Large-Scale Sparse PCA through Low-rank Approximations

Alex Dimakis
UT Austin

Based on Joint work with:
Dimitris Papailiopoulos
Overview: PCA and Sparse PCA

- Principal Component Analysis (PCA) is a classical algorithm for dimensionality reduction, clustering etc.

- Sparse PCA is a very useful variant because of interpretability

- We present a new algorithm for Sparse PCA that is fast for large data sets.

- We present novel approximation guarantees.

- We test on a large twitter data set (millions of tweets).
Tweets to vectors

Each tweet as a long (50K), super-sparse vector (5-10 non-zeros) with 1s in word indices

God, I Love the IMF

word1
word2
word n
Data Sample Matrix

We collect all tweet vectors in a sample matrix of size $n \times m$

$$S = \begin{pmatrix}
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots 
\end{pmatrix} \text{m tweets}

$$
Correlation matrix

\[ A = S S^T \]
vanilla PCA

\[
\text{arg} \, \max \ x^T A x \\
\|x\|_2 = 1
\]

Largest Eigenvector.
Maximizes `explained variance' of the data set
Very useful for dimensionality reduction
Easy to compute
PCA finds An `EigenTweet’

Finds a vector that **closely matches** most tweets

\[
\max \left\| x^T S \right\|^2
\]

i.e., a vector that **maximizes the sum of projections** with each tweet
The problem with PCA

• Top Eigenvector will be dense!

Dense =
A tweet with thousands of words
(makes no sense)

<table>
<thead>
<tr>
<th>Eurovision</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protests</td>
<td>0.02</td>
</tr>
<tr>
<td>Greece</td>
<td></td>
</tr>
<tr>
<td>Morning</td>
<td></td>
</tr>
<tr>
<td>Deals</td>
<td></td>
</tr>
<tr>
<td>Engage</td>
<td></td>
</tr>
<tr>
<td>Offers</td>
<td></td>
</tr>
<tr>
<td>Uprising</td>
<td></td>
</tr>
<tr>
<td>Protest</td>
<td></td>
</tr>
<tr>
<td>Elections</td>
<td></td>
</tr>
<tr>
<td>teachers</td>
<td></td>
</tr>
<tr>
<td>Summer</td>
<td></td>
</tr>
<tr>
<td>support</td>
<td></td>
</tr>
<tr>
<td>Schools</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Crisis</td>
<td>0.001</td>
</tr>
<tr>
<td>Earthquake</td>
<td></td>
</tr>
<tr>
<td>IMF</td>
<td></td>
</tr>
</tbody>
</table>
The problem with PCA

- Top Eigenvector will be dense!

  Dense =
  A tweet with thousands of words
  (makes no sense)

- We want super sparse

  Sparse = Interpretable
Sparse PCA

$$x_* = \arg \max \quad x^T A x.$$  
$$\|x\|_2 = 1, \|x\|_0 = k$$
Sparse PCA

\[ x_\star = \arg \max x^T A x. \]
\[ ||x||_2 = 1, ||x||_0 = k \]

NP hard (Moghaddam et al., 2006)

Sparse PCA

\[ x_\ast = \arg \max_{\|x\|_2=1, \|x\|_0=k} x^T A x. \]

NP hard (Moghaddam et al., 2006)


Our result

We present a novel combinatorial algorithm for sparse PCA. Obtain general provable approximation guarantees.

\[ x_\ast = \arg \max_{\|x\|_2=1, \|x\|_0=k} x^T A x. \]

**Theorem:** For any desired accuracy parameter \( d \), our **Spannogram** algorithm runs in time \( O(n^d) \) and constructs a \( k \)-sparse vector \( x_d \) such that:

\[
x_d^T A x_d \geq (1 - \epsilon_d) x_\ast^T A x_\ast
\]

\[
\epsilon_d \leq \min \left\{ \frac{n}{k} \cdot \frac{\lambda_{d+1}}{\lambda_1}, \frac{\lambda_{d+1}}{\lambda_1^{(1)}} \right\}
\]
Corollaries

**Theorem:** For any desired accuracy parameter $d$, our Spannogram algorithm runs in time $O(n^d)$ and constructs a $k$-sparse vector $x_d$ such that:

$$x_d^T A x_d \geq (1 - \epsilon_d) x_*^T A x_*$$

$$\epsilon_d \leq \min \left\{ \frac{n}{k} \cdot \frac{\lambda_{d+1}}{\lambda_1}, \frac{\lambda_{d+1}}{\lambda_1^{(1)}} \right\}$$

Cor1: If there is any decay in the eigenvalues, i.e. $\lambda_1 > \lambda_d$ then there exists a constant $\delta$ s.t. for all linear size supports $k > \delta n$ we obtain a constant factor approximation to sparse PCA.
Corollaries

**Theorem:** For any desired accuracy parameter $d$, our Spannogram algorithm runs in time $O(n^d)$ and constructs a $k$-sparse vector $x_d$ such that:

$$x_d^T A x_d \geq (1 - \epsilon_d) x_*^T A x_*$$

$$\epsilon_d \leq \min \left\{ \frac{n}{k} \cdot \frac{\lambda_{d+1}}{\lambda_1}, \frac{\lambda_{d+1}}{\lambda_1^{(1)}} \right\}$$

Cor2: If there is a power law decay in the eigenvalues:

$$\lambda_i = C i^{-\alpha}$$

Then for any $\epsilon$ we can approximate Sparse PCA within a factor of $\epsilon$ in time polynomial in $n,k$

(but not in $1/\epsilon$) (PTAS approximation guarantees)
how it works

• 1. Approximate $A$ by best rank $d$ approximation $A_d$ (SVD)
how it works

1. Approximate $A$ by best rank $d$ approximation $A_d$ (SVD)

2. Use $A_d$ to obtain $n^d$ candidate supports (Spannogram)
how it works

• 1. Approximate $A$ by best rank $d$ approximation $A_d$ (SVD)

• 2. Use $A_d$ to obtain $n^d$ candidate supports (Spannogram)

• 3. Try $n^d$ candidate supports on $A$ and choose the best one.

• 4. Prove approximation guarantees
how it works for Rank d

If we knew the support of the sparse PC, it’s easy.
(Zero out everything except $k \times k$ submatrix of $A$, find largest eigenvector of that).
how it works for Rank $d$

If we knew the support of the sparse PC, it’s easy.
(Zero out everything except $k \times k$ submatrix of $A$, find largest eigenvector of that).

We can naively solve sparse PCA by testing all $(n \choose k)$ supports.
how it works for Rank $d$

If we knew the support of the sparse PC, it’s easy. (Zero out everything except $k \times k$ submatrix of $A$, find largest eigenvector of that).

We can naively solve sparse PCA by testing all $(n \choose k)$ supports.

Key lemma: If the matrix is rank $d$, only $O(\ n \ choose \ d\ )$ supports must be tested.
Rank d=1

Say d=1, i.e. $A_d$ is rank 1.

\[
A = \lambda_1 v_1 v_1^T
\]

\[
x^T A x = \lambda_1 x^T v_1 v_1^T x = \lambda_1 (v_1^T x)^2
\]
Rank d=1

Say d=1, i.e. A_d is rank 1.

\[
A = \lambda_1 v_1 v_1^T
\]

\[
x^T A x = \lambda_1 x^T v_1 v_1^T x = \lambda_1 (v_1^T x)^2
\]

Q: find a k-sparse vector that maximizes the inner product with a given vector \(v_1\).

Sort the absolute entries of \(v_1\) and keep the k largest.
Rank d=1

Say d=1, i.e. $A_d$ is rank 1.

$$A = \lambda_1 v_1 v_1^T$$

$$x^T A x = \lambda_1 x^T v_1 v_1^T x = \lambda_1 (v_1^T x)^2$$

Q: find a k-sparse vector that maximizes the inner product with a given vector $v_1$.

Sort the absolute entries of $v_1$ and keep the k largest.

Thresholding the largest eigenvector is a well-known heuristic for sparse PCA which is optimal when $A$ is rank 1.

There is **one candidate top-k support**, the support of the k largest entries of $v_1$. 
Rank $d=2$

$$A_2 = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T$$
Rank d=2

\[ A_2 = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T \]

Observation: There is a special vector \( v_c \) in the span of \( v_1, v_2 \) such that

\[ x^T A x = (v_c^T x)^2 \]
Observation: There is a special vector $v_c$ in the span of $v_1, v_2$ such that

$$A_2 = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T$$

We only need to find the support of the top $k$ elements of $v_c$

How many top-$k$ supports can there be in a two dimensional subspace?

$(n \text{ choose } k)$?
key combinatorial fact (2 dimensions)

\[ v_c = c_1 v_1 + c_2 v_2 \]
key combinatorial fact (2 dimensions)

\[ \mathbf{v}_c = c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 \]

if \( c_1=1, c_2=0 \), we get one top-k set, the top-k elements of \( \mathbf{v}_1 \).
If \( c_1=0, c_1=1 \), we get one more, the top-k elements of \( \mathbf{v}_2 \).
key combinatorial fact (2 dimensions)

\[ v_c = c_1 v_1 + c_2 v_2 \]

if \( c_1 = 1, c_2 = 0 \), we get one top-k set, the top-k elements of \( v_1 \).
If \( c_1 = 0, c_2 = 1 \), we get one more, the top-k elements of \( v_2 \).

As \( c = [c_1 \ c_2] \) is changing how many other top-k sets can appear?

\[
\binom{n}{k}
\]
key combinatorial fact (2 dimensions)

\[ v_c = c_1 v_1 + c_2 v_2 \]

if \( c_1 = 1, c_2 = 0 \), we get one top-k set, the top-k elements of \( v_1 \).
If \( c_1 = 0, c_2 = 1 \), we get one more, the top-k elements of \( v_2 \).

As \( c = [c_1 \ c_2] \) is changing how many other top-k sets can appear?

\[ \binom{n}{k} \times 4 \binom{n}{2} \]
key combinatorial fact (2 dimensions)

\[ \mathbf{v}_c = c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 \]

Use spherical variable transformation

\[ \mathbf{c} = [\sin \phi \; \cos \phi]^T \]

\[ \mathbf{v}_c = [\mathbf{v}_1 \mathbf{v}_2] \mathbf{c} \]
key combinatorial fact (2 dimensions)

\[ \mathbf{v}_c = c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 \]

Use spherical variable transformation

\[ \mathbf{c} = [\sin \phi \quad \cos \phi]^T \]

\[ \mathbf{v}_c = [\mathbf{v}_1 \mathbf{v}_2] \mathbf{c} = \begin{bmatrix}
  v_1(1) \sin(\phi) + v_2(1) \cos(\phi) \\
  \vdots \\
  v_1(n) \sin(\phi) + v_2(n) \cos(\phi)
\end{bmatrix} \]
The Spannogram

\[ \mathbf{v}(\phi) = [\mathbf{v}_1 \ \mathbf{v}_2]^T \mathbf{c}(\phi) = \begin{bmatrix} v_1(1) \sin(\phi) + v_2(1) \cos(\phi) \\ \vdots \\ v_1(n) \sin(\phi) + v_2(n) \cos(\phi) \end{bmatrix} \]

- Each element is a **continuous curve in** \( \phi \)
The Spannogram

\[ \mathbf{v}(\phi) = [\mathbf{v}_1 \, \mathbf{v}_2]^T \mathbf{c}(\phi) = \begin{bmatrix} v_1(1) \sin(\phi) + v_2(1) \cos(\phi) \\ \vdots \\ v_1(n) \sin(\phi) + v_2(n) \cos(\phi) \end{bmatrix} \]

- Each element is a continuous curve in \( \phi \)

\[ \mathbf{v}(\phi) = [\mathbf{v}_1 \, \mathbf{v}_2]^T \mathbf{c}(\phi) = \begin{bmatrix} \mathbf{v}_1(\phi) \\ \mathbf{v}_2(\phi) \\ \mathbf{v}_3(\phi) \\ \mathbf{v}_4(\phi) \\ \mathbf{v}_5(\phi) \end{bmatrix} \]

\[ n=5, k=3 \]

Top k set: \{2, 5, 1\}
The Spannogram

\[ \mathbf{v}(\phi) = [\mathbf{v}_1 \mathbf{v}_2]^T \mathbf{c}(\phi) = \begin{bmatrix} v_1(1) \sin(\phi) + v_2(1) \cos(\phi) \\ \vdots \\ v_1(n) \sin(\phi) + v_2(n) \cos(\phi) \end{bmatrix} \]

- Each element is a continuous curve in \( \phi \)

\[ \mathbf{v}(\phi) = [\mathbf{v}_1 \mathbf{v}_2]^T \]

\[ \mathbf{c}(\phi) = \begin{bmatrix} v_1(1) \sin(\phi) + v_2(1) \cos(\phi) \\ \vdots \\ v_1(n) \sin(\phi) + v_2(n) \cos(\phi) \end{bmatrix} \]

n=5, k=3
Top k set: \{5,2,1\}
The Spannogram

- Lets count top-k sets.
- $n$ lines
- every pair of lines intersects in exactly 2 points.

$$2 \binom{n}{2}$$ Intersection points
How many top-k supports can there be in a $d$-dimensional subspace of $\mathbb{R}^n$?

Theorem: There are at most

$$2^{d-1} \binom{d}{\lceil d/2 \rceil} \binom{n}{d}$$

top k-sets in a general position $d$-dimensional subspace.
general Rank $d$

$$v_c = c_1 v_1 + c_2 v_2 + \ldots + c_d v_d$$

How many top-$k$ supports can there be in a $d$-dimensional subspace of $\mathbb{R}^n$?

$O(n^d)$ and the spannogram algorithm constructs them explicitly.
Experiments

<table>
<thead>
<tr>
<th>$m \times n$</th>
<th>$k$</th>
<th>$#PCs$</th>
<th>*japan</th>
<th>1-5 May 2011</th>
<th>May 2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>12k x 15k</td>
<td>10</td>
<td>5</td>
<td>0.600</td>
<td>0.815</td>
<td>0.885</td>
</tr>
<tr>
<td>267k x 148k</td>
<td>4</td>
<td>7</td>
<td>0.595</td>
<td>0.869</td>
<td>0.915</td>
</tr>
<tr>
<td>1.9mil x 222k</td>
<td>5</td>
<td>3</td>
<td>0.940</td>
<td>0.936</td>
<td>0.954</td>
</tr>
</tbody>
</table>

3 experiments on a large-twitter data set. (1.9M Tweets total over a few months).
Experiments (5 days in May 2011)

k=10, top 4 sparse PCs for the data set (65,000 tweets)

skype, microsoft, acquisition, billion, acquired, acquires, buy, dollars, acquire, google
eurovision greece lucas finals final stereo semifinal contest greek watching
love received greek know damon amazing hate twitter great sweet
downtown athens murder years brutal stabbed incident camera year crime
Experiments (5 days in May 2011)

k=10, top 4 sparse PCs for the data set (65,000 tweets)

skype, microsoft, acquisition, billion, acquired, acquires, buy, dollars, acquire, google
eurovision greece lucas finals final stereo semifinal contest greek watching
love received greek know damon amazing hate twitter great sweet
downtown athens murder years brutal stabbed incident camera year crime

FullPath:
eurovision finals greek greece lucas semifinal final contest stereo watching
love received damon greek hate know amazing sweet great songs
skype microsoft billion acquisition acquires acquired buying dollars official google
Twitter facebook welcome account good followers census population home starts
Experiments (5 days in May 2011)

k=10, top 4 sparse PCs for the data set (65,000 tweets)

skype, microsoft, acquisition, billion, acquired, acquires, buy, dollars, acquire, google

eurovision greece lucas finals final stereo semifinal contest greek watching

love received greek know damon amazing hate twitter great sweet
downtown athens murder years brutal stabbed incident camera year crime

FullPath:
eurovision finals greek greece lucas semifinal final contest stereo watching

love received damon greek hate know amazing sweet great songs

skype microsoft billion acquisition acquires acquired buying dollars official google

Twitter facebook welcome account good followers census population home starts

Tpower:
greece greece love loukas finals athens final stereo country sailing

Rank1:
greece love lucas finals greek athens finals stereo country camera
Feature elimination

\[ \mathbf{v}(\phi) = [\mathbf{v}_1 \ \mathbf{v}_2]^T \mathbf{c}(\phi) = \begin{bmatrix} v_1(1) \sin(\phi) + v_2(1) \cos(\phi) \\ \vdots \\ v_1(n) \sin(\phi) + v_2(n) \cos(\phi) \end{bmatrix} \]

- Each element is a continuous curve in \( \phi \)

Red line has no hope of being in a top-k set for \( k = 2 \).
Conclusions

• We presented a novel combinatorial algorithm for Sparse PCA
Conclusions

• We presented a novel combinatorial algorithm for Sparse PCA
• Constant factor approximation for any reasonable matrix
• Arbitrary approximation for power-law decay
Conclusions

• We presented a novel combinatorial algorithm for Sparse PCA
• Constant factor approximation for any reasonable matrix
• Arbitrary approximation for power-law decay
• General spectral bound
Conclusions

- We presented a novel combinatorial algorithm for Sparse PCA
- Constant factor approximation for any reasonable matrix
- Arbitrary approximation for power-law decay
- General spectral bound
- Empirically outperforms previous state of the art
- Parallel Mapreduce implementation?
fin
The Spanogram

• Lets revisit the “variable vector”

\[ \mathbf{v}(\phi) = [\mathbf{v}_1 \mathbf{v}_2]^T \mathbf{c}(\phi) = \begin{bmatrix} v_1(1) \sin(\phi) + v_2(1) \cos(\phi) \\ \vdots \\ v_1(n) \sin(\phi) + v_2(n) \cos(\phi) \end{bmatrix} \]

• Each element is a continuous curve in \( \phi \)
Rank-2 Approximation

• Rank-2 Approximation $R_2 = v_1v_1^T + v_2v_2^T$

• The Sparse PC is

$$\arg \max \left\| [v_1, v_2]^T x \right\| \quad \text{subject to} \quad \left\| x \right\|_2 = 1, \left\| x \right\|_0 = K$$

• How to unlock the “low-rank-ness”? The key is a polar vector

$$c(\phi) = \begin{bmatrix} \sin \phi \\ \cos \phi \end{bmatrix}$$

• From the Cauchy Swartz Inequality we obtain

$$\left| c^T (\phi) [v_1, v_2] x \right| \leq \left\| [v_1, v_2] x \right\|$$

• Colinear polar vector achieves “=“
Rank-2 Approximation

• The sparse $\mathbf{x}$ of $\text{pair}(\mathbf{x}, \phi)$ that maximizes the left, maximizes the right:

$$\left| \mathbf{c}^T(\phi)[\mathbf{v}_1 \mathbf{v}_2]^T \mathbf{x} \right| \leq \left\| [\mathbf{v}_1 \mathbf{v}_2]^T \mathbf{x} \right\|$$

The sparse PC is associated with a polar vector that gives equality.

• So,

$$\max_{\mathbf{x}} \left\| [\mathbf{v}_1 \mathbf{v}_2]^T \mathbf{x} \right\| = \max_{\phi} \max_{\mathbf{x}} \left| \mathbf{c}(\phi)[\mathbf{v}_1 \mathbf{v}_2]^T \mathbf{x} \right|$$

Q: What happens if we fix the angle?