Structured Spinners: Approach for fast and large-scale Machine Learning computations

Anne Morvan $^{1,2}$

Joint work with: M. Bojarski$^3$, A. Choromanska$^4$, K. Choromanski$^5$, F. Fagan$^6$, C. Gouy-Pailler$^2$, N. Sakr$^6$, T. Sarlós$^4$, J. Atif$^1$

$^1$Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE

$^2$CEA, LIST

$^3$NVIDIA

$^4$NYU

$^5$Google Brain Robotics

$^6$Columbia University

April 6, 2018
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
5. Deep neural networks as application in the adaptive setting
6. Conclusion
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
5. Deep neural networks as application in the adaptive setting
6. Conclusion
Approach for fast and large-scale Machine Learning computations

Structured Spinners’ principle

- Replace $Gx$ by $G_{struct}x$ in large-scale ML applications containing random projections, $G, G_{struct} \in \mathbb{R}^{m \times n}$
- Significant speedups $O(mn) \rightarrow O(n \log m)$
- Memory space savings $O(mn) \rightarrow O(n \log m)$
- Almost no loss of accuracy

Locality-Sensitive Hashing (LSH)

- Kernel approximation
- Newton sketches
- Deep neural networks
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
5. Deep neural networks as application in the adaptive setting
6. Conclusion
Why random projections?

When all your data do not fit into memory...

- Massive data + high dimensionality
- Ex: finding near duplicates in holiday photos
Observation

- Lot of high dimensional data lie on a lower-dimensional manifold
- Concept of intrinsic dimension

Perform dimensionality reduction / find a suitable projection onto a lower dimensional space
Why random projections?

Dimensionality reduction

\[ \| \Phi x_i - \Phi x_j \|_2 \approx \| x_i - x_j \|_2, \quad \| \Phi x_i \|_2 \approx \| x_i \|_2 \]

Desirable properties of the projection
- Near isometric embedding
- \((1 \pm \epsilon)\) distortion
- Distance and angle preserved between points

Classical projections
- Principal Component Analysis (PCA)
- Random Projection (RP)
Random projections: theoretical justification

Founder Lemma: [Johnson and Lindenstrauss, 1984]:

Let \( \epsilon \in ]0, 1[ \), \( \mathcal{X} = \{ \mathbf{x}_1, \ldots, \mathbf{x}_N \} \subset \mathbb{R}^n \).

Let \( m \in \mathbb{N} \), s.t. \( m \geq C\epsilon^{-2} \log N \).

Then there exists a linear map \( \Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m \) s.t. :

\[
\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}, \quad (1 - \epsilon) \| \mathbf{x}_i - \mathbf{x}_j \|_2 \leq \| \Phi \mathbf{x}_i - \Phi \mathbf{x}_j \|_2 \leq (1 + \epsilon) \| \mathbf{x}_i - \mathbf{x}_j \|_2.
\]

One can take \( \Phi = \text{Random} \) (near orthonormal) which works with high probability.
Random projections applications

- Linear embedding / Dimensionality reduction,
- Approximate nearest neighbor algorithms, e.g.:
  - Random Projection Trees,
  - Locality Sensitive Hashing-based algorithms.
- Compressed sensing,
- Efficient kernel computations via random feature maps,
- Convex optimization algorithms,
- Quantization techniques,
- etc.

⇒ information retrieval, similarity search, classification, clustering.
Why random projections?

**Brief random projections evolution**

Φ: Dense i.i.d. distribution
- [Frankl and Maehara, 1987]: \( \Phi_{i,j} \sim \mathcal{N}(0, \frac{1}{\sqrt{m}}) \)
- [Achlioptas, 2003]: \( \Phi_{i,j} \sim \{-1, 1\} \) uniformly

Φ: Sparse i.i.d. distribution
- [Kane and Nelson, 2010]: \#nonzero entries in \( \Phi = O(n \log N/\epsilon) \),
- Fast Johnson-Lindenstrauss Transform - FJLT
  - [Ailon and Chazelle, 2006]: \( \Phi = \text{PHD} \)
    - \( P_{i,j} = \)
      \[
      \begin{cases}
      \sim \mathcal{N}(0, \frac{1}{q}) & \text{with probability } q \\
      0 & \text{with probability } 1 - q
      \end{cases}
      \]
    - \( H \) normalized Hadamard,
    - \( D \) with independent Rademacher \((\pm 1)\) entries.

\[
H_0 = 1
\]
\[
H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}
\]
\[
H_m = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{pmatrix}
\]
Why random projections?

Classical structured matrices

\[ H_0 = 1 \]

\[ H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \]

\[ H_m = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{pmatrix} \]

\[ G_{\text{skew-circ}} = \begin{pmatrix} a & -b & -c & -d & -e \\ e & a & -b & -c & -d \\ d & e & a & -b & -c \\ c & d & e & a & -b \\ b & c & d & e & a \end{pmatrix} \]

\[ G_{\text{Toeplitz}} = \begin{pmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ j & h & g & f & a \end{pmatrix} \]

\[ K = R_1 \otimes R_2 \otimes \ldots \otimes R_m \in \mathbb{R}^{2^m \times 2^m} \]

\[ R_i \in \mathbb{R}^{2 \times 2} \text{ or } R_i \in \{-1, 1\}^{2 \times 2} \]

\[ R_i R_i^T = R_i^T R_i = I_2 \]
And what is about our *Structured Spinners*-family?

Main purpose of *Structured Spinners*-family
Speed up several machine learning algorithms relying on unstructured random matrices with almost no loss of accuracy!

Arguments

- **Speedups:**
  - Fast Fourier Transform (FFT) or Fast Hadamard Transform (FHT):
    \( O(n \log m) \) instead of \( O(mn) \) for matrix-vector product.

- **Less storage:**
  - \( H \) is not stored
  - Sparse matrices: diagonal ones
  - Structured matrices: \( n \times n \)-circulant one \( \Rightarrow \) only \( n \) parameters (linear)
  - Structured matrices with \( \pm 1 \) entries: only bits.
Why random projections?

Some of state-of-the-art for structured matrices in applications (1/2)

Approximate Nearest Neighbor search (ANN), e.g.:
- [Andoni et al., 2015]: Locality-Sensitive Hashing (LSH), $HD_3 HD_2 HD_1$.

Quantization, e.g.:
- [Yu et al., 2014]: $G_{circulant}$

$$G_{circulant} = \begin{pmatrix}
a & b & c & d & e \\
e & a & b & c & d \\
d & e & a & b & c \\
c & d & e & a & b \\
b & c & d & e & a \\
\end{pmatrix}$$
Why random projections?

Some of state-of-the-art for structured matrices in applications (2/2)

Kernel approximation via random feature maps
[Rahimi and Recht, 2007, Rahimi and Recht, 2009]

- [Le et al., 2013]: ”FastFood”, $\frac{1}{\sqrt{n}} \text{SHGPHB}$,

- [Feng et al., 2015]: $\pm 1 \text{G}_{\text{circulant}}$,

- [Choromanski and Sindhwani, 2016]: ”$P$-model”, and Toeplitz-like semi-Gaussian matrices,

$$\sum_{i=1}^{r} \text{Circ}[g^i] \text{SkewCirc}[h^i] \text{ for some } \{g^i, h^i\}_{i=1}^{r} \in \mathbb{R}^n.$$
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
5. Deep neural networks as application in the adaptive setting
6. Conclusion
Definition of *Structured Spinners* family

**Structured Spinners for 3 blocks**

\[
G \rightarrow G_{\text{struct}}
\]

\[
G_{\text{struct}} = M_3 M_2 M_1 \in \mathbb{R}^{n \times n},
\]

where matrices \(M_1, M_2\) and \(M_3\) satisfy 3 conditions.

**Examples**

- \([G_{\text{circ}} \mid G_{\text{skew-circ}} \mid G_{\text{Toeplitz}} \mid G_{\text{Hankel}}]D_2 HD_1\),
- \(\sqrt{n} HD_{g_1,\ldots,g_n} HD_2 HD_1\),
- \(\sqrt{n} HD_3 HD_2 HD_1\).
Role of each *Structured Spinner* block

$$G_{\text{struct}} = M_3 M_2 M_1$$
Role of each *Structured Spinner* block - $M_1$

$$G_{struct} = M_3 M_2 M_1$$

$\hookrightarrow$ Balances data.
Role of each *Structured Spinner* block - $\mathbf{M}_2$

$$
\mathbf{G}_{\text{struct}} = \mathbf{M}_3 \mathbf{M}_2 \mathbf{M}_1
$$

$\hookrightarrow$ Makes the rows of the final matrix almost independent.
Role of each \textit{Structured Spinner} block - $M_3$

\[ G_{\text{struct}} = M_3 M_2 M_1 \]

\rightarrow \text{Budget of randomness.}
Condition 1 - Balanceness

**Condition 1:** $M_1$ and $M_2M_1$ are $(\delta(n), p(n))$-balanced isometries.

**Definition:** $(\delta(n), p(n))$-balanced matrices

A randomized matrix $M \in \mathbb{R}^{n \times n}$ is $(\delta(n), p(n))$-balanced if it represents an isometry and for every $x \in \mathbb{R}^n$ with $\|x\|_2 = 1$ we have:

$$\mathbb{P}[\|Mx\|_\infty > \frac{\delta(n)}{\sqrt{n}}] \leq p(n).$$

**Example**

$M_1 = HD_1$, since $HD_1$ is $(\log(n), 2ne^{-\frac{\log^2(n)}{8}})$-balanced.
Condition 2 - Decorrelation (1/2)

Condition 2: \( M_2 = V(W^1, \ldots, W^n)D_{\rho_1, \ldots, \rho_n} \) for some \((\Lambda_F, \Lambda_2)\)-smooth set \( W^1, \ldots, W^n \in \mathbb{R}^{k \times n} \) and some i.i.d sub-Gaussian random variables \( \rho_1, \ldots, \rho_n \) with sub-Gaussian norm \( K \).

\[
V(W^1, \ldots, W^n) = \begin{pmatrix} W^1 \\ W^2 \\ \vdots \\ W^n \end{pmatrix} \quad D_{\rho_1, \ldots, \rho_n} = \begin{pmatrix} \rho_1 & 0 & \ldots & 0 \\ 0 & \rho_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & \rho_n \end{pmatrix}
\]

Typically, \( K = 1 \).
Condition 2 - Decorrelation (2/2)

Definition: \((\Lambda_F, \Lambda_2)\)-smooth sets

A deterministic set of matrices \(\mathbf{W} = \{\mathbf{W}^1, ..., \mathbf{W}^n\}\), where \(\mathbf{W}^i = \{w^i_k \}_{k,l \in \{1, ..., n\}}\) is \((\Lambda_F, \Lambda_2)\)-smooth if:

- for \(i = 1, ..., n\):
  
  \[ \|\mathbf{W}^i\|_2 = \ldots = \|\mathbf{W}^i_l\|_2 = \ldots = \|\mathbf{W}^i_n\|_2 \]

- for \(i \neq j\) and \(l = 1, ..., n\):

  \[ \text{max}_{i,j} \| (\mathbf{W}^j)^T \mathbf{W}^i \|_F \leq \Lambda_F \text{ and } \max_{i,j} \| (\mathbf{W}^j)^T \mathbf{W}^i \|_2 \leq \Lambda_2. \]
**Condition 3 - Budget of randomness**

**Condition 3:** $M_3 = C(r, n)$ for $r \in \mathbb{R}^k$, where $r$ is random Rademacher ($\pm 1$ entries) or Gaussian.

$$M_3 = \begin{pmatrix} r_1 & \ldots & r_k & 0 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & \ldots & 0 & r_1 & \ldots & r_k & 0 & \ldots & \ldots & \ldots & 0 \\ \vdots & & \vdots & & & & \vdots & & \vdots & & \\ 0 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 & r_1 & \ldots & r_k \end{pmatrix}$$
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
   - Locality-Sensitive Hashing (LSH)
   - Kernel approximation
   - Newton sketches
5. Deep neural networks as application in the adaptive setting
6. Conclusion
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
   - Locality-Sensitive Hashing (LSH)
   - Kernel approximation
   - Newton sketches
5. Deep neural networks as application in the adaptive setting
   - Deep neural networks and parameters
   - Some existing structured neural networks
   - Experiments
6. Conclusion
Locality-Sensitive Hashing (LSH) for Nearest Neighbor (NN) search

NN search naive approach
- Linear search.
- Prohibitive cost when lots of high dimensional data.
- **Solution**: Approximate Nearest Neighbor (ANN) search with LSH algorithm in sublinear time.

LSH: Two phases
- Build a data structure (**hash table**) for fast lookup.
- NN search phase: query the database with query point $q$. 
Some applications in the randomized setting

Locality-Sensitive Hashing (LSH)

Hashing vs LSH

Hashing principle

- Mapping data from a potential high dimensionality to a fixed-size hash value.
- Fast lookup in a database.

LSH principle

- Exploiting collision probabilities.

<table>
<thead>
<tr>
<th>keys</th>
<th>hash function</th>
<th>hashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>Lisa Smith</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>Sam Doe</td>
<td></td>
<td>02</td>
</tr>
<tr>
<td>Sandra Dee</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>
**LSH in details**

**Hash value computation**

- Hash value $h$ of a point $x \in \mathbb{R}^n$ is a combination of $k$ hash function results $h_i$, $i = 1...k$ s.t. $h_i = f(A_i x)$ with $A_i \in \mathbb{R}^{m \times n}$ a projection matrix s.t. $m \ll n$.

- **Example**: Concatenation: $h = h_1 h_2 ... h_k$.
Some applications in the randomized setting

Locality-Sensitive Hashing (LSH)

LSH in details

L hash tables

\[
\begin{align*}
    h_{\mathcal{H}_1}(p) &= f(A_{1_{\mathcal{H}_1}}(p)) \ldots f(A_{k_{\mathcal{H}_1}}(p)) \\
    h_{\mathcal{H}_L}(p) &= f(A_{1_{\mathcal{H}_L}}(p)) \ldots f(A_{k_{\mathcal{H}_L}}(p))
\end{align*}
\]
Some applications in the randomized setting

Locality-Sensitive Hashing (LSH)

ANN search with LSH

ANN search

- Hash query $q$.
- Determine pool of candidates (in green).
- Linear scan in the pool of candidates.
Cross-polytope LSH

Cross-polytope from [Terasawa and Tanaka, 2007]

\[ h_i(x) = f\left( \frac{Gx}{\|Gx\|_2} \right) \]

- \( h = (2m)^{k-1} h_1 + ... + h_k \).
- \( G \in \mathbb{R}^{m \times n} \) a random matrix with i.i.d. Gaussian entries.
- \( f(y) \) returns the closest vector to \( y \) from the set \( \{ \pm 1e_i \}_{1 \leq i \leq m} \), where \( \{ e_i \}_{1 \leq i \leq m} \) stands for the canonical basis.

- State-of-the-art cross-polytope LSH [Andoni et al., 2015]
  \( G \rightarrow HD_3 HD_2 HD_1 \).
- Our variant: \( G_{struct} = M_3 M_2 M_1 \) + theoretical guarantees.
Cross-polytope LSH experiment with *Structured Spinners*

**Experimental protocol**

- Plot $Pr[h(p) = h(q)]$ as a function of $dist(p, q)$,
- 100 runs,
- $k = 1$,
- Draw points from the hypersphere $\Rightarrow \max_{p, q} dist(p, q) = \sqrt{2}$,
- 20000 points per interval of distance: $[0, 0.2), [0.2, 0.4), [0.4, 0.6), [0.6, 0.8), [0.8, 1.2), [1.2, \sqrt{2}]$,
- $n = 256$,
- $m = 64$. 
Cross-polytope LSH experiment with Structured Spinners

Collision probabilities with cross–polytope LSH

Distance
Collision probability

G
G_{circ}K_2K_1
G_{Toeplitz}D_2HD_1
G_{skew−circ}D_2HD_1
HD_{g_1, g_2, ..., g_n}HD_2HD_1
HD_3HD_2HD_1

0.01 0.05 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 √2
Cross-polytope LSH experiment with Structured Spinners
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
   - Locality-Sensitive Hashing (LSH)
   - Kernel approximation
   - Newton sketches
5. Deep neural networks as application in the adaptive setting
   - Deep neural networks and parameters
   - Some existing structured neural networks
   - Experiments
6. Conclusion
Kernel methods

Principle

- **Goal**: To solve nonlinear problems with linear methods.
- **How?** Map all data into a higher dimensional (possibly infinite) dot product space $\nu$ with feature map $\phi : \chi \rightarrow \nu$.
- **Access to mapped data**:

$$\kappa(x, y) = \langle \phi(x), \phi(y) \rangle$$
Kernel approximation in Support Vector Machines (SVM)

Decision evaluation in SVM: the "kernel trick"

\[
f(x) = \langle w, \phi(x) \rangle = \left\langle \sum_{i=1}^{N'} \alpha_i \phi(x_i), \phi(x) \right\rangle = \sum_{i=1}^{N'} \alpha_i \kappa(x_i, x)
\]

\(N'\): number of nonzero \(\alpha_i\) = number of "support vectors"

Why approximation?

- **Problem**: evaluating \(f\) cost increases as the dataset grows.
  
  \(N\) number of training samples.

- **Kernel or Gram matrix** \(K\):
  
  \[
  K_{ij} = \kappa(x_i, x_j)
  \]

  \(\Rightarrow\) storage cost: \(O(N^2)\)
Kernel approximation via random feature maps

**Random Kitchen Sinks** [Rahimi and Recht, 2007, Rahimi and Recht, 2009]

\[
\langle z(x), z(y) \rangle \in \mathbb{R}^k \\
\approx \langle \phi(x), \phi(y) \rangle \in \mathbb{R}^D
\]

where \( k \gg n \); \( D \) high, possibly infinite.

- \( \langle z(x), z(y) \rangle \approx \langle \phi(x), \phi(y) \rangle \)
- \( \phi(x) = \frac{1}{\sqrt{k}} s(Gx) \)
- random Gaussian matrix \( G \in \mathbb{R}^{k \times n} \) with \( k \gg n \), \( k = O(n\epsilon^{-2} \log \frac{1}{\epsilon^2}) \)
- \( s \) is a nonlinearity function.

Still a problem...

- Storage of \( G \): \( O(kn) \)
- Computation of \( Gx \): \( O(kn) \)

Solution

- Storage of \( G_{struct} \): \( O(k \log n) \)
- Computation of \( G_{struct}x \): \( O(k \log n) \)
Some applications in the randomized setting

Kernel approximation

Experimental protocol for kernel approximation (1/2)

\(A \in \mathbb{R}^{k \times n}\) with \(k \gg n\),

Gaussian kernel

- \(\kappa_G(x, y) = e^{-\frac{||x-y||^2}{2\sigma^2}}\),
- \(\tilde{\kappa}_G(x, y) = \frac{1}{k} s(Ax)^T s(Ay)\) with \(s(x) = e^{-\frac{ix}{\sigma}}\) applied pointwise.

Angular kernel

- \(\kappa_0(x, y) = 1 - \frac{\theta}{\pi}\) with \(\theta = \cos^{-1}\left(\frac{x^T y}{||x|| ||y||}\right)\),
- \(\tilde{\kappa}_0(x, y) = 1 - \frac{dist_{Hamming}(s(Ax), s(Ay))}{k}\) with \(s(x) = \text{sign}(x)\) applied pointwise.
Experiments for kernel approximation (1/4)

Some applications in the randomized setting

Kernel approximation

Speedups with Gaussian kernel

<table>
<thead>
<tr>
<th>Matrix dimensions</th>
<th>$2^9$</th>
<th>$2^{10}$</th>
<th>$2^{11}$</th>
<th>$2^{12}$</th>
<th>$2^{13}$</th>
<th>$2^{14}$</th>
<th>$2^{15}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{\text{Toeplitz}}D_2\text{HD}_1$</td>
<td>x1.4</td>
<td>x3.4</td>
<td>x6.4</td>
<td>x12.9</td>
<td>x28.0</td>
<td>x42.3</td>
<td>x89.6</td>
</tr>
<tr>
<td>$G_{\text{skew-circ}}D_2\text{HD}_1$</td>
<td>x1.5</td>
<td>x3.6</td>
<td>x6.8</td>
<td>x14.9</td>
<td>x31.2</td>
<td>x49.7</td>
<td>x96.5</td>
</tr>
<tr>
<td>$\text{HD}_{g_1,\ldots,g_n}\text{HD}_2\text{HD}_1$</td>
<td>x2.3</td>
<td>x6.0</td>
<td>x13.8</td>
<td>x31.5</td>
<td>x75.7</td>
<td>x137.0</td>
<td>x308.8</td>
</tr>
<tr>
<td>$\text{HD}_3\text{HD}_2\text{HD}_1$</td>
<td>x2.2</td>
<td>x6.0</td>
<td>x14.1</td>
<td>x33.3</td>
<td>x74.3</td>
<td>x140.4</td>
<td>x316.8</td>
</tr>
</tbody>
</table>

ex: $\text{HD}_3\text{HD}_2\text{HD}_1$, $k = 2^{15}$, 1.382s $\rightarrow$ 4363µs in comparison with $G$
Experimental protocol for kernel approximation (2/2)

Measure of accuracy

- 10 runs,
- Dataset: USPST,
- $16 \times 16$ grayscale images,
- 2007 points of dimensionality 256 ($n = 256$),
- $\sigma = 9.4338$,
- Plots Gram reconstruction error: $\frac{\|K - \tilde{K}\|_F}{\|K\|_F}$,
- $K_{i,j} = \kappa(x_i, x_j)$. 
Experiments for kernel approximation (2/3)

Gram matrix reconstruction error
USPST dataset for the Gaussian kernel

\[
\frac{\|K - \tilde{K}\|_F}{\|K\|_F}
\]

\[K_{i,j} = \kappa(x_i, x_j)\]
Experiments for kernel approximation (3/3)

Gram matrix reconstruction error
USPST dataset for the angular kernel

\[
\frac{\| K - \tilde{K} \|_F}{\| K \|_F}
\]

\[K_{i,j} = \kappa(x_i, x_j)\]

\[\text{Number of random features}\]

\[\text{Gram matrix reconstruction error}\]

- $G$
- $G_{\text{circ}} K_2 K_1$
- $G_{\text{Toeplitz}} D_2 H D_1$
- $G_{\text{skew-circ}} D_2 H D_1$
- $H D_{g_1, g_2, ..., g_n} H D_2 H D_1$
- $H D_3 H D_2 H D_1$
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
   - Locality-Sensitive Hashing (LSH)
   - Kernel approximation
   - Newton sketches
5. Deep neural networks as application in the adaptive setting
   - Deep neural networks and parameters
   - Some existing structured neural networks
   - Experiments
6. Conclusion
Unconstrained convex optimization with Newton step gradient descent

minimize $f(x)$

where $f : \mathbb{R} \to \mathbb{R}$ is convex and twice continuously differentiable.

- $x^{(t+1)} = x^{(t)} - \mu^{(t)} \nabla^2 f(x)^{-1} \nabla f(x)$

- **Newton decrement:**
  \[ \lambda = (\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x))^{1/2} \]

  ↪ Used as stopping criterion + in backtracking line search:
  \[ \text{while } \lambda^2 = -\nabla f(x)^T \Delta x > \epsilon \]
Principle of Newton sketch’s algorithm
[Pilanci and Wainwright, 2015]

Newton sketch’s algorithm [Pilanci and Wainwright, 2015]

If analytic expression for the square root of the Hessian matrix:

\[ x^{(t+1)} = x^{(t)} - \mu \underbrace{\left( S^{(t)} \left( \nabla^2 f(x^{(t)}) \right)^{1/2} \right)^T}_{(SM)^T} \underbrace{S^{(t)} \left( \nabla^2 f(x^{(t)}) \right)^{1/2}}_{SM}^{-1} \nabla f(x^{(t)}) \]

where \( S^{(t)} \in \mathbb{R}^{m \times n} \) is a sequence of isotropic sketchs matrices.
Large scale logistic regression problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

with $f(x) = \sum_{i=1}^{N} \log(1 + \exp(-y_ia_i^T x))$

$N$ observations $(a_i, y_i)_{i=1\ldots N}$

s.t. $a_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$. 
Example for Newton sketch’s algorithm (2/2)

Analytic expressions for the gradient and the Hessian matrix

- \( \nabla f(x(t)) = \sum_{i=1}^{n} (\frac{1}{1+\exp(-y_i a_i^T x)} - 1) y_i a_i \in \mathbb{R}^n, \)

- \( \nabla^2 f(x(t)) = A^T \text{diag}(\frac{1}{1+\exp(-a_i^T x)}(1 - \frac{1}{1+\exp(-a_i^T x)})) A \in \mathbb{R}^{n \times n}, \)

\[ A = [a_1^T \ldots a_N^T] \in \mathbb{R}^{N \times n}, \text{ with } N \gg n, \]

- We set

\[ \nabla^2 f(x(t))^{1/2} = \text{diag}(\frac{1}{1+\exp(-a_i^T x)}(1 - \frac{1}{1+\exp(-a_i^T x)}))^{1/2} A \in \mathbb{R}^{N \times n}. \]
Comparison

- **Exact Newton:**

\[
\nabla^2 f(x)^{-1} \\
\n\nabla^2 f(x^{(t)}) = A^T \text{diag}(\frac{1}{1+\exp(-a_i^T x)}(1 - \frac{1}{1+\exp(-a_i^T x)}))A
\]

\[
\text{Cost} = O(Nn^2 + n^3) \; (n \ll N)
\]
Newton sketch’s algorithm, complexity analysis (2/2)

Comparison

- **Exact Newton:**
  \[
  \text{Cost} = O(Nn^2 + n^3) \quad (n \ll N)
  \]

- **Sketching:**
  \[
  \left( (S(t) (\nabla^2 f(x(t)))^{1/2})^T (SM)^T \right) S(t) (\nabla^2 f(x(t)))^{1/2} \left( SM \right)^{-1}
  \]

  \[
  \nabla^2 f(x(t))^{1/2} = \text{diag} \left( \frac{1}{1 + \exp(-a_i^T x)} \left( 1 - \frac{1}{1 + \exp(-a_i^T x)} \right) \right) ^{1/2} A \in \mathbb{R}^{N \times n}
  \]

  \[
  \text{Cost} = O(3nN \log N + mn^2 + n^3) \quad \text{with} \quad m \ll N
  \]

**Critical issue:** when is \( O(3nN \log N + mn^2) \) better than \( O(Nn^2) \)?
Experimental results (1/2)

Convergence analysis

Structured Spinners
- Exact Newton
- $G_{\text{circ}}D_2HD_1$
- $G_{\text{Toeplitz}}D_2HD_1$
- $G_{\text{Hankel}}D_2HD_1$
- $HD_3HD_2HD_1$

Anne Morvan
Expedia seminar, Geneva
April 6, 2018 53 / 71
Experimental results (2/2)

Hessian computation time

- Log-log plot showing the computation time (wall-clock time) for different structured spinners and matrices.
- The x-axis represents the number of samples (N), while the y-axis represents the wall-clock time.
- The plot compares the performance of Exact Newton, $G_{\text{circ}}D_2H_1$, $G_{\text{Toeplitz}}D_2H_1$, $G_{\text{Hankel}}D_2H_1$, and $HD_3HD_2H_1$.

Structured Spinners

- Exact Newton
- $G_{\text{circ}}D_2H_1$
- $G_{\text{Toeplitz}}D_2H_1$
- $G_{\text{Hankel}}D_2H_1$
- $HD_3HD_2H_1$
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
5. Deep neural networks as application in the adaptive setting
   - Deep neural networks and parameters
   - Some existing structured neural networks
   - Experiments
6. Conclusion
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
   - Locality-Sensitive Hashing (LSH)
   - Kernel approximation
   - Newton sketches
5. Deep neural networks as application in the adaptive setting
   - Deep neural networks and parameters
   - Some existing structured neural networks
   - Experiments
6. Conclusion
Deep neural networks and parameters

- Explosion of deep neural networks (e.g. convolutional networks) applications: with billions of parameters!
- Standard architecture: convolutional and fully-connected layers
- Convolutional layers: most of the computational effort
- Fully-connected layers: 90% of the parameters!
- Necessity to reduce the number of parameters for deployment on embedded mobile devices (speed up train + test time)

\[ s(A_1x) \quad s(A_2x) \]
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
   - Locality-Sensitive Hashing (LSH)
   - Kernel approximation
   - Newton sketches
5. **Deep neural networks as application in the adaptive setting**
   - Deep neural networks and parameters
   - Some existing structured neural networks
   - Experiments
6. Conclusion
Related work

Some structured neural networks

- Deep Fried Convnets [Yang et al., 2015]: \textbf{SHGPHB} (Fastfood)

- [Moczulski et al., 2016]: \textbf{ACDC}^{-1}, \textbf{A}, \textbf{D} diagonal, \textbf{C} is the discrete cosine transform

- [Denil et al., 2013]: $\textbf{U}\textbf{V}$, fix $\textbf{U} \in \mathbb{R}^{m \times r}$ and learn $\textbf{V} \in \mathbb{R}^{r \times n}$, $r \ll m, n$

- [Sainath et al., 2013]: low-rank matrix factorization $\textbf{U}\textbf{V}$

- [Xue et al., 2013]: $\textbf{U}(\Sigma \textbf{V}^T)$, $\textbf{U} \in \mathbb{R}^{m \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$, $\textbf{V} \in \mathbb{R}^{n \times r}$ (after training)
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
   - Locality-Sensitive Hashing (LSH)
   - Kernel approximation
   - Newton sketches
5. Deep neural networks as application in the adaptive setting
   - Deep neural networks and parameters
   - Some existing structured neural networks
   - Experiments
6. Conclusion
Deep neural networks as application in the adaptive setting

Experiments

Structured MLP with 2 fully-connected layers on MNIST

MLP neural network error

Running time (in $\mu$s), $h$: size of the hidden layers

<table>
<thead>
<tr>
<th>h</th>
<th>$2^4$</th>
<th>$2^5$</th>
<th>$2^6$</th>
<th>$2^7$</th>
<th>$2^8$</th>
<th>$2^9$</th>
<th>$2^{10}$</th>
<th>$2^{11}$</th>
<th>$2^{12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>unstructured</td>
<td>42.9</td>
<td>51.9</td>
<td>72.7</td>
<td>99.9</td>
<td>163.9</td>
<td>350.5</td>
<td>716.7</td>
<td>1271.5</td>
<td>2317.4</td>
</tr>
<tr>
<td>HD$_3$HD$_2$HD$_1$</td>
<td>109.2</td>
<td>121.3</td>
<td>109.7</td>
<td>114.2</td>
<td>117.4</td>
<td>123.9</td>
<td>130.6</td>
<td>214.3</td>
<td>389.8</td>
</tr>
</tbody>
</table>
Structured convolutional network on MNIST

- Conv. layer with filter size $5 \times 5$, 4 feature maps + ReLU + Max Pooling (region $2 \times 2$ and step $2 \times 2$)
- Conv. layer with filter size $5 \times 5$, 6 feature maps + ReLU + Max Pooling (region $2 \times 2$ and step $2 \times 2$)
- FC layer ($h$ outputs) + ReLU
- FC layer (10 outputs)
- LogSoftMax.
Plan

1. Introduction
2. Why random projections?
3. Brief review of Structured Spinners family
4. Some applications in the randomized setting
5. Deep neural networks as application in the adaptive setting
6. Conclusion
Conclusion

*Structured Spinners* paper brings:

- a general structured paradigm for large scale machine learning computations with random matrices, providing computational speedups and storage compression with various applications:
  - kernel approximations via random feature maps
  - dimensionality reduction algorithms
  - deep learning
  - convex optimization via Newton sketches
  - quantization with random projection trees

- theoretical guarantees on the effectiveness of the structured approach.

Open question

Can one obtain computation speedups for these matrices from the *Structured Spinners* model for which the Fast Fourier Transform trick does not work?
Thank you for your attention!

Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform.
In Proceedings of the 38th STOC, pages 557–563. ACM.

Practical and optimal LSH for angular distance.
In NIPS, pages 1225–1233.

Choromanski, K. and Sindhwani, V. (2016).
Recycling randomness with structure for sublinear time kernel expansions.

Predicting parameters in deep learning.
In NIPS.

Random feature mapping with signed circulant matrix projection.
The Johnson-Lindenstrauss lemma and the sphericity of some graphs.

Extensions of Lipschitz mappings into a Hilbert space.

A sparser Johnson-Lindenstrauss transform.
\textit{CoRR}, abs/1012.1577.

Fastfood-computing hilbert space expansions in loglinear time.
In \textit{Proceedings of the 30th ICML}, pages 244–252.

ACDC: A structured efficient linear layer.

Newton sketch: A linear-time optimization algorithm with linear-quadratic convergence.
\textit{CoRR}, abs/1505.02250.

Random features for large-scale kernel machines.
In \textit{NIPS}, pages 1177–1184.
Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning.

Low-rank matrix factorization for deep neural network training with high-dimensional output targets.
In ICASSP.

Spherical LSH for approximate nearest neighbor search on unit hypersphere.
In WADS, pages 27–38.

Restructuring of deep neural network acoustic models with singular value decomposition.

Deep fried convnets.
In 2015 IEEE International Conference on Computer Vision (ICCV), pages 1476–1483.

Circulant binary embedding.
Kronecker matrices

Gaussian or discrete Kronecker matrix

\[ K = R_1 \otimes R_2 \otimes \ldots \otimes R_m \in \mathbb{R}^{2^m \times 2^m} \]

\[ R_i \in \mathbb{R}^{2 \times 2} \text{ or } R_i \in \{-1, 1\}^{2 \times 2} \]

\[ R_i R_i^T = R_i^T R_i = I_2 \]

Kronecker product [Zhang et al., 2015]

For \( A \in \mathbb{R}^{k_1 \times d_1} \), \( B \in \mathbb{R}^{k_1 \times d_1} \),

\[ A \otimes B = \begin{pmatrix}
A_{11} B & \ldots & A_{1d_1} B \\
A_{21} B & \ldots & A_{2d_1} B \\
\vdots & \ddots & \vdots \\
A_{k_11} B & \ldots & A_{k_1 d_1} B 
\end{pmatrix} \in \mathbb{R}^{k_1 k_2 \times d_1 d_2} \]
Thank you for your attention!