TripleSpin strikes back

New framework for fast structured ML computations

Google Research Seminar, New York

Krzysztof Choromanski [1] François Fagan [2] Cédric Gouy-Pailler [3]
Anne Morvan [3,4] Tamás Sarlós [1] Jamal Atif [4]

[1]Google Research

[2]Columbia University

[3]CEA, LIST, LADIS

[4]Paris-Dauphine University, LAMSADE

Plan

Plan

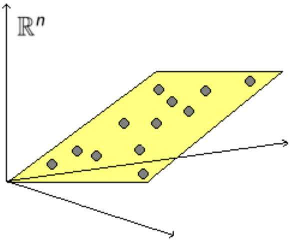
Why random projections? (1/3)

When all you data do not fit into memory:

- Massive data ...
- ... in high dimensionality.

Observation

Lot of high dimensional data with low intrinsic dimension.



Perform dimensionality reduction, e.g.:

- Principal Component Analysis (PCA);
- Random Projection (RP).

Why random projections? (2/3)

Founder Lemma: [Johnson and Lindenstrauss, 1984]:

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

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

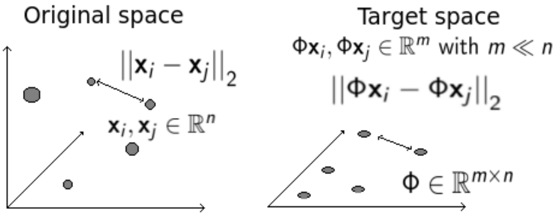
Then there exists a linear map $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ s.t. :

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

- One can take $\Phi =$ **Random** (near orthonormal) which works with high probability.

Why random projections? (3/3)

Dimensionality reduction



$$||\Phi\mathbf{x}_i - \Phi\mathbf{x}_j||_2 \approx ||\mathbf{x}_i - \mathbf{x}_j||_2, \; ||\Phi\mathbf{x}_i||_2 \approx ||\mathbf{x}_i||_2$$

Properties

- Near isometric embedding,
- $(1 \pm \epsilon)$ distorsion,
- Distance and angle preserved between points.

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.

$\Longrightarrow$ information retrieval, similarity search, classification, clustering.

Brief random projections evolution (1/2)

Φ: Dense i.i.d. distribution

- [Johnson and Lindenstrauss, 1984],
- [Frankl and Maehara, 1987]: $\Phi_{i,j} \sim \mathcal{N}(0, \frac{1}{\sqrt{m}})$,
- [Indyk and Motwani, 1998] & [Dasgupta and Gupta, 1999]: simplification of JL lemma's proof,
- [Achlioptas, 2003]: $\Phi_{i,j} \sim \{-1, 1\}$ uniformly,
- [Matoušek, 2008]: $\Phi_{i,j} \sim$ any subgaussian distribution.

Can one sparsify the projection matrix Φ?

Can one sparsify the projection matrix Φ?

Brief random projections evolution (2/2)

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]: $\Phi = \mathbf{PHD}$
  - $\mathbf{P}_{i,j} = \begin{cases} \sim \mathcal{N}(0, \frac{1}{q}) & \text{with probability} \quad q \\ 0 & \text{with probability} \quad 1-q \end{cases}$,
  - $\mathbf{H}$ normalized Hadamard,
  - $\mathbf{D}$ with independent Rademacher (± 1) entries.
- [Matoušek, 2008]: For some $q \in O(\eta^2 m) \leq 1$:

$$\mathbf{P}_{i,j} = \begin{cases} \frac{1}{\sqrt{q}} & \text{with probability} \quad \frac{q}{2} \\ 0 & \text{with probability} \quad 1-q \\ \frac{-1}{\sqrt{q}} & \text{with probability} \quad \frac{q}{2} \end{cases}$$

for $\mathbf{x}$ s.t. $||\mathbf{x}||_\infty / ||\mathbf{x}||_2 \leq \eta$ (i.e. not sparse).

And what is about our *TripleSpin*-family?

Main purpose of *TripleSpin*-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): $O(n \log n)$ instead of $O(mn)$ for matrix-vector product.
- Less storage:
  - $\mathbf{H}$ is not stored,
  - Sparse matrices: diagonal ones,
  - Structured matrices: $n \times n$-circulant one $\implies$ only n parameters,
  - Structured matrices with ± 1 entries: only bits.

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), $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$.

Quantization, e.g.:

- [Yu et al., 2014]: $\mathbf{G}_{circulant}$,
- [Choromanska et al., 2016]: Ψ-regular random matrix.

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", $\frac{1}{\sqrt{n}}\mathbf{SHGPHB}$,
- [Feng et al., 2015]: $\pm 1\mathbf{G}_{circulant}$,
- [Choromanski and Sindhwani, 2016]: "$\mathcal{P}$-model", and Toeplitz-like semi Gaussian matrices,

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

Plan

1. Introduction
2. Brief review of TripleSpin family
3. Some applications
4. Conclusion

Definition of *TripleSpin* family

TripleSpin for 3 blocks

$$\mathbf{G} \rightarrow \mathbf{G}_{struct}$$

$$\mathbf{G}_{struct} = \mathbf{M}_3\mathbf{M}_2\mathbf{M}_1 \in \mathbb{R}^{n\times n},$$

where matrices $\mathbf{M}_1, \mathbf{M}_2$ and $\mathbf{M}_3$ satisfy 3 conditions.

Examples

- $[\mathbf{G}_{circ} \mid \mathbf{G}_{skew-circ} \mid \mathbf{G}_{Toeplitz} \mid \mathbf{G}_{Hankel}]\mathbf{D}_2\mathbf{H}\mathbf{D}_1$,
- $\sqrt{n}\ \mathbf{H}\mathbf{D}_{g_1,\dots,g_n}\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$,
- $\sqrt{n}\ \mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$.

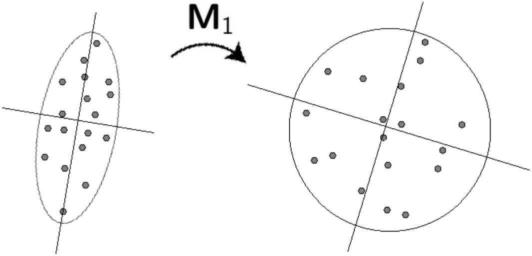
Role of each *TripleSpin* block

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

Role of each *TripleSpin* block - $\mathbf{M}_1$

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

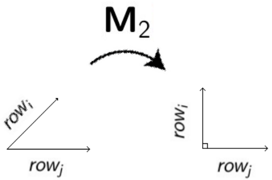
$\hookrightarrow$ Balances data.



Role of each *TripleSpin* block - $\mathbf{M}_2$

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

$\hookrightarrow$ Makes the rows of the final matrix almost independent.



Role of each *TripleSpin* block - $\mathbf{M}_3$

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

$\hookrightarrow$ Budget of randomness.

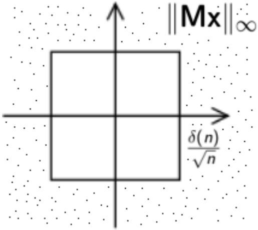
Condition 1

Condition 1: $\mathbf{M}_1$ and $\mathbf{M}_2\mathbf{M}_1$ are $(\delta(n), p(n))$-balanced isometries.

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

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

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



Example

$\mathbf{M}_1 = \mathbf{HD}_1$, since $\mathbf{HD}_1$ is $(\log(n), 2ne^{-\frac{\log^2(n)}{8}})$-balanced.

Condition 2 (1/2)

Condition 2: $\mathbf{M}_2 = \mathbf{V}(\mathbf{W}^1, ..., \mathbf{W}^n)\mathbf{D}_{\rho_1,...,\rho_n}$ for some (Λ_F, Λ_2)-smooth set $\mathbf{W}^1, ..., \mathbf{W}^n \in \mathbb{R}^{k \times n}$ and some i.i.d sub-Gaussian random variables $\rho_1, ..., \rho_n$ with sub-Gaussian norm K.

$$\mathbf{V}(\mathbf{W}^1, ..., \mathbf{W}^n) = \begin{pmatrix} \mathbf{W}^1 \\ \mathbf{W}^2 \\ ... \\ \mathbf{W}^n \end{pmatrix} \qquad \mathbf{D}_{\rho_1,...,\rho_n} = \begin{pmatrix} \rho_1 & 0 & \dots & 0 \\ 0 & \rho_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \rho_n \end{pmatrix}$$

Typically, $K = 1$.

Condition 2 (2/2)

Definition: (Λ_F, Λ_2)-smooth sets

A deterministic set of matrices $\mathbf{W} = \{\mathbf{W}^1, ..., \mathbf{W}^n\}$, where $\mathbf{W}^i = \{w^i_{k,l}\}_{k,l \in \{1,...,n\}}$ is (Λ_F, Λ_2)-smooth if:

- for $i = 1, ..., n$:

$$\mathbf{W}^i = \begin{pmatrix} \vdots & & \vdots & & \vdots \\ \mathbf{W}^i_1 & \cdots & \mathbf{W}^i_l & \cdots & \mathbf{W}^i_n \\ \vdots & & \vdots & & \vdots \end{pmatrix}$$

$$\|\mathbf{W}^i_1\|_2 = .. = \|\mathbf{W}^i_l\|_2 = .. = \|\mathbf{W}^i_n\|_2$$

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

$$\mathbf{W}^i = \begin{pmatrix} \cdots & \vdots & \cdots \\ \cdots & \mathbf{W}^i_l & \cdots \\ \cdots & \vdots & \cdots \end{pmatrix} \quad \mathbf{W}^j = \begin{pmatrix} \cdots & \vdots & \cdots \\ \cdots & \mathbf{W}^j_l & \cdots \\ \cdots & \vdots & \cdots \end{pmatrix}$$

$$(\mathbf{W}^i_l)^T \cdot \mathbf{W}^j_l = 0$$

- $\max_{i,j} \|(\mathbf{W}^j)^T \mathbf{W}^i\|_F \leq \Lambda_F$ and $\max_{i,j} \|(\mathbf{W}^j)^T \mathbf{W}^i\|_2 \leq \Lambda_2$.

Condition 3

Condition 3: $\mathbf{M}_3 = \mathbf{C}(\mathbf{r}, n)$ for $\mathbf{r} \in \mathbb{R}^k$, where $\mathbf{r}$ is random Rademacher (± 1 entries) or Gaussian.

$$M_3 = \begin{pmatrix} \mathbf{r}_1 & \dots & \mathbf{r}_k & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & 0 & \mathbf{r}_1 & \dots & \mathbf{r}_k & 0 & \dots & \dots & 0 \\ & & & & \vdots & \vdots & & & & \\ & & & & \vdots & \vdots & & & & \\ 0 & \dots & \dots & \dots & \dots & \dots & 0 & \mathbf{r}_1 & \dots & \mathbf{r}_k \end{pmatrix}$$

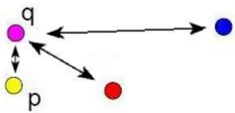
Plan

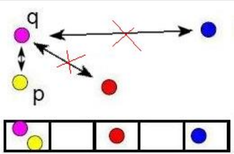
Plan

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.





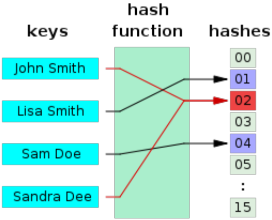
LSH : Two phases

- Build a data structure (hash table) for fast lookup.
- NN search phase: query the database with query point q.

Hashing vs LSH

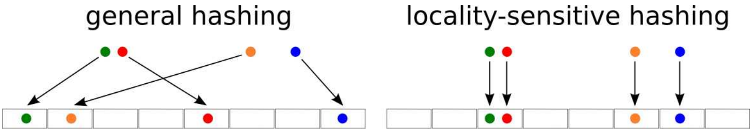
Hashing principle

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



LSH principle

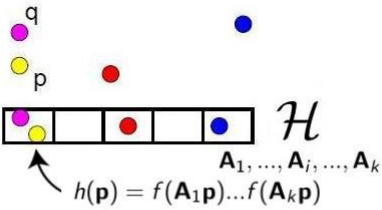
- Exploiting collision probabilities.



LSH in details

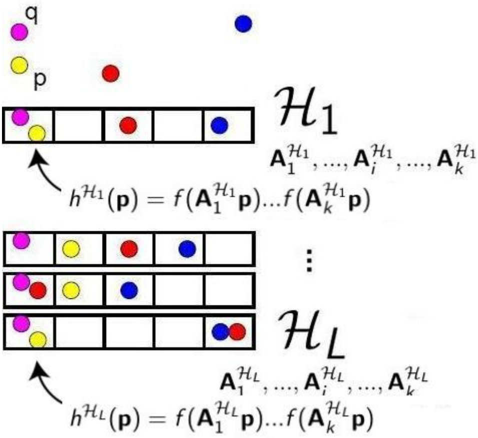
Hash value computation

- Hash value h of a point $\mathbf{x} \in \mathbb{R}^n$ is a combination of k hash function results $h_i, i = 1...k$ s.t. $h_i = f(\mathbf{A}_i \mathbf{x})$ with $\mathbf{A}_i \in \mathbb{R}^{m \times n}$ a projection matrix s.t. $m \ll n$.
- Example: Concatenation: $h = h_1 h_2 ... h_k$.



LSH in details

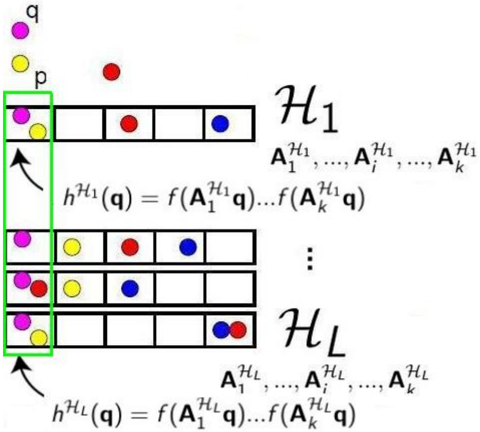
L hash tables



ANN search with LSH

ANN search

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

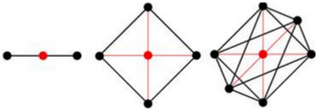


Cross-polytope LSH

Cross-polytope from [Terasawa and Tanaka, 2007]

$$h_i(\mathbf{x}) = f\left(\frac{\mathbf{Gx}}{||\mathbf{Gx}||_2}\right)$$

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



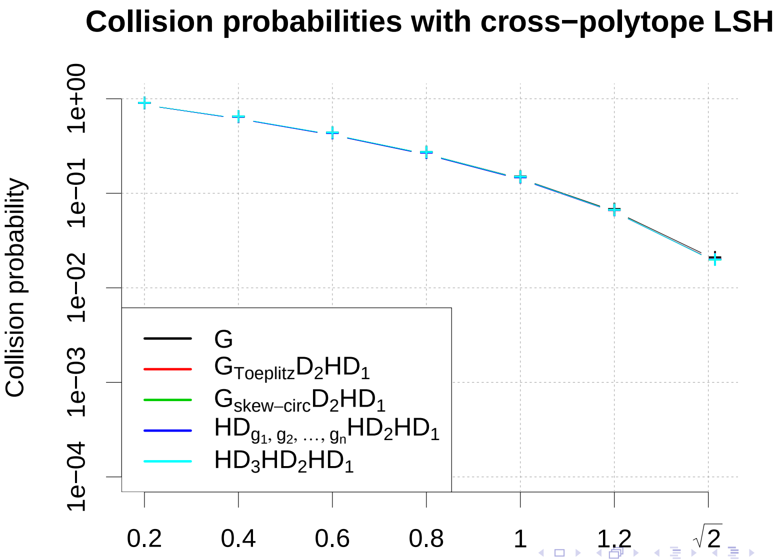
- State-of-the-art cross-polytope LSH [Andoni et al., 2015] $\mathbf{G} \rightarrow \mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$.
- Our variant: $\mathbf{G}_{struct} = \mathbf{M}_3\mathbf{M}_2\mathbf{M}_1$ + theoretical guarantees.

Cross-polytope LSH experiment with *TripleSpin*-matrices

Experimental protocol

- Plot $Pr[h(p) = h(q)]$ as a function of $dist(p, q)$,
- 100 runs,
- $k = 1$,
- Draw points from the hypersphere $\implies \max_{p,q} dist(p, q) = \sqrt{2}$,
- 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 *TripleSpin*-matrices



Plan

Kernel methods

Principle

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

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

- <u>Example:</u> the Gaussian radial basis function or Gaussian kernel,

$$\kappa(\mathbf{x}, \mathbf{y}) = e^{\frac{-||\mathbf{x}-\mathbf{y}||_2^2}{2\sigma^2}}.$$

Why kernel approximation?

Decision evaluation in kernel machines: the "kernel trick"

$$f(x) = \langle \mathbf{w},\ \phi(\mathbf{x})\rangle = \left\langle \sum_{i=1}^{N'} \alpha_i\ \phi(\mathbf{x}_i),\ \phi(\mathbf{x})\right\rangle = \sum_{i=1}^{N'} \alpha_i\ \kappa(\mathbf{x}_i, \mathbf{x})$$

N' : number of nonzero α_i = number of "support vectors"

Why approximation ?

- Problem: evaluating f cost inscreases as the dataset grows
  N number of training samples.
- Kernel or Gram matrix K:

$$K_{ij} = \kappa(x_i, x_j)$$

$\Longrightarrow$ storage cost: $O(N^2)$.

Kernel approximation via random feature maps

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

- $$\left\langle \underbrace{z(\mathbf{x})}_{\in \mathbb{R}^k}, z(\mathbf{y}) \right\rangle \approx \left\langle \underbrace{\phi(\mathbf{x})}_{\in \mathbb{R}^D}, \phi(\mathbf{y}) \right\rangle = \kappa(\underbrace{\mathbf{x}}_{\in \mathbb{R}^n}, \mathbf{y})$$

  where $k \gg n$; D high, possibly infinite.
- $z(\mathbf{x}) = \frac{1}{\sqrt{k}} s(\mathbf{Gx})$,
- random Gaussian matrix $\mathbf{G} \in \mathbb{R}^{k \times n}$ with $k \gg n$, $k = O(n\epsilon^{-2} \log \frac{1}{\epsilon^2})$,
- s is a nonlinearity function.

Still a problem...

- Storage of $\mathbf{G}$: $O(kn)$,
- Computation of $\mathbf{Gx}$: $O(kn)$.

Solution

- Storage of $\mathbf{G}_{struct}$: $O(k \log n)$,
- Computation of $\mathbf{G}_{struct}\mathbf{x}$: $O(k \log n)$.

Experimental protocol for kernel approximation (1/2)

$\mathbf{A} \in \mathbb{R}^{k \times n}$ with $k \gg n$,

Gaussian kernel

- $\kappa_G(\mathbf{x}, \mathbf{y}) = e^{\frac{-||\mathbf{x}-\mathbf{y}||_2^2}{2\sigma^2}}$,
- $\tilde{\kappa}_G(\mathbf{x}, \mathbf{y}) = \frac{1}{k} s(\mathbf{Ax})^T s(\mathbf{Ay})$ with $s(x) = e^{\frac{-ix}{\sigma}}$ applied pointwise.

Angular kernel

- $\kappa_0(\mathbf{x}, \mathbf{y}) = 1 - \frac{\theta}{\pi}$ with $\theta = \cos^{-1}(\frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||||\mathbf{y}||})$,
- $\tilde{\kappa}_0(\mathbf{x}, \mathbf{y}) = 1 - \frac{dist_{Hamming}(s(\mathbf{Ax}), s(\mathbf{Ay}))}{k}$

  with $s(x) = sign(x)$ applied pointwise.

Experiments for kernel approximation (1/4)

Speedups with Gaussian kernel

$Time(\mathbf{G}) / Time(\mathbf{G}_{struct})$

Matrix dimensions	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
$\mathbf{G}_{Toeplitz}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x1.4	x3.4	x6.4	x12.9	x28.0	x42.3	x89.6
$\mathbf{G}_{skew-circ}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x1.5	x3.6	x6.8	x14.9	x31.2	x49.7	x96.5
$\mathbf{H}\mathbf{D}_{g_1,\dots,g_n}\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x2.3	x6.0	x13.8	x31.5	x75.7	x137.0	x308.8
$\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x2.2	x6.0	x14.1	x33.3	x74.3	x140.4	x316.8

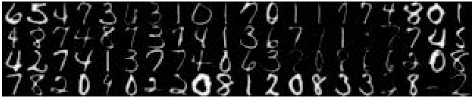
Experiments for kernel approximation (2/4)

Speedups with Gaussian kernel

$Time(\mathbf{G}) / Time(\mathbf{G}_{struct})$

$(n = 2^{11})$ k	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
$\mathbf{G}_{Toeplitz}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x5.97	x6.68	x6.51	x6.52	x6.95
$\mathbf{G}_{skew-circ}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x6.61	x6.73	x6.54	x6.65	x7.36
$\mathbf{H}\mathbf{D}_{g_1,\ldots,g_n}\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x13.74	x11.35	x10.86	x10.82	x11.90
$\mathbf{H}\mathbf{D}_3\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_1$	x10.67	x11.39	x10.22	x10.36	x11.8

Experimental protocol for kernel approximation (2/2)



Source [1]

Measure of accuracy

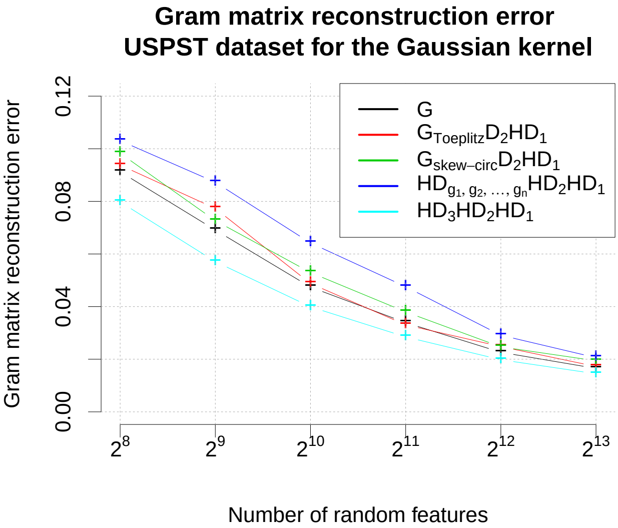
- 10 runs,
- Dataset: USPST,
- 16×16 grayscale images,
- 2007 points of dimensionality 256 ($n = 256$),
- $\sigma = 9.4338$,
- Plots Gram reconstruction error: $\frac{||\mathbf{K}-\tilde{\mathbf{K}}||_F}{||\mathbf{K}||_F}$,
- $\mathbf{K}_{i,j} = \kappa(x_i, x_j)$.

[1] http://statweb.stanford.edu/~tibs/ElemStatLearn/data.html

Experiments for kernel approximation (3/4)

$$\frac{||\mathbf{K}-\tilde{\mathbf{K}}||_F}{||\mathbf{K}||_F}$$

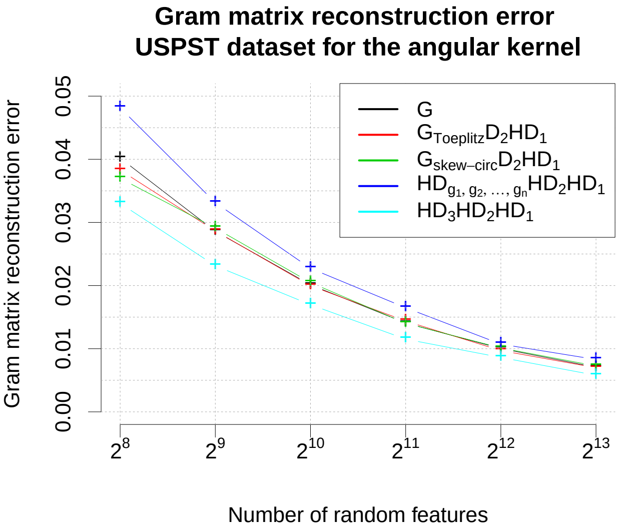
$$\mathbf{K}_{i,j} = \kappa(x_i, x_j)$$



Experiments for kernel approximation (4/4)

$$\frac{||\mathbf{K}-\tilde{\mathbf{K}}||_F}{||\mathbf{K}||_F}$$

$$\mathbf{K}_{i,j} = \kappa(x_i, x_j)$$



Plan

Brief review of unconstrained convex optimization (1/5)

The unconstrained optimization problem

$$\text{minimize} \;\; f(x)$$

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

Descent methods

- $x^{(t+1)} = x^{(t)} + \mu^{(t)} \Delta x^{(t)}$,
- $f(x^{(t+1)}) < f(x^{(t)})$,
- $\mu^{(t)} > 0$ except when $x^{(t)}$ is optimal,
- $\Delta x^{(t)}$ is the *step* or *search direction*,
- $\mu^{(t)}$ is called the *step size* or *step length*.

Brief review of unconstrained convex optimization (2/5)

General descent method

given a starting point x

repeat

  Determine a descent direction Δx

  Backtracking line search. Choose a step size $\mu > 0$

  Update. $x := x + \mu\Delta x$

until stopping criterion is satisfied;

Brief review of unconstrained convex optimization (3/5)

Gradient descent method with Newton step

Newton step: $\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$ (vs. $\Delta x = -\nabla f(x)$)

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:

$$\lambda^2 = -\nabla f(x)^T \Delta x$$

Brief review of unconstrained convex optimization (4/5)

Newton's method

given a starting point x, tolerance $\epsilon > 0$

repeat

Compute the Newton step Δx and λ^2.

Stopping criterion. **quit** if $\lambda^2 \leq \epsilon$

Backtracking line search. Choose a step size $\mu > 0$

Update. $x := x + \mu \Delta x$

until stopping criterion is satisfied;

Brief review of unconstrained convex optimization (5/5)

Backtracking line search

given a descent direction Δx, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$

$\mu := 1$

while $\underline{f(x + \mu \Delta x) > f(x) + \alpha \mu \nabla f(x)^T \Delta x}$ **do**

$\quad \mu := \beta \mu$

end

Principle of Newton sketch's algorithm [Pilanci and Wainwright, 2015]

Newton's method of unconstrained convex optimization

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

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

Is of interest where we have an analytic expression for the square root of the Hessian matrix. The problem is cast as the following:

$$x^{(t+1)} = x^{(t)} - \mu \, (\underbrace{(S^{(t)} (\nabla^2 f(x^{(t)}))^{1/2})^T}_{(SM)^T} \underbrace{S^{(t)} (\nabla^2 f(x^{(t)}))^{1/2}}_{SM})^{-1} \nabla f(x^{(t)})$$

where $S^{(t)} \in \mathbb{R}^{m \times n}$ is a sequence of isotropic sketchs matrices.

Example for Newton sketch's algorithm (1/2)

Large scale logistic regression problem

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

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

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

$$\text{s.t. } a_i \in \mathbb{R}^n,$$

$$y_i \in \{-1, 1\}.$$

Example for Newton sketch's algorithm (2/2)

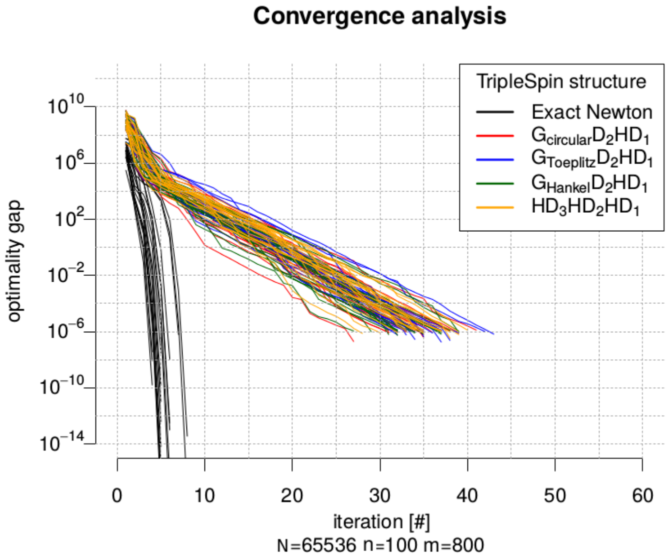
Analytic expressions for the gradient and the Hessian matrix

- $\nabla f(x^{(t)}) = \sum_{i=1}^{n} \left(\frac{1}{1+\exp(-y_i a_i^T x)} - 1\right) y_i a_i \ \in \mathbb{R}^n$,
- $\nabla^2 f(x^{(t)}) = A^T diag\left(\frac{1}{1+\exp(-a_i^T x)}\left(1 - \frac{1}{1+\exp(-a_i^T x)}\right)\right) A \ \in \mathbb{R}^{n \times n}$,

  $A = [a_1^T ... a_N^T] \in \mathbb{R}^{N \times n}$, with $N \gg n$,
- We set

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

Experimental results (1/2)



Newton sketch's algorithm, complexity analysis (1/3)

Comparison

- Exact Newton:

$$\nabla^2 f(x)^{-1}$$

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

$$Cost = O(Nn^2 + n^3) \ (n \ll N)$$

Newton sketch's algorithm, complexity analysis (2/3)

Comparison

- Exact Newton:

$$Cost = O(Nn^2 + n^3)\ (n \ll N)$$

- Sketching: $\Big(\underbrace{(S^{(t)}\ (\nabla^2 f(x^{(t)}))^{1/2})^T}_{(SM)^T}\ \underbrace{S^{(t)}(\nabla^2 f(x^{(t)}))^{1/2}}_{SM}\ \Big)^{-1}$

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

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

Critical issue: when is $O(3nN \log N + mn^2)$ better than $O(Nn^2)$?

Newton sketch's algorithm, complexity analysis (3/3)

Comparison

- Exact Newton:

$$Cost = O(Nn^2 + n^3) \ (n \ll N)$$

- Sketching: $Cost = O(3nN \log N + mn^2 + n^3)$ with $m \ll N$

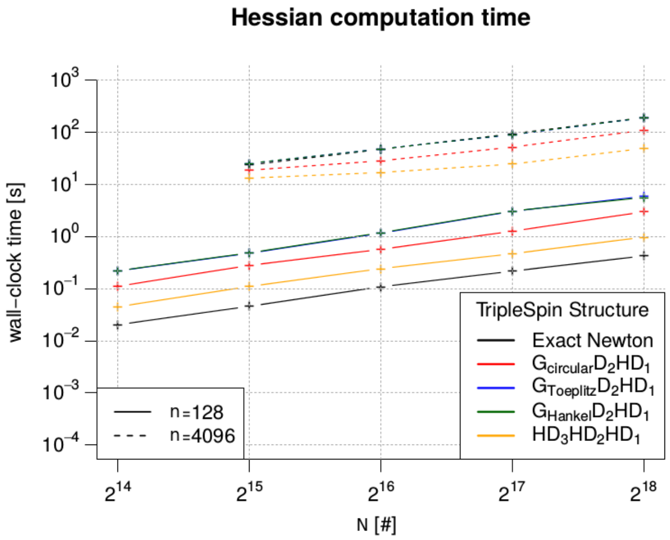
- Sub-sampling (m rows):

$$(\underbrace{SampleRows(\ (\nabla^2 f(x^{(t)}))^{1/2})^T}_{(M)^T}\ \underbrace{SampleRows((\nabla^2 f(x^{(t)}))^{1/2}}_{M}\ ))^{-1}$$

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

$$Cost = O(mn^2 + n^3) \text{ with } m \ll N$$

Experimental results (2/2)



Plan

Conclusion

TripleSpin paper brings:

- first theoretical guarantees for the fastest known cross-polytope LSH [Andoni et al., 2015] based on the $\mathbf{HD}_3\mathbf{HD}_2\mathbf{HD}_1$ structured matrix,
- a general structured paradigm for large scale machine learning computations with random matrices, providing computational speedups and storage compression.

Questions

- Can one obtain computations speedups for these matrices from the *TripleSpin* model for which the Fast Fourier Transform trick does not work ?
- Theoretical guarantees for learning with structured matrices ? (work in progress)

Thank you for your attention!

References I

Achlioptas, D. (2003).
Database-friendly random projections: Johnson-lindenstrauss with binary coins.
J. Comput. Syst. Sci., 66(4):671–687.

Ailon, N. and Chazelle, B. (2006).
Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform.
In Proceedings of the 38th STOC, pages 557–563. ACM.

Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I. P., and Schmidt, L. (2015).
Practical and optimal LSH for angular distance.
In NIPS, pages 1225–1233.

Choromanska, A., Choromanski, K., Bojarski, M., Jebara, T., Kumar, S., and LeCun, Y. (2016).
Binary embeddings with structured hashed projections.
To appear in ICML.

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

Dasgupta, S. and Gupta, A. (1999).
An elementary proof of the johnson-lindenstrauss lemma.
Technical report, UC Berkeley.

Feng, C., Hu, Q., and Liao, S. (2015).
Random feature mapping with signed circulant matrix projection.
In Proceedings of the 24th IJCAI, pages 3490–3496.

References II

Frankl, P. and Maehara, H. (1987).
The johnson-lindenstrauss lemma and the sphericity of some graphs.
J. Comb. Theory Ser. A, 44(3):355–362.

Indyk, P. and Motwani, R. (1998).
Approximate nearest neighbors: Towards removing the curse of dimensionality.
In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC '98, pages 604–613, New York, NY, USA. ACM.

Johnson, W. and Lindenstrauss, J. (1984).
Extensions of Lipschitz mappings into a Hilbert space.
In Conference in modern analysis and probability (New Haven, Conn., 1982), volume 26 of Contemporary Mathematics, pages 189–206. American Mathematical Society.

Kane, D. M. and Nelson, J. (2010).
A sparser johnson-lindenstrauss transform.
CoRR, abs/1012.1577.

Le, Q., Sarlós, T., and Smola, A. (2013).
Fastfood-computing hilbert space expansions in loglinear time.
In Proceedings of the 30th ICML, pages 244–252.

Matoušek, J. (2008).
On variants of the johnson–lindenstrauss lemma.
Random Struct. Algorithms, 33(2):142–156.

References III

Pilanci, M. and Wainwright, M. J. (2015).
Newton sketch: A linear-time optimization algorithm with linear-quadratic convergence.
CoRR, abs/1505.02250.

Rahimi, A. and Recht, B. (2007).
Random features for large-scale kernel machines.
In NIPS, pages 1177–1184.

Rahimi, A. and Recht, B. (2009).
Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning.
In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, Advances in Neural Information Processing Systems 21, pages 1313–1320. Curran Associates, Inc.

Terasawa, K. and Tanaka, Y. (2007).
Spherical LSH for approximate nearest neighbor search on unit hypersphere.
In WADS, pages 27–38.

Yu, F. X., Kumar, S., Gong, Y., and Chang, S. (2014).
Circulant binary embedding.
In Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014, pages 946–954.

Hadamard transform - recursive definition

$$\mathbf{H}_0 = 1$$

$$\mathbf{H}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\mathbf{H}_m = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{H}_{m-1} & \mathbf{H}_{m-1} \\ \mathbf{H}_{m-1} & -\mathbf{H}_{m-1} \end{pmatrix}$$

Thank you for your attention!