( (Rx_n)^T (Ru) \leq x_n^T u + \epsilon \) except w.p. \( 2 \exp(-k\epsilon^2/8) \).

Put together, \( (Rx_n)^T (Ru) - x_n^T u \leq \epsilon \cdot \frac{\sqrt{x_n^T u + \epsilon^2}}{\epsilon} \leq \epsilon \) 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.\((18)\) must hold except w.p. \( 4N \exp(-k\epsilon^2/8) \). Finally, solving for \( \delta \) we obtain that \( k \geq \frac{8}{\delta} \log(4N/\delta) \).