Linear and Nonlinear Low-Rank Approximations

in theory and practice

Alex Gittens
International Computer Science Institute (ICSI)
Department of Statistics and AMPLab
University of California, Berkeley

RPI CS Colloquium
May 5, 2016

OUTLINE

Low-rank matrix approximation is a fundamental tool, used for multiple reasons: noise elimination, computational tractability, statistical modeling, ...

We consider the following instances of low-rank approximation:

- Randomized SPSD Sketching for fast linear low-rank approximation
- Polynomial Random Features for fast approximate polynomial kernel learning
- Word2Vec for semantically meaningful word embeddings
- The performance of Low-rank approximation algorithms in Spark

SPSD SKETCHES

Our basic task

$\mathbf{A} \in \mathbb{R}^{n \times n}$ is a *huge* symmetric positive-semidefinite matrix. Given $k \ll n$, we would like a low-rank approximation to $\mathbf{A}$ with rank about k.

1. This abstract problem is ubiquitous in data processing tasks: statistical analysis, spectral clustering, kernel methods, optimization, ...

2. Traditional deterministic approaches (via truncated EVD, Krylov methods, etc.) cost at least $O(n^2 k \log n)$ operations, and can have high communications costs.

One can consider using randomness to assist in the design of time and communication efficient algorithms for finding low-rank approximations of large matrices.

We give new error bounds for "sketching" schemes for symmetric positive semidefinite matrices.
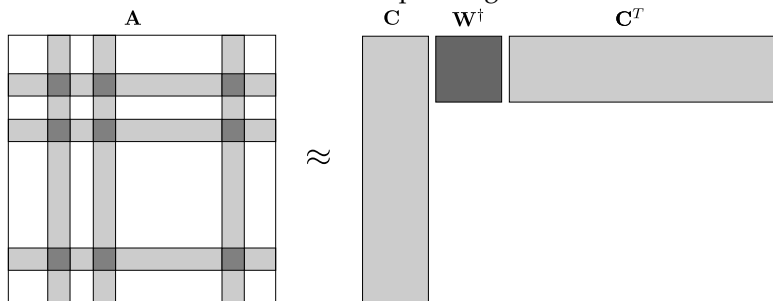
Our objective

Determine how the errors of SPSD sketches in the spectral, Frobenius, and trace norms compare with the errors of $\mathbf{A}_k$, the best rank-k approximation to $\mathbf{A}$.

NYSTRÖM EXTENSIONS

The simplest SPSD sketch is the Nyström extension:

$$\mathbf{A} = \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^{T},$$

where $\mathbf{C}$ consists of uniformly randomly chosen columns of $\mathbf{A}$ and $\mathbf{W}$ consists of $\mathbf{C}$ restricted to the corresponding rows.

Nyström extensions perform well when the information in its top k-dimensional eigenspace is spread throughout **A** :

$$\mathbf{A} = 20 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

versus

$$\mathbf{A} = 20 \begin{bmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{bmatrix} \begin{bmatrix} 1/2 & -1/2 & -1/2 & 1/2 \end{bmatrix} = \begin{bmatrix} 5 & -5 & -5 & -5 \\ -5 & 5 & 5 & -5 \\ -5 & 5 & 5 & -5 \\ 5 & -5 & -5 & 5 \end{bmatrix}$$

key point: we need the support of the top k eigenvectors to be spread out.
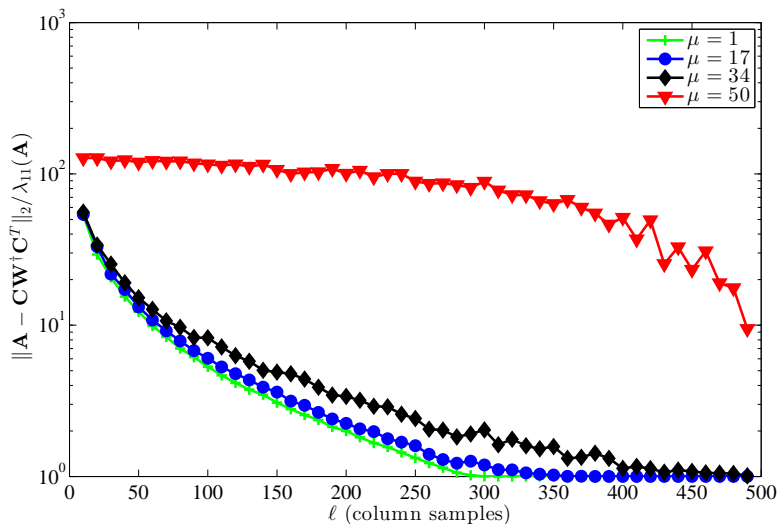
Let $\mathbf{U}_1$ be an orthonormal basis for the dominant k-dimensional eigenspace of $\mathbf{A}$.

A measure of the "spreadness" of the eigenvectors in $\mathbf{U}_1$ is given by the *coherence* of $\mathbf{U}_1$:

$$\mu := \frac{n}{k} \max_j \|(\mathbf{U}_1)_j\|_2^2.$$

- μ is between 1 (best case) and n/k (worst case)
- μ both theoretically *and empirically* determines the feasibility of Nyström extensions...

$\mathbf{A} \in \mathbb{R}^{500 \times 500}$ is full-rank, but numerically rank 20. The target rank $k = 10$. Each point is the average of 60 trials.

The best previous error bound on the error of Nyström extensions in terms of the coherence is

(Talwalkar and Rostamizadeh 2010)

Let $\mathbf{A}$ be exactly rank-k. If $\ell = \Omega(\mu k \log(k/\delta))$, then

$$\|\mathbf{A} - \mathbf{CW}^\dagger \mathbf{C}^T\| = 0$$

with probability at least $1 - \delta$.

What about $\mathbf{A}$ that are not exactly low-rank?

OPTIMAL BOUNDS FOR NYSTROM EXTENSIONS

The approximation errors can be bounded when ℓ is proportional to the coherence; our framework gives the following result.

Spectral-norm error bound (G. 2011)

If $\ell \geq 8\mu k \log(k/\delta)$, then

$$\|\mathbf{A} - \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T\|_2 \leq \|\mathbf{A} - \mathbf{A}_k\|_2 \left(1 + \frac{2n}{\ell}\right)$$

with probability at least $1 - \delta$.

A matrix constructed in (Boutsidis et al. 2011) shows that this bound is **tight**: there are matrices for which the relative spectral-norm error is on the order of n/ℓ.

SPSD SKETCHES

SPSD sketches generalize the Nyström extension, by potentially mixing information across the columns of $\mathbf{A}$, thereby removing the sensitivity to μ.

Fix an arbitrary "sketching matrix" $\mathbf{S} \in \mathbb{R}^{n \times \ell}$. The corresponding sketch of the SPSD matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is

$$\hat{\mathbf{A}} = \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^{T},$$

where $\mathbf{C} = \mathbf{AS}$ and $\mathbf{W} = \mathbf{S}^{T}\mathbf{AS}$.

- ▶ The sketch $\hat{\mathbf{A}}$ is also SPSD.
- ▶ This model allows for both column-sampling based approximations (e.g. Nyström extensions) and column-mixture based approximations.
- ▶ To form the sketch requires only one pass over $\mathbf{A}$.

CHOICE OF SKETCHING MATRICES

Dominant arithmetic cost of forming the sketch is the matrix–matrix multiply **AS**.

A natural choice for **S** is a matrix of i.i.d. $\mathcal{N}(0, 1)$ Gaussians, proposed in (Martinsson et al. 2006).

- ► Computation of **AS** takes $O(n^2\ell)$ time for general **A**.
- ► The columns of **A** are well-mixed.

(Woolfe et al. 2008) proposed using *structured* random projections.

- ► Computation of **AS** takes reduced time $O(n^2 \log(\ell))$.
- ► Mixing not as uniform, so potential accuracy loss.

We consider the following specific sketches, corresponding to different distributions on $\mathbf{S}$:

- When $\mathbf{S}$ selects columns uniformly at random without replacement from $\mathbf{A}$, we call $\hat{\mathbf{A}}$ a Nyström extension.
- When $\mathbf{S}$ selects columns from $\mathbf{A}$ randomly with replacement with probabilities proportional to their *leverage scores*, $\hat{\mathbf{A}}$ is a leverage sketch.
- When $\mathbf{S}$ consists of i.i.d. $\mathcal{N}(0,1)$ Gaussians, $\hat{\mathbf{A}}$ is a Gaussian sketch.
- When $\mathbf{S}$ is a subsampled randomized Fourier transform (SRFT), $\hat{\mathbf{A}}$ is an SRFT sketch.

LEVERAGE SCORES

Write the SVD of $\mathbf{A}_k = \mathbf{U}_1 \mathbf{\Sigma}_1 \mathbf{U}_1^T$. The statistical leverage scores of the columns of $\mathbf{A}$ (with respect to rank k), are the scaled column norms of $\mathbf{U}_1^T$:

$$\left\{ \ell_j := \frac{n}{k} \|(\mathbf{U}_1^T)_j\|_2^2, j = 1, \ldots, n \right\}.$$
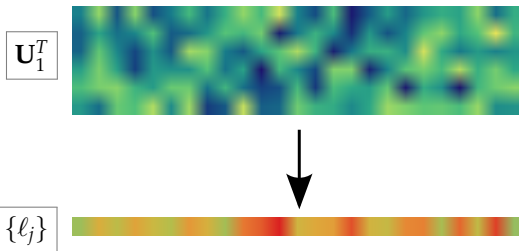
Note that μ, the coherence, is the largest of the leverage scores of the columns of $\mathbf{A}$.

Leverage sketches:

- The idea of leverage score sampling for forming column-sampling based low-rank approximations due to (Drineas et al. 2008).
- Columns are sampled randomly from **A** with probability proportional to their leverage scores.
- Intuitively, leverage score sampling ensures that no important columns are ignored.
- Assuming the leverage scores as given, costs $O(\ell^3 + n\ell^2)$ operations to form.
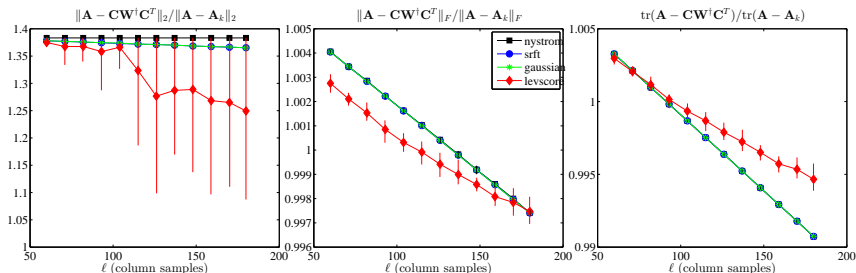- The leverage scores can be approximated.

EMPIRICAL PERFORMANCE (EXACT SCHEMES)
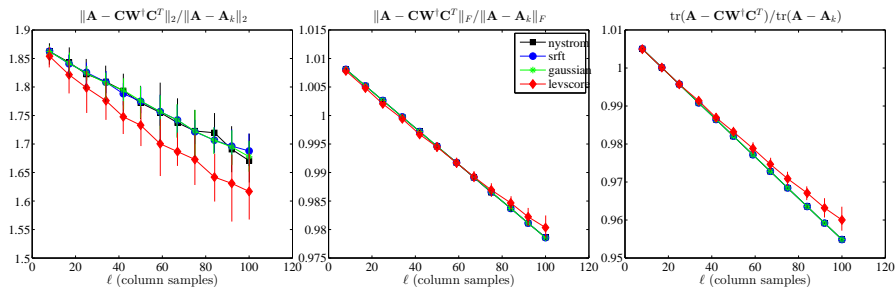
Dexter, a 2000×2000 Gram matrix from the UCI Machine Learning Repository. Target rank $k = 8$.
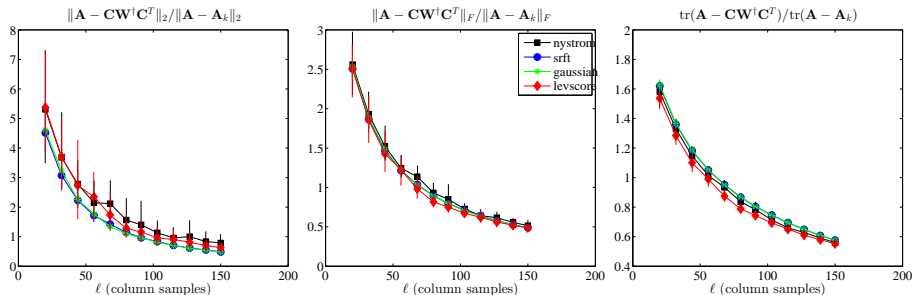
Abalone, a 4898×4898 Radial Basis Kernel matrix from the UCI Machine Learning Repository. Target rank $k = 20$.

Enron, a $10K \times 10K$ Graph Laplacian matrix from the Stanford SNAP collection. Target rank $k = 60$.

PRIOR WORK

Source, sketch	ℓ	$\|\mathbf{A} - \mathbf{CW}^\dagger \mathbf{C}^T\|_2$
(Drineas and Mahoney 2005), column-sampling	$\Omega(\epsilon^{-4}k)$	$\|\mathbf{A} - \mathbf{A}_k\|_2 + \epsilon \sum_{i=1}^n A_{ii}^2$
(Talwalkar and Rostamizadeh 2010), Nyström	$\Omega(\mu k \log k)$	0, if $\text{rank}(\mathbf{A}) = k$
(Kumar et al. 2012), Nyström	$\Omega(1)$	$\|\mathbf{A} - \mathbf{A}_k\|_2 + (n/\sqrt{\ell}) \max_{ii} A_{ii}$
(Chiu and Demanet 2012), Nyström	$\Omega(\mu k \log n)$	$(1 + n/\ell)\|\mathbf{A} - \mathbf{A}_k\|_2$
(Chiu and Demanet 2012), SRFT sketch	$\Omega(k \log^2 n)$	$(1 + n/\ell)\|\mathbf{A} - \mathbf{A}_k\|_2$

▶ The estimated additional error in (Drineas and Mahoney 2005) can be on the order of $\epsilon \, \text{tr}(\mathbf{A})$.

▶ The (Talwalkar and Rostamizadeh 2010) exact recovery result requires $\mathbf{A}$ to be exactly low-rank.

▶ The (Chiu and Demanet 2012) results require $\Omega(k \log n)$ samples as opposed to $\Omega(k \log k)$. The factor n/ℓ is optimal in the Nyström bound, but unnecessary in the SRFT bound.

(G. and Mahoney 2013) provides a framework for deriving significantly improved asymptotic error bounds.

DETERMINISTIC ERROR BOUNDS FOR SPSD SKETCHES

Recall the partitioned eigendecomposition of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T = [\mathbf{U}_1 \ \mathbf{U}_2] \begin{bmatrix} \mathbf{\Sigma}_1 & \\ & \mathbf{\Sigma}_2 \end{bmatrix} [\mathbf{U}_1 \ \mathbf{U}_2]^T$$

and let

$$\mathbf{\Omega}_1 = \mathbf{U}_1^T\mathbf{S} \quad \text{and} \quad \mathbf{\Omega}_2 = \mathbf{U}_2^T\mathbf{S}$$

capture the interactions of the sketching matrix with the dominant and residual eigenspaces of $\mathbf{A}$.

Geometrical interpretation (when $\mathbf{S}$ has orthogonal columns):

- $\mathbf{U}_1^T\mathbf{S}$ has full row-rank $\Leftrightarrow \tan(\mathbf{U}_1, \mathbf{S}) \neq \infty$.
- $\left\| \mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger \right\|_2 = \tan(\mathbf{U}_1, \mathbf{S})$.

Our error bounds for SPSD sketches follow from the key observation that

SPSD sketches approximate $\mathbf{A}^{1/2}$, (G. 2011)

$$\mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T = (\mathbf{A}^{1/2}\mathbf{P}_{\mathbf{A}^{1/2}\mathbf{S}})(\mathbf{P}_{\mathbf{A}^{1/2}\mathbf{S}}\mathbf{A}^{1/2}).$$

Corollaries:

- $\mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T \approx \mathbf{A}_k$ when the range of $\mathbf{A}^{1/2}\mathbf{S}$ is close to the range of the dominant k-dimensional eigenspace of $\mathbf{A}^{1/2}$, spanned by $\mathbf{U}_1$.
- The errors of SPSD sketches are given by:

$$\|\mathbf{A} - \mathbf{C}\mathbf{W}^{\dagger}\mathbf{C}^T\|_{\xi} = \|\mathbf{A} - \mathbf{A}^{1/2}\mathbf{P}_{\mathbf{A}^{1/2}\mathbf{S}}\mathbf{A}^{1/2}\|_{\xi}$$

for $\xi \in \{2, \mathrm{F}, \mathrm{tr}\}$.

By extending the perturbation arguments of (Halko et al. 2011),

Simplified error bounds from (G. and Mahoney 2013)

If $\Omega_1 = \mathbf{U}_1^T \mathbf{S}$ has full row rank, then

$$\|\mathbf{A} - \mathbf{C}\mathbf{W}^\dagger \mathbf{C}^T\|_2 \leq \left(1 + \left\|\Omega_2\Omega_1^\dagger\right\|_2^2\right) \|\mathbf{A} - \mathbf{A}_k\|_2,$$

$$\|\mathbf{A} - \mathbf{C}\mathbf{W}^\dagger \mathbf{C}^T\|_F \leq \|\mathbf{A} - \mathbf{A}_k\|_F + 2\sqrt{2}\left\|\Omega_2\Omega_1^\dagger\right\|_2^2 \cdot \mathrm{tr}(\mathbf{A} - \mathbf{A}_k), \text{ and}$$

$$\mathrm{tr}(\mathbf{A} - \mathbf{C}\mathbf{W}^\dagger \mathbf{C}^T) \leq \left(1 + \left\|\Omega_2\Omega_1^\dagger\right\|_2^2\right) \cdot \mathrm{tr}(\mathbf{A} - \mathbf{A}_k)$$

The randomness of **S** enters only through the sketching interaction matrix $\Omega_2\Omega_1^\dagger$, so these bounds can be used for arbitrary sampling schemes. Recall that

$$\left\|\Omega_2\Omega_1^\dagger\right\|_2 = \tan(\mathbf{U}_1, \mathbf{S}).$$

APPLICATION: GAUSSIAN SKETCHES

Gaussian sketches (G. and Mahoney 2013)

If $\ell = \Omega((1 + \epsilon^{-1})k)$, then

$$\|\mathbf{A} - \mathbf{C}\mathbf{W}^\dagger\mathbf{C}^T\|_2 \le (1 + \epsilon)\|\mathbf{A} - \mathbf{A}_k\|_2 + \frac{\epsilon}{k}\operatorname{tr}(\mathbf{A} - \mathbf{A}_k),$$

$$\|\mathbf{A} - \mathbf{C}\mathbf{W}^\dagger\mathbf{C}^T\|_F \le \|\mathbf{A} - \mathbf{A}_k\|_F + \sqrt{\epsilon\|\mathbf{A} - \mathbf{A}_k\|_2\operatorname{tr}(\mathbf{A} - \mathbf{A}_k)}$$
$$+ \sqrt{\frac{\epsilon}{k}}\operatorname{tr}(\mathbf{A} - \mathbf{A}_k), \text{ and}$$

$$\operatorname{tr}(\mathbf{A} - \mathbf{C}\mathbf{W}^\dagger\mathbf{C}^T) \le (1 + \epsilon)\operatorname{tr}(\mathbf{A} - \mathbf{A}_k).$$

with probability at least $1 - k^{-1} - e^{-k\epsilon^{-1}}$.

This and similarly improved bounds for the other SPSD sketches
follow from our deterministic bounds and analyses of the sketching
interaction matrices available for the different choices of $\mathbf{S}$.

POLYNOMIAL RANDOM FEATURE MAPS FOR FAST KERNEL LEARNING

- **Goal**: Learn non-linear classifiers using linear methods and non-linear feature mappings
- **Challenge**: How to deal with an extremely large number of non-linear features?
- **Solution**: The kernel trick

$$\Phi : \mathbb{R}^d \to \mathcal{H}$$

defines a kernel function $K(\mathbf{x}, \mathbf{y})$ by

$$K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}}, \quad \forall \, \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$$

- Using K, a classifier **H** is then learned for some $\mathbf{w} \in \mathcal{H}$, where

$$\mathbf{H} : \mathbf{x} \mapsto \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle = \sum_{i=1}^{n} \alpha_i \mathbf{K}(\mathbf{x}_i, \mathbf{x})$$

THE TROUBLES WITH KERNELS

- **Dual formulation:** Allows using kernel instead of feature maps, but requires dealing with an $n \times n$ matrix; expensive and possibly can't hold in memory
- **Curse of support**: the number of data points used in **w** can grow unboundedly with the training data size $\Rightarrow$ applying classifier **H** is expensive
- **Solution**: Randomized Features Maps (RFM)

$$\mathbf{Z} : \mathbb{R}^d \to \mathbb{R}^D$$

such that $\forall\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$

$$\langle \mathbf{Z}(\mathbf{x}), \mathbf{Z}(\mathbf{y}) \rangle \approx \mathrm{K}(\mathbf{x}, \mathbf{y})$$

Use **Z** as a feature map in primal formulations

- RFMs can be used to approximate several classes of kernels

POLYNOMIAL KRR

We focus on RFMs for the polynomial kernel:

$$K(\mathbf{x}, \mathbf{y}) := (\langle \mathbf{x}, \mathbf{y} \rangle + q)^r = \langle [\sqrt{q}, \mathbf{x}], [\sqrt{q}, \mathbf{y}] \rangle^r$$

where $q \in \mathbb{R}^+$ and $r \in \mathbb{N}_0$

- The explicit feature map

$$x \mapsto (1, x_1^r, x_2^r, \dots, x_1^2 x_2 \cdots x_{r-1}, \dots)$$

is *extremely* high-dimensional

- RFM approaches ((Kar–Karnick 2012) and TensorSketch (Pagh–Pham 2013)) waste capacity: D is larger than desirable, theoretically and empirically.

- Our contributions:
    1. A much refined analysis of the Kar–Karnick RFM
    2. A scheme to generate more efficient random features

KAR–KARNICK RFMS

Let $\mathbf{w}$ be a vector of random signs. Then

$$\mathbb{E}\langle \mathbf{w}, \mathbf{x}\rangle\langle \mathbf{w}, \mathbf{y}\rangle = \mathbf{x}^{\mathsf{T}}\mathbb{E}(\mathbf{w}\mathbf{w}^{\mathsf{T}})\mathbf{y} = \langle \mathbf{x}, \mathbf{y}^1\rangle.$$

Likewise,

$$\mathbb{E}\left[\left(\prod_{i=1}^{r}\langle \mathbf{w}_i, \mathbf{x}\rangle\right)\left(\prod_{j=1}^{r}\langle \mathbf{w}_j, \mathbf{y}\rangle\right)\right] = \langle \mathbf{x}, \mathbf{y}\rangle^r.$$
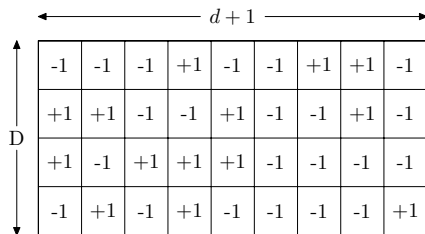
So $z : \mathbf{x} \mapsto \left(\prod_{i=1}^{r}\langle \mathbf{w}_i, [\sqrt{q}, \mathbf{x}]\rangle\right)$ is a RFM for the polynomial kernel:

$$\mathbb{E}\big(z(\mathbf{x}) \cdot z(\mathbf{y})\big) = (\langle \mathbf{x}, \mathbf{y}\rangle + q)^r.$$

A better performing RFM can be obtained using i.i.d. copies of this random map z:

$$\mathbf{Z} : \mathbf{x} \mapsto \frac{1}{\sqrt{D}}(z_1(\mathbf{x}), \ldots, z_D(\mathbf{x})).$$

Another view: let $\mathbf{W}_i \in \mathbb{R}^{D \times (d+1)}$ be a matrix of random signs for $i = 1, \ldots, r$.
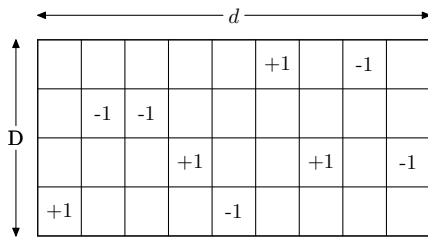

-1	-1	-1	+1	-1	-1	+1	+1	-1
+1	+1	-1	-1	+1	-1	-1	+1	-1
+1	-1	+1	+1	+1	-1	-1	-1	-1
-1	+1	-1	+1	-1	-1	-1	-1	+1

Then $\mathbf{Z} : \mathbf{x} \mapsto \frac{1}{\sqrt{D}} (\mathbf{W}_1[\sqrt{q}, \mathbf{x}]) \odot (\mathbf{W}_2[\sqrt{q}, \mathbf{x}]) \odot \cdots \odot (\mathbf{W}_r[\sqrt{q}, \mathbf{x}])$.

TENSORSKETCH POLYNOMIAL RFMS

Def: A CountSketch matrix $\mathbf{C} : \mathbb{R}^d \to \mathbb{R}^D$ has one nonzero entry per column that is ± 1 with equal probability

(Pagh–Pham 2013) show that if $\mathbf{C}_i$ for $i = 1, \ldots, r$, are independent CountSketch matrices, then

$$\mathbf{Z} : \mathbf{x} \mapsto \frac{1}{\sqrt{D}} \mathbf{F}^{-1} \cdot ((\mathbf{F}\mathbf{C}_1[\sqrt{q}, \mathbf{x}]) \odot \cdots \odot (\mathbf{F}\mathbf{C}_r[\sqrt{q}, \mathbf{x}]))$$

is a polynomial RFM.

HOW MANY KK RANDOM FEATURES?

- Fix two arbitrary points on the unit sphere in $\mathbb{R}^d$; consider $K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^r$.
- How many random features are needed to approximate the kernel function at those points: $|K(\mathbf{x}, \mathbf{y}) - \langle \mathbf{Z}(\mathbf{x}), \mathbf{Z}(\mathbf{y}) \rangle| < \varepsilon$?
- Original estimate from Kar–Karnick 2012: $D \gtrsim \varepsilon^{-2} d^{2r}$.
- There are substantially fewer degree-r monomials in d features: $\binom{d+r-1}{r}$.

By carefully estimating the moments of $|K(\mathbf{x}, \mathbf{y}) - \langle \mathbf{Z}(\mathbf{x}), \mathbf{Z}(\mathbf{y}) \rangle|$, we show that D is independent of the input dimensionality: $D \gtrsim 4^r \varepsilon^{-2}$ suffices.
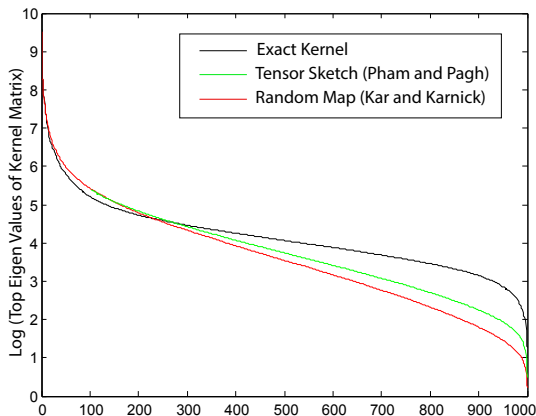
Uniform kernel approximation (G et al. 2014)

Let $\mathbf{X} \subset \mathbb{R}^d$ be a set of n unit vectors. If $D \gtrsim (4 \log(n)^2)^{r+1} \varepsilon^{-2}$, then with constant probability

$$\max_{\mathbf{x}, \mathbf{y} \in \mathbf{X}} |K(\mathbf{x}, \mathbf{y}) - \langle \mathbf{Z}(\mathbf{x}), \mathbf{Z}(\mathbf{y}) \rangle| < \varepsilon$$

Thus the D needed for a fixed point-wise accuracy grows polylogarithmically in the number of data points.

POLYNOMIAL RFMS – LIMITATIONS
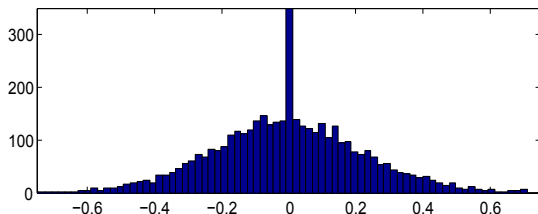

Rank deficiency of TensorSketch and Random Feature Maps

Figure: *

Histograms of weight vectors for MNIST two-class classifier learned using a 2^{12} dimensional Random Feature Map

- **Dilemma**: To approximate $K(\mathbf{x}, \mathbf{y})$ accurately, D might have to be increased to undesirably large values.
- **Opportunity**: since $\mathbf{Z}$ is rank-deficient, the same information can be encoded in fewer than D random linear combinations of its entries.

$$Z$$

$$\mathbb{R}^d$$

$$\mathbf{R} \cdot \mathbf{Z}$$
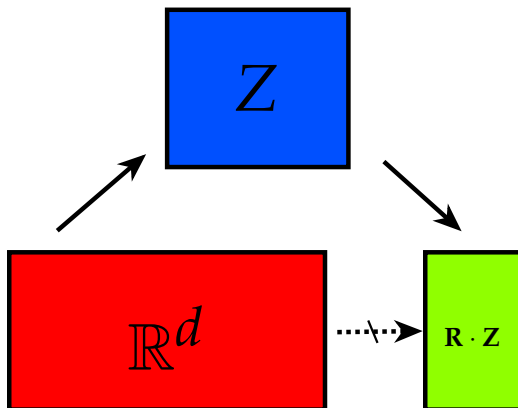
Algorithm 1: – CRAFTMAPS

Input: q, r, and D and E satisfying E < D
Output: CRAFTMap $\mathbf{G} : \mathbb{R}^d \to \mathbb{R}^E$, such that $\langle \mathbf{G}(\mathbf{x}), \mathbf{G}(\mathbf{y}) \rangle \approx K(\mathbf{x}, \mathbf{y})$
  Up Project: Construct $\mathbf{Z} : \mathbb{R}^d \to \mathbb{R}^D$ so that $\langle \mathbf{Z}(\mathbf{x}), \mathbf{Z}(\mathbf{y}) \rangle \approx K(\mathbf{x}, \mathbf{y})$
  Down Project: Let $\mathbf{R}$ be a JLT from $\mathbb{R}^D$ to $\mathbb{R}^E$ and set $\mathbf{G} = \mathbf{R} \cdot \mathbf{Z}$.
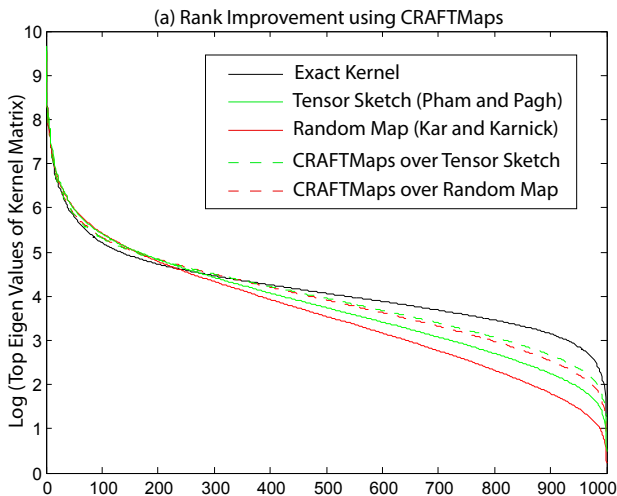

Since $\mathbf{R}$ is a JLT,

$$\langle \mathbf{G}(\mathbf{x}), \mathbf{G}(\mathbf{y}) \rangle \approx \langle \mathbf{Z}(\mathbf{x}), \mathbf{Z}(\mathbf{y}) \rangle \approx K(\mathbf{x}, \mathbf{y}).$$

NB: In our experiments, we take $\mathbf{R}$ to be a Subsampled Randomized Hadamard Transform matrix for speed.
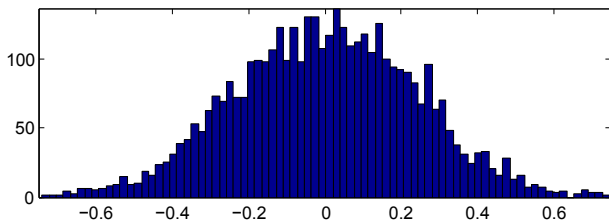
IMPROVEMENT WITH CRAFTMAPS

Figure: *

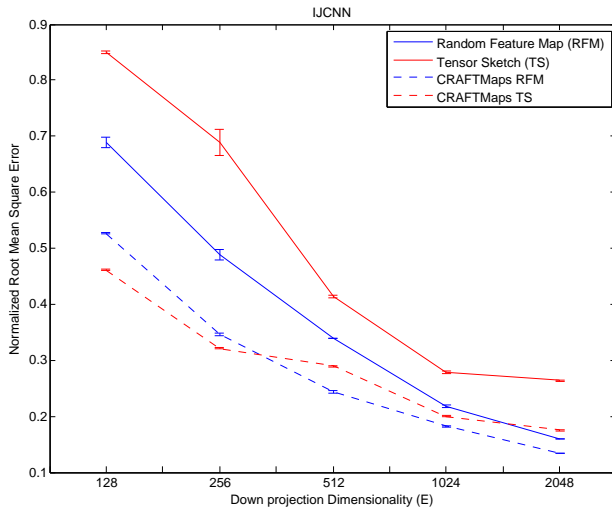Histogram of weight vectors for MNIST data learned using a 2^{12} dimensional KK-RFM with $D = 2^{14}$.

Figure: *

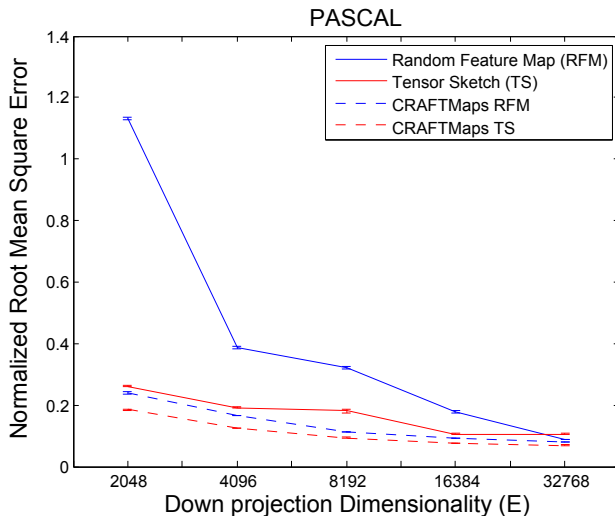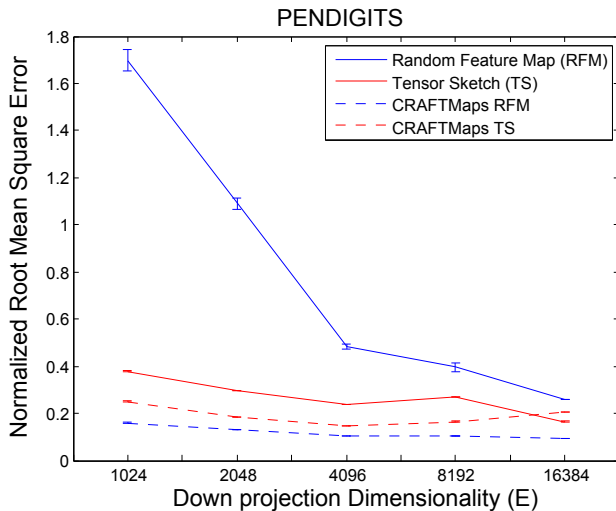NRMSE for polynomial kernel reconstruction using $r = 7$ and $q = 1$ on MNIST. Top row is for KK-RFM and the bottom table is for CRAFTMaps on KK-RFM.

CRAFTMAPS VS RFMS: KERNEL APPROXIMATION

PASCAL

PENDIGITS

CRAFTMAPS FOR MULTI-CLASS LEARNING USING ECOCS

ECOC idea: use multiple linear regressions to predict the individual coordinates, then assign the class corresponding to the nearest codeword
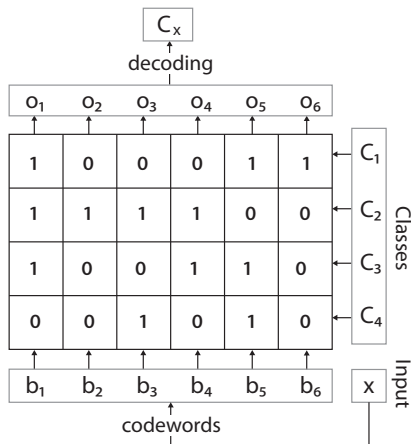
ADVANTAGES OF CRAFTMAPS + ECOC

1. Requires only *one pass* over the data
2. The Hessian is shared, so is only computed once.
3. The pipeline simple: is a series of matrix-matrix multiplications plus a matrix factorization (Cholesky of the Hessian). Amenable to GPU or distributed computation.

ECOC TEST ERRORS

| e-PENDIGITS	2^6	2^7	2^8	2^9	2^{10}
FastFood	8.74	5.03	3.08	2.85	2.71
RFF	5.83	4.25	2.63	2.17	2.08
KK-RFM	7.94	3.94	2.85	2.28	1.91
TS-RFM	11.20	4.57	2.37	1.80	1.77
CM KK-RFM	**7.43**	**3.57**	**2.28**	**1.97**	**1.57**
CM TS-RFM	8.03	3.80	2.37	2.05	1.74
Exact	9.29	3.74	2.74	2.31	2.87

Table: *

Test classification error percentages on a 10 class problem with 200 ECOCs;
r=9 and q=1; the first row is E, and $D = 8 \times E$.

d-COIL100	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
FastFood	8.25	7.83	6.80	6.32	5.21
RFF	8.14	7.36	6.50	5.97	4.81
KK-RFM	11.11	7.55	6.33	5.05	4.83
TS-RFM	10.08	7.19	5.69	4.75	4.27
CM KK-RFM	8.94	6.86	5.47	4.52	4.08
CM TS-RFM	**8.16**	**5.97**	**4.75**	**4.02**	**3.96**
Exact	9.55	——	——	——	——

Table: *

Test classification error percentages on a 100 class problem with 200 ECOCs;
r=5 and q=1; the first row is E, and $D = 8 \times E$.

TIME-ACCURACY TRADEOFFS
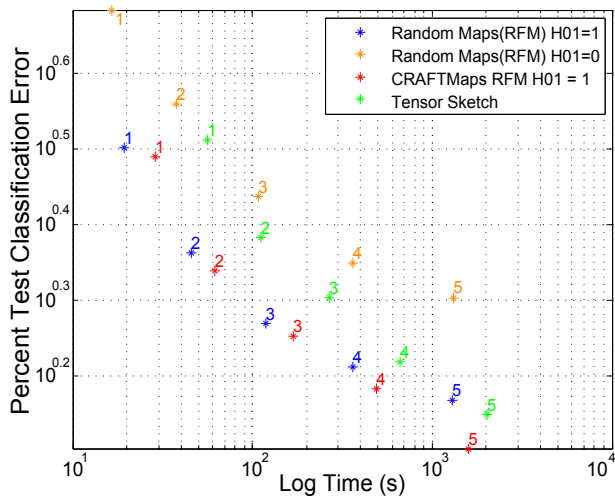
Word embeddings are vector representations of words such that simple geometric manipulation of the vectors are semantically meaningful:

$$\mathbf{v}_{Paris} - \mathbf{v}_{France} + \mathbf{v}_{Italy} \approx \mathbf{v}_{Rome}$$

$$\mathbf{v}_{king} - \mathbf{v}_{man} + \mathbf{v}_{woman} \approx \mathbf{v}_{queen}$$

Multiple models used to fit word embeddings exhibit similar properties of additivity. All are trained using only word cooccurrence information taken from a training corpus.
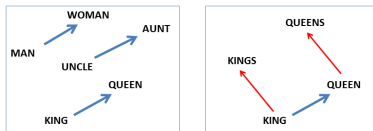
Recurrent Neural Networks (Mikolov et al. 2013)
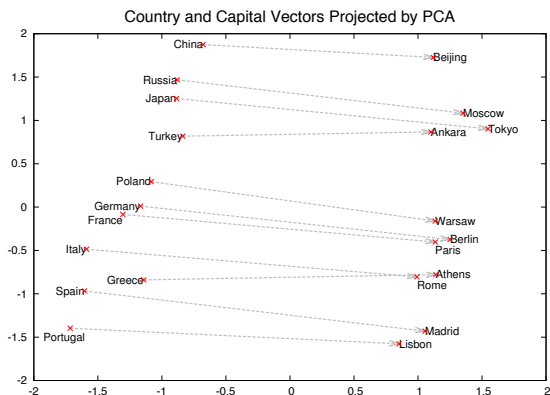
Word2Vec word embedding (Mikolov et al. 2013)

Basic Question

Why do word embeddings have the property of additively capturing semantics?

PRIOR WORK

The Rand-Walk model of (Arora et al. 2016):

▶ *Assumes* the training corpus is generated by a log-linear discourse model

$$\mathbb{P}(\text{word } w \text{ occurs at time } t \mid \mathbf{c}_t) \propto \exp(\langle \mathbf{v}_w, \mathbf{c}_t \rangle)$$

▶ *Assumes a priori* that the discourse vector is a Markov chain with uniform stationary distribution on the sphere, and the word vectors are isotropic

▶ Establishes that the word vectors can be recovered by factorizing the pointwise mutual information matrix

$$\text{PMI}(\mathbf{w}_1, \mathbf{w}_2) := \log \left(\frac{p(w_1, w_2)}{p(w_1)p(w_2)} \right) \approx \langle \mathbf{v}_{w1}, \mathbf{v}_{w2} \rangle$$

▶ *Explains* why analogies can be solved approximately using linear relationships under this model

▶ *Does not explain* linear compositionality of word embeddings

OUR WORK

- ▶ *No a priori assumptions* on the distribution of the word vectors
- ▶ Applies to embeddings learned using the canonical *Word2Vec* algorithm
- ▶ *Identifies the correct nonlinear operation* for composing multiple words
- ▶ Identifies an assumption on the corpus that leads to linear compositionality and linear solvability of analogies

LANGUAGE MODELS

Language models model the probability of a sequence of words

$$\mathbb{P}(w_1, \ldots, w_m)$$

- Useful for speech recognition, machine translation, information retrieval, NLP, ...
- Simplest language models, n-gram models, are Markov models based on observed frequencies of n-tuples
- n-gram models require an unreasonable amount of data to get accurate models for large n — the "data-sparsity" problem

NEURAL LANGUAGE MODELS

- ▶ address the data-sparsity problem with *n*-gram models (Bengio et al. 2003)
- ▶ learn a word embedding vector $\mathbf{C}(w)$ for each word
- ▶ maximize the (regularized) log-likelihood of the training data

$$L(\mathbf{C}, \boldsymbol{\theta}) = \frac{1}{T} \sum_t \log \mathbb{P}(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) + R(\boldsymbol{\theta}),$$

where

$$\mathbb{P}(w_t = i | w_{t-1}, \ldots, w_{t-n+1})$$
$$:= g(i, \mathbf{C}(w_{t-1}), \ldots, \mathbf{C}(w_{t-n+1}), \boldsymbol{\theta}).$$

NEURAL LANGUAGE MODEL OF (BENGIO ET AL. 2003)
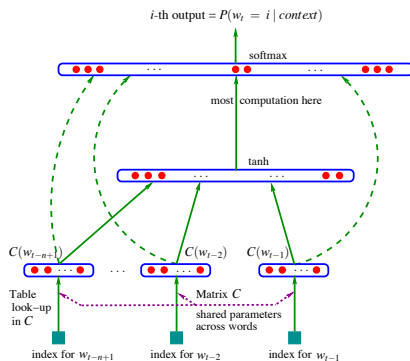
with

$$\mathbb{P}(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

$$\mathbf{y} = \mathbf{b} + \mathbf{W}\mathbf{x} + \mathbf{U}\tanh(\mathbf{d} + \mathbf{H}\mathbf{x})$$

Since (Bengio et al. 2003) many similar but simpler and faster models have been proposed. The most well-known:

- ▶ *Word2Vec* (canonical, hierarchical softmax, negative sampling) (3 papers in 2013; 2825 citations)
- ▶ *Global Vectors for Word Representation* (2014; 335 citations)
- ▶ *Noise-contrastive Log-Bilinear models* (2013; 93 citations)

These all the share the properties:

- ▶ Based on co-occurrence counts within a context width
- ▶ Involve nonlinear dimensionality reduction

CANONICAL (SKIPGRAM) WORD2VEC

Based on the distributional hypothesis of Harris and Firth: *words that occur in similar contexts have similar meanings*

… it was the best of times, it was the worst of times …
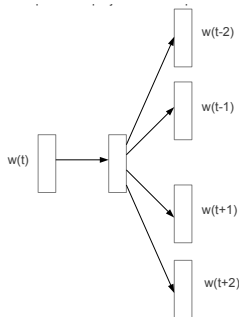
$$\frac{1}{T} \sum_{i=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)$$

$$p(w_j | w_i) = \frac{e^{\langle \mathbf{u}_i, \mathbf{v}_j \rangle}}{\sum_{k=1}^{n} e^{\langle \mathbf{u}_i, \mathbf{v}_k \rangle}}$$

DEFINING COMPOSITIONALITY

What does it mean for a word to mean the same as a sequence of words, e.g. 'king' meaning the same as 'royal man'?

Ideally, that all for all context words c

$$p(c|w_1, \ldots, w_m) = p(c|w).$$

But because we have a finite vocabulary, we'll find the best word in the KL-divergence sense:

$$\min_w D_{\mathrm{KL}}(p(\cdot|w_1, \ldots, w_m) \,||\, p(\cdot|w))$$

To solve this, we need to determine

$$p(\cdot|w_1, \ldots w_m)$$

Assuming that the prior distribution over the words is uniform, Bayes rule gives that the probability of a context word given multiple predictor words factors as

$$p(c|w_1, \ldots, w_m) = \Gamma \prod_{i=1}^{m} p(c|w_i),$$

where the constant Γ is independent of c.

Additive compositionality of Word2Vec (G et al. 2016)

Assume $p(w)$ is uniform, then given words $w_1, \ldots, w_m$, the word that minimizes

$$\min_w D_{\mathrm{KL}}(p(\cdot|w_1, \ldots, w_m) \,||\, p(\cdot|w))$$

is the word with predictor embedding

$$\mathbf{u}^* = \sum_{i=1}^{m} \mathbf{u}_i$$

if such a word exists.

ANALOGY ANSWERING

Using Word2Vec (canonical and approximate) models as well as other word embeddings, (Mikolov et al. 2013) found that multiple simple relationships were automatically satisfied additively by the learned embeddings.

For all these embeddings, to answer analogy questions of the form

$$w_a : w_b :: w_c : ?$$

e.g., man : king :: woman : ?, they used the procedure

1. compute

$$\mathbf{y} = (\mathbf{u}_b - \mathbf{u}_a) + \mathbf{u}_c$$

2. find

$$w = \mathrm{argmax}_w \frac{\langle \mathbf{u}_w, \mathbf{y} \rangle}{\|\mathbf{u}_w\|_2 \|\mathbf{y}\|_2}$$

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Examples from the analogies test dataset (Mikolov et al. 2013)

Model	Semantic-Syntactic Word Relationship test set	
Architecture	Semantic Accuracy [%]	Syntactic Accuracy [%]
RNNLM	9	36
NNLM	23	53
CBOW	24	64
Skip-gram	55	59

Results from training language models using 640-dimensional word vectors on a 320M word corpus with 82K vocabulary (Mikolov et al. 2013). This is the canonical skip-gram Word2Vec model.

ANALOGY ANSWERING, JUSTIFIED

How to answer the analogy China : Beijing :: USA : ?

We (conceptually) expand our vocabulary to include relationships as unseen predictor words, e.g.

$$p(\cdot \mid \text{is capital}, \text{China}) \approx p(\cdot \mid \text{Beijing})$$
$$p(\cdot \mid \text{is capital}, \text{USA}) \approx p(\cdot \mid \text{Washington, D.C.})$$

Assume the prior over the words is unifrom. From our results on additive compositionality, we have

$$\mathbf{u}_{\text{is capital}} + u_{\text{China}} \approx \mathbf{u}_{\text{Beijing}}$$
$$\mathbf{u}_{\text{is capital}} + u_{\text{USA}} \approx \mathbf{u}_{\text{Washington, D.C.}}$$

So it follows that

$$u_{\text{USA}} + \left(u_{\text{Beijing}} - u_{\text{China}} \right) \approx \mathbf{u}_{\text{Washington, D.C.}}$$

is the desired solution

NONUNIFORM PRIORS

In general, the prior distribution is nonuniform (Zipf!). In this case,

$$p(c|w_1, \ldots, w_m) = \Gamma_c \prod_{i=1}^{m} p(c|w_i)$$

and the operation that gives compositionality is nonlinear.

Nonlinear compositionality (G et al. 2016)

Given words $w_1, \ldots, w_m$, the word that minimizes

$$\min_w D_{\mathrm{KL}}(p(\cdot|w_1, \ldots, w_m) \,||\, p(\cdot|w))$$

is the word whose predictor embedding satisfies

$$\mathbb{E}[\mathbf{v}_c \,|\, w_1, \ldots, w_m] = \mathbb{E}[\mathbf{v}_c \,|\, w]$$

if such a word exists.

Word2Vec's objective can be specified entirely in terms of the co-occurence matrix **G** and embedding matrices **U**, **V**:

- G_{ij} is the number of times w_i and w_j co-occur within a context window in the corpus
- The rows of **U** and **V** give the embeddings for w_i

Batch Formulation of Word2Vec (G et al. 2016)

Word2Vec seeks maximizers **U** and **V** of

$$\max_{\mathbf{U},\mathbf{V}} \text{Trace}(\mathbf{G}\mathbf{U}\mathbf{V}^T) - \mathbf{1}^T\mathbf{G}\log(e^{\mathbf{U}\mathbf{V}^T}\mathbf{1}).$$

This follows directly from expanding out the objective stated previously. Note that

$$(\mathbf{G1})_i = G_i \propto p(w_i)$$

So Word2Vec is trying to maximize a matrix inner-product subject to an entropy regularization.

Define the weighted row-normalized KL-divergence of two matrices by

$$D_{\mathrm{KL}}^{\omega}(\mathbf{A} \,||\, \mathbf{B}) = \sum_i \omega_i D_{\mathrm{KL}}(\hat{\mathbf{a}}_i \,||\, \hat{\mathbf{b}}_i)$$

Normalizing the rows of **G** yields a collection of empirical distributions $p(w_j|w_i)$ characterizing each word w_i. Likewise, normalizing the rows of $e^{\mathbf{UV}^T}$ yields a collection of probability distributions. Word2Vec attempts to minimize a weighted sum of the divergence between the corresponding distributions.

> ### KL-divergence minimization formulation of Word2Vec (G et al. 2016)
>
> Word2Vec minimizes
>
> $$D_{\mathrm{KL}}^{\omega}\left(\mathbf{G} \,||\, e^{\mathbf{UV}^T}\right)$$
>
> for $\omega = \mathbf{G1}$.

This follows from observing that maximizing likelihood is the same as minimizing KL divergence.

According to the principle of sufficient dimensionality reduction (Globerson et al. 2003), given co-occurence information $\mathbf{G}$ for two discrete random variables X and Y with n and m states respectively, a model of the form

$$\mathbf{G} = e^{\mathbf{U}\mathbf{V}^T}, \quad \mathbf{U} \in \mathbb{R}^{n \times d} \text{ and } \mathbf{V} \in \mathbb{R}^{m \times d}$$

preserves the maximum amount of mutual information between X and Y and this model can be obtained by minimizing

$$D_{\mathrm{KL}}(z_G^{-1}\mathbf{G} \,||\, z_e^{-1}e^{\mathbf{U}\mathbf{V}^T}).$$

That is, rescale $\mathbf{G}$ and $e^{\mathbf{U}\mathbf{V}^T}$ to be *joint* probability distributions over (X, Y) and find the closest exponential factorization in the KL divergence.
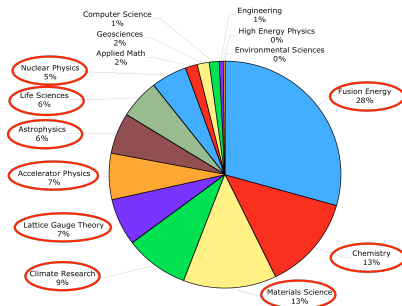
WORD2VEC FITS SDR

(G et al. 2016)

Word2Vec fits an SDR model.

(Globerson et al. 2003) shows that the SDR model performs well at text categorization and information retrieval, but fitting an SDR model is expensive. After each update to $\mathbf{U}$, $\mathbf{V}$, the normalizer $z_e = \sum_{ij} e^{\mathbf{U}\mathbf{V}^T}$ must be recalculated.

This suggests fitting Word2Vec-like models with SGD-like algorithms instead, as Word2Vec only needs to recompute row-wise normalization constants.
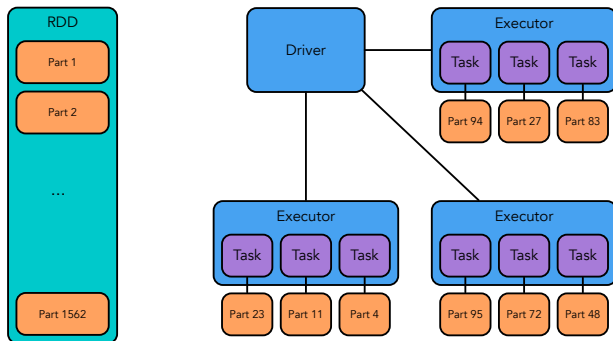
LOW-RANK APPROXIMATIONS IN SPARK
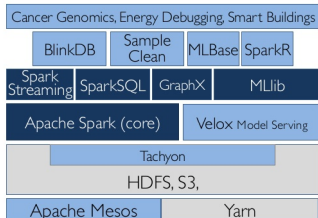
At the AMPLab, we are interested in adding linear algebra++ algorithms to the Berkeley Data Analytics Stack.

Our collaborators at NERSC are investigating Spark as a unified platform for data analytics pipelines with diverse scientific inputs.

- ▶ Spark is a distributed programming environment that implements a data-parallel programming model.
- ▶ The *driver* forms the computational DAG, schedules tasks on the *executors*
- ▶ All data is stored as *Resilient Distributed Datasets* (RDDs) that are partitioned across the executors and optionally cached in memory.
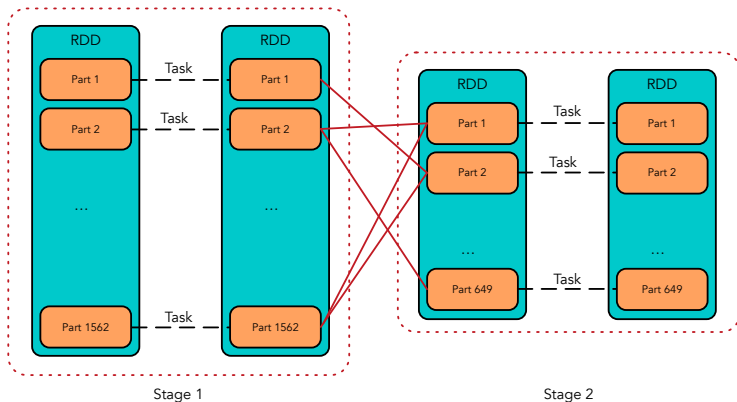
- Each *job* (DAG) is broken into *stages*
- Stages are divided into parallel, independent *tasks*
- Communication happens only between stages

Spark's performance generally depends on **problem scale** and **level of parallelism** (DAG complexity), as well as the amount of communication required, as measured in **stages**.
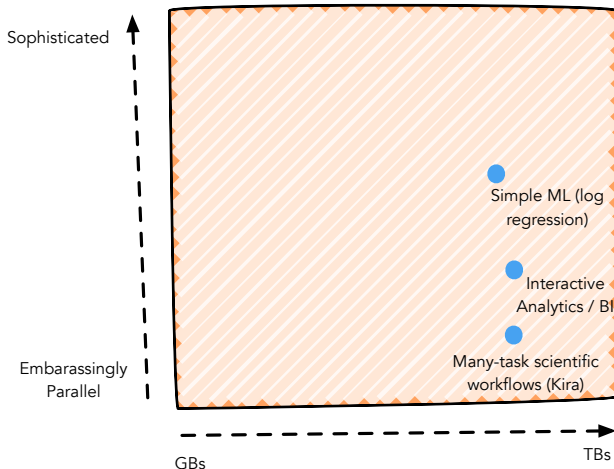
OUR GOALS

- *Provide implementations* of low-rank factorizations (PCA and NMF) in Spark
- Successfully apply low-rank matrix factorizations to *TB-scale scientific datasets* in Spark
- *Quantify the gaps* between highly-tuned C+MPI and Spark implementations
- *Investigate the scalability* of Spark for linear algebra on HPC platforms

DATASETS

Science Area	Format/Files	Dimensions	Size
Daya Bay	HDF5/1	$1,099,413,914 \times 192$	1.6TB
Ocean	HDF5/1	$6,349,676 \times 46,715$	2.2TB
Atmosphere	HDF5/1	$26,542,080 \times 81,600$	16TB

Daya Bay — neutrino sensor array measurements; used for NMF

Ocean and Atmosphere — climate variables (ocean temperature, atmospheric humidity) measured on a 3D grid at 3 or 6 hour intervals over about 30 years; used for PCA

These datasets are tall and skinny, so 1D partitioning makes sense and fits nicely into Spark's RDD formalism.

RUNNING TIMES

On the NERSC Cori supercomputer (Cray XC40):

- ▶ 1630 compute nodes
- ▶ 128 GB/node
- ▶ 32 2.3GHz Haswell cores/node

Algo	Size	# Nodes	MPI Time (s)	Spark Time (s)
NMF	1.6 TB	50	66	278
		100	45	207
		300	30	70
PCA	2.2 TB	100	94	934
		300	60	827
		500	56	1160
	16 TB	MPI: 1600 Spark: 1522	160	4175

PERFORMANCE GAPS

For data analysis, performance with IO is more relevant. For HPC tasks, performance without IO is more relevant.

Algo	# Nodes	Gap with I/O	Gap without I/O
NMF	50	$4\times$	$21.2\times$
	100	$4.6\times$	$14.9\times$
	300	$2.3\times$	$15.7\times$
PCA	100	$10.2\times$	$12.6\times$
	300	$14.5\times$	$24.7\times$
	500	$22\times$	$39.3\times$
	MPI: 1600 Spark: 1522	$26\times$	$43.8\times$

PCA ALGORITHM

The rank-*k* truncated PCA of a matrix **A** with centered rows is given by

$$\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$$

where $\mathbf{\Sigma}_k$ contains the top *k* singular values of **A** and $\mathbf{U}_k, \mathbf{V}_k$ contain the corresponding singular vectors.
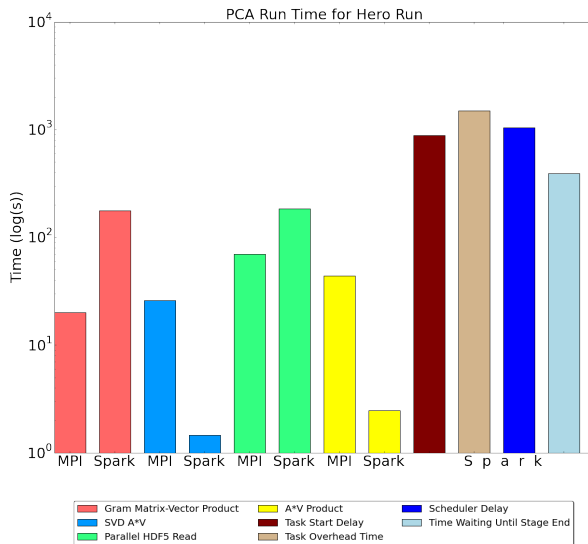
We use the following algorithm to compute $\mathbf{A}_k$ when **A** is tall and skinny:

1. Compute $\mathbf{V}_k$ as the top *k* eigenvectors of $\mathbf{A}^T\mathbf{A}$ using a Lanczos eigensolver (ARPACK) with multiple rounds of distributed matrix-vector multiplies.
2. Compute $\mathbf{AV}_k$ and collect on the driver.
3. On the driver, compute the SVD of $\mathbf{AV}_k$ to get $\mathbf{U}_k$ and $\mathbf{\Sigma}_k$.
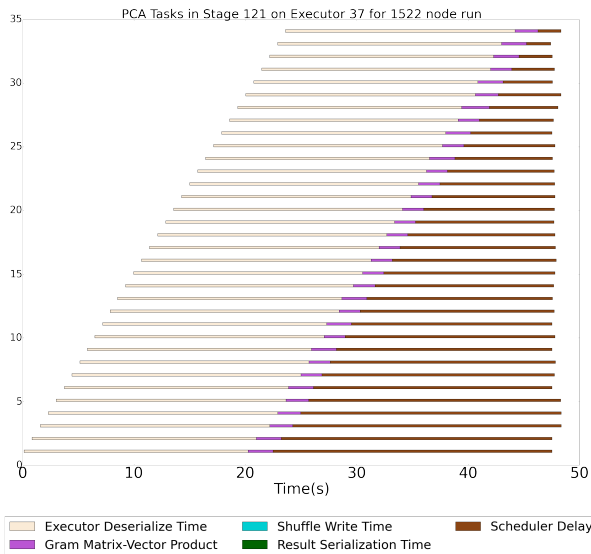
SCALING FOR PCA
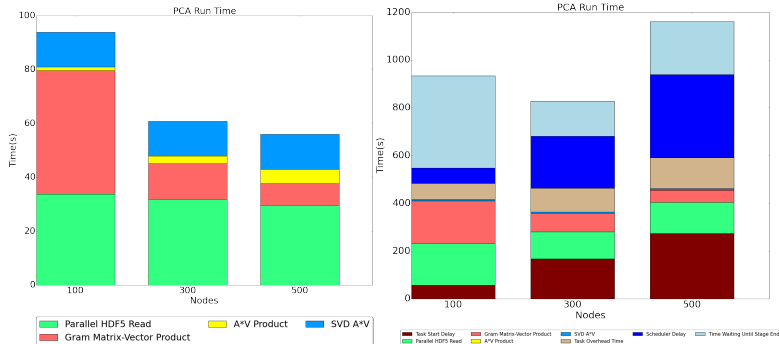
Running time breakdown of PCA on 2.2TB Ocean data at node counts of 100, 300 and 500. MPI is to the left, Spark to the right.

Running time comparison of MPI PCA and Spark PCA for the 1600 node run.

PCA Tasks in Stage 121 on Executor 37 for 1522 node run

Time(s)

Executor Deserialize Time   Shuffle Write Time   Scheduler Delay
Gram Matrix-Vector Product   Result Serialization Time

A timeline of tasks on one node during a multiply-Gramian stage during the 16TB PCA run.

NMF ALGORITHM

NMF is used when observations are positive, and assumed to be positive linear combinations of basis vectors (e.g. medical imaging modalities, hyperspectral imaging)

The NMF factorization is given by

$$(\mathbf{W}, \mathbf{H}) = \operatorname*{argmin}_{\substack{\mathbf{W} \geq 0 \\ \mathbf{H} \geq 0}} \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2$$

We use the one-pass approach of (Benson et al. 2014) that assumes $\mathbf{W}$ can be selected from the columns of $\mathbf{A}$. Then the problem reduces to that of taking a tall-skinny QR of $\mathbf{A}$ and postprocessing the $\mathbf{R}$ factor, on the driver, to choose those columns.

When **A** is tall and skinny, its QR factorization can be computed efficiently using the TSQR algorithm:
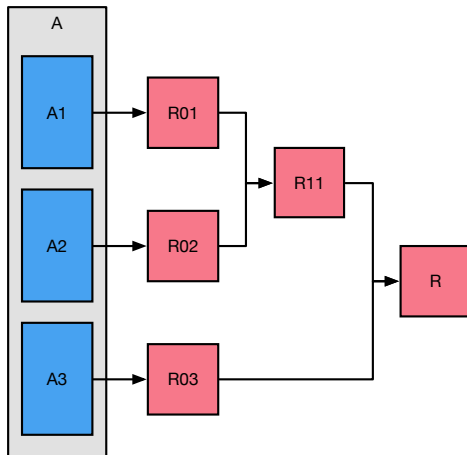
This uses a tree reduction, so communicates minimally between stages, and requires only one pass over the matrix.

SCALING FOR NMF
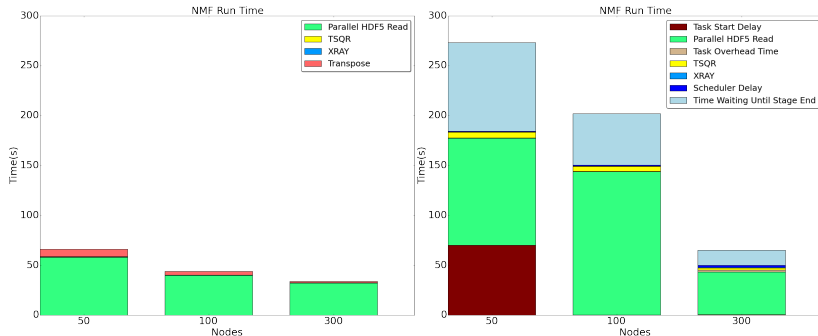
Running time breakdown of NMF on 1.6TB Daya Bay data at node counts of 50, 100 and 300. MPI is to the left, Spark to the right.

LESSONS LEARNED

- Even with favorable data (tall and skinny) and well-adapted algorithms, *Spark is 2x-10x slower than MPI when IO is included, 10x-40x when IO is not included*
- *Spark overheads are orders of magnitude higher than the computations*
- The straggler effect from large overheads means the *impact of overheads increases as concurrency does*
- This gap suggests it is worthwhile to *interface MPI-based codes with Spark or augment Spark's communication model*

RECAP

We considered four instances of low-rank approximation:

- ► *SPSD Sketches*, where we obtained bounds for the approximation error of randomized SPSD sketches

- ► *Polynomial Random Features*, where guidance was provided for the number of features needed to achieve low kernel approximation error, and the theory motivated an effective algorithm for futher reducing the number of features needed

- ► *Word2Vec*, where we offered a rigorous justification for the observation that word embeddings capture semantics additively, and the theory led to a faster algorithm for fitting SDR models.

- ► *Low-rank factorization in Spark*, where we quantified the slow-down of large-scale low-rank factorization algorithms in Spark relative to their performance in C+MPI.