Approximate Principal Components Analysis of Large Data Sets

Daniel J. McDonald

Department of Statistics
Indiana University
mypage.iu.edu/∼dajmcdon

October 28, 2015

Motivation

Obligatory "data is big" slide

Modern statistical applications — genomics, neural image analysis, text analysis — have large numbers of covariates p

Also frequently have lots of observations n.

Need to do some dimension reduction

Many methods — multidimensional scaling, discriminant analysis, locally linear embeddings, Laplacian eigenmaps, and many others

Early motivation

Joey Richards dissertation on spectral embedding methods for astronomical applications.

> *As SCA relies upon eigen-decomposition, our training set size is limited to $\lesssim 10^4$ galaxies; we use the Nyström extension to quickly estimate diffusion coordinates for objects not in the training set.…we find that use of the Nyström extension leads to a negligible loss of prediction accuracy."*[1]

What is this Nyström and why does it work?

[1] Freeman et. al. "Photometric redshift estimation using spectral connectivity analysis." **MNRAS**.

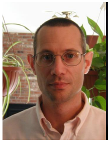
Lesson of the talk

Many statistical methods use (perhaps implicitly) a singular value decomposition (SVD).

The SVD is computationally expensive.

We want to understand the statistical properties of some approximations which speed up computation.

Collaborators



Musical reconstruction

Reconstruct an audio recording from a database of other recordings.

Application to “de-soloing”.

Examine the STFT of the recording and the database.

Sanna this summer: We’re using PCA, but there are too many frequencies, so we just subselect some of them.

Me: Wait, I have something better! And now I can play music in my talk!

Notation and intuition

Notation

- Observe $\widetilde{X}_1, \ldots, \widetilde{X}_n$, where $\tilde{X}_i \in \mathbb{R}^p$
- Form $\widetilde{\mathbf{X}} = \begin{bmatrix} \widetilde{X}_1 & \cdots & \widetilde{X}_n \end{bmatrix}^\top \in \mathbb{R}^{n \times p}$
- Call $\mathbf{X} = \widetilde{\mathbf{X}} - \overline{\mathbf{X}}$ where $\overline{\mathbf{X}} = n^{-1}\mathbf{1}\mathbf{1}^\top \widetilde{\mathbf{X}}$

Notation (cont.)

- Write the SVD of $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{U}\Lambda\mathbf{V}^\top,$$

- For general rank r matrices $\mathbf{A}$ we write

$$\mathbf{A} = U(\mathbf{A})\Lambda(\mathbf{A})V(\mathbf{A})^\top$$

where

$$\Lambda(\mathbf{A}) = \operatorname{diag}(\lambda_1(\mathbf{A}), \ldots, \lambda_r(\mathbf{A}), 0, \ldots, 0)$$

- For some matrix $\mathbf{A}$, we use $\mathbf{A}_d$ to be the first d columns of $\mathbf{A}$.

PRINCIPAL COMPONENTS ANALYSIS

- For $d \leq r$, PCA gives a projection that minimizes the squared error distance between the data and the projection

- Statistical quantities of interest involve $\mathbf{U}_d$, $\mathbf{V}_d$, and Λ_d

EXAMPLE

- The first d principal components or weights are $\mathbf{V}_d$.
- The best (linear) d-dimensional embedding or principal coordinates is $\widehat{\mathbf{X}}_d = \mathbf{U}\Lambda\mathbf{V}^\top\mathbf{V}_d = \mathbf{U}_d\Lambda_d$

Principal Components Analysis

- For $d \leq r$, PCA gives a projection that minimizes the squared error distance between the data and the projection
- Can do PCA on the covariance

$$S = \frac{1}{n-p}\mathbf{X}^\top\mathbf{X} = \mathbf{V}\frac{1}{n-p}\Lambda^2\mathbf{V}^\top$$

- Can get $\mathbf{U}_d$ from the outer product

$$Q = \mathbf{X}\mathbf{X}^\top = \mathbf{U}\Lambda^2\mathbf{U}^\top$$

- Numerical properties are worse
- `R` provides the functions `prcomp` and `princomp`

Other embeddings

- Other methods of dimension reduction use different forms of "covariance" and then take SVD of those
- LLE, Kernel PCA, Diffusion Maps: (pseudo-pseudo-code)

$$\mathbf{X} \rightsquigarrow \mathbb{W} \in \mathbb{R}^{?\times ?}$$
$$\mathbb{W} \to W\Sigma W^\top$$

- I'm going to focus on PCA

A BIG PROBLEM

- PCA requires computing an SVD
- So do most other data reduction methods
- Datasets are large: a recent ImageNet contest had $n \approx 10^6$ and $p \approx 65000$[2]
- That was only about 8% of the data set
- SVD requires $O(np^2 + pn^2)$ computations
- And you need to store the entire matrix in fast memory
- This is bad.

[2] see e.g. Krizhevsky, Sutskever, and Hinton. **NIPS** 2013

A BIG PROBLEM

- PCA requires computing an SVD
- So do most other data reduction methods
- Datasets are large: a recent ImageNet contest had $n \approx 10^6$ and $p \approx 65000$[2]
- That was only about 8% of the data set
- SVD requires $O(np^2 + pn^2)$ computations
- And you need to store the entire matrix in fast memory
- This is bad.

[2] see e.g. Krizhevsky, Sutskever, and Hinton. **NIPS** 2013

A BIG PROBLEM

- PCA requires computing an SVD
- So do most other data reduction methods
- Datasets are large: a recent ImageNet contest had $n \approx 10^6$ and $p \approx 65000$[2]
- That was only about 8% of the data set
- SVD requires $O(np^2 + pn^2)$ computations
- And you need to store the entire matrix in fast memory
- This is bad.

[2] see e.g. Krizhevsky, Sutskever, and Hinton. **NIPS** 2013

Approximate matrix decompositions

Approximations

Can't take the SVD of $\mathbf{X}$

What about approximating it?

Focus on two methods of "approximate SVD"

1. Nyström extension
2. Column sampling

Not our methods.

A QUICK "SKETCH" OF THE INTUITION

- Both methods fall into a larger class
- Suppose we want to approximate $\mathbf{A} \in \mathbb{R}^{q \times q}$
- Assume $\mathbf{A}$ is symmetric and positive semi-definite
- Choose l and form a "sketching" matrix $\Phi \in \mathbb{R}^{q \times l}$
- Then write $\mathbf{A} \approx (\mathbf{A}\Phi)(\Phi^\top \mathbf{A}\Phi)^\dagger (\mathbf{A}\Phi)^\top =: \mathbf{B}$.

Previous work

- This field is developing very quickly along a few directions
- Sparse approximations $\rightarrow$ enable algorithms which leverage the sparsity to compute SVD or relate to matrix completion/collaborative filtering
- Random projection methods $\rightarrow$ capture the "action" of $\mathbf{A}$
- Halko, Martinsson, and Tropp (2009) gives an excellent overview of algorithms and theory
- Theory compares these methods by bounding $\|\mathbf{A} - \mathbf{B}\|$

Matrix sketching

- Different Φ yield different approximations
- For Nyström and column sampling use

$$\Phi = \pi\tau$$

  Where π is a permutation of $\mathbf{I}_q$ and $\tau = \begin{bmatrix}\mathbf{I}_l & \mathbf{0}\end{bmatrix}^\top$.
- We don't actually need to form these matrices, but this is the idea

Intuition (not the algorithm)

- Essentially, let

$$\mathbf{S} = \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{n \times n} \qquad \text{and} \qquad \mathbf{Q} = \mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{p \times p}$$

- Both of these are symmetric, positive semi-definite
- Randomly choose l entries in $\{1, \ldots, n\}$ and $\{1, \ldots, p\}$
- Then partition the matrix so the selected portion is $\mathbf{S}_{11}$ and $\mathbf{Q}_{11}$

$$\mathbf{S} = \mathbf{V}\Lambda^2\mathbf{V}^\top = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} \quad \mathbf{Q} = \mathbf{U}\Lambda^2\mathbf{U}^\top = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{bmatrix}$$

Approximating things we don't care about

If we want to approximate **S** (or **Q**), we have for example

Nyström

$$\mathbf{S} \approx \begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix} \mathbf{S}_{11}^{\dagger} \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \end{bmatrix}$$

Column sampling

$$\mathbf{S} \approx U\left(\begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix}\right) \Lambda\left(\begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix}\right) U\left(\begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix}\right)^{\top}$$

Previous theoretical results have focused on the accuracy of these approximations (and variants) for a fixed computational budget.

Approximating things we don't care about

If we want to approximate **S** (or **Q**), we have for example

Nyström

$$\mathbf{S} \approx \begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix} \mathbf{S}_{11}^{\dagger} \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \end{bmatrix}$$

Column sampling

$$\mathbf{S} \approx U\left(\begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix}\right) \Lambda\left(\begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix}\right) U\left(\begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix}\right)^{\top}$$

We don't care about these approximations. We don't want these matrices.

What we actually want. . .

We really want $\mathbf{U}$, $\mathbf{V}$, and Λ.

Then we can get the principal components, principal coordinates, and the amount of variance explained.

It turns out that there are quite a few ways to use these two methods to get the things we want.

Let

$$L(\mathbf{S}) = \begin{bmatrix} \mathbf{S}_{11} \\ \mathbf{S}_{21} \end{bmatrix} \qquad\qquad L(\mathbf{Q}) = \begin{bmatrix} \mathbf{Q}_{11} \\ \mathbf{Q}_{21} \end{bmatrix}$$

Lots of approximations

After some reasonable algebra…

Quantity of interest	Label	Approximations
$\mathbf{V}$	$\mathbf{V}_{nys}$	$L(\mathbf{S})\mathbf{V}(\mathbf{S}_{11})\Lambda(\mathbf{S}_{11})^{\dagger}$
	$\mathbf{V}_{cs}$	$\mathbf{U}(L(\mathbf{S}))$
$\mathbf{U}$	$\mathbf{U}_{nys}$	$L(\mathbf{Q})\mathbf{V}(\mathbf{Q}_{11})\Lambda(\mathbf{Q}_{11})^{\dagger}$
	$\mathbf{U}_{cs}$	$\mathbf{U}(L(\mathbf{Q}))$
	$\widehat{\mathbf{U}}_{nys}$	$\mathbf{X}\mathbf{V}_{nys}\Lambda_{nys}^{\dagger/2}$
	$\widehat{\mathbf{U}}_{cs}$	$\mathbf{X}\mathbf{V}_{cs}\Lambda_{cs}^{\dagger/2}$
	$\widehat{\mathbf{U}}$	$\mathbf{U}(\mathbf{x}_1)$

Lots of approximations

After some reasonable algebra…

Quantity of interest	Label	Approximations
$\mathbf{V}$	$\mathbf{V}_{nys}$	$L(\mathbf{S})\mathbf{V}(\mathbf{S}_{11})\Lambda(\mathbf{S}_{11})^{\dagger}$
	$\mathbf{V}_{cs}$	$\mathrm{U}(L(\mathbf{S}))$
$\mathbf{U}$	$\mathbf{U}_{nys}$	$L(\mathbf{Q})\mathbf{V}(\mathbf{Q}_{11})\Lambda(\mathbf{Q}_{11})^{\dagger}$
	$\mathbf{U}_{cs}$	$\mathrm{U}(L(\mathbf{Q}))$
	$\widehat{\mathbf{U}}_{nys}$	$\mathbf{X}\mathbf{V}_{nys}\Lambda_{nys}^{\dagger/2}$
	$\widehat{\mathbf{U}}_{cs}$	$\mathbf{X}\mathbf{V}_{cs}\Lambda_{cs}^{\dagger/2}$
	$\widehat{\mathbf{U}}$	$\mathrm{U}(\mathbf{x}_1)$

Algorithm

Algorithm 1: Space-efficient computation of V_{nys}

```
Input: Data $\mathbf{X}$, approximation parameter $l < p$
    1 Form $\mathbf{x}_1$ by randomly selecting $l$ columns of $\mathbf{X}$
    2 Set $\mathbf{S}_{11} = n^{-1}\mathbf{x}_1^\top \mathbf{x}_1$
    3 Compute $\mathbf{S}_{11} = U(\mathbf{S}_{11})\Lambda(\mathbf{S}_{11})V(\mathbf{S}_{11})^\top$
Return: $\mathbf{X}^\top \mathbf{x}_1 V(\mathbf{S}_{11})\Lambda(\mathbf{S}_{11})^\dagger$
```

Algorithm 2: Computationally stable version

```
Input: Data $\mathbf{X}$, approximation parameter $l < p$
    1 Form $\mathbf{x}_1$ by randomly selecting $l$ columns of $\mathbf{X}$
    2 Compute $\mathbf{x}_1 = U(\mathbf{x}_1)\Lambda(\mathbf{x}_1)V(\mathbf{x}_1)^\top$
Return: $\mathbf{X}^\top U(\mathbf{x}_1)\Lambda(\mathbf{x}_1)^\dagger$
```

Lots of approximations

After some reasonable algebra…

Quantity of interest	Label	Approximations
$\mathbf{V}$	$\mathbf{V}_{nys}$	$L(\mathbf{S})\mathbf{V}(\mathbf{S}_{11})\Lambda(\mathbf{S}_{11})^{\dagger}$
	$\mathbf{V}_{cs}$	$\mathbf{U}(L(\mathbf{S}))$
$\mathbf{U}$	$\mathbf{U}_{nys}$	$L(\mathbf{Q})\mathbf{V}(\mathbf{Q}_{11})\Lambda(\mathbf{Q}_{11})^{\dagger}$
	$\mathbf{U}_{cs}$	$\mathbf{U}(L(\mathbf{Q}))$
	$\widehat{\mathbf{U}}_{nys}$	$\mathbf{X}\mathbf{V}_{nys}\Lambda_{nys}^{\dagger/2}$
	$\widehat{\mathbf{U}}_{cs}$	$\mathbf{X}\mathbf{V}_{cs}\Lambda_{cs}^{\dagger/2}$
	$\widehat{\mathbf{U}}$	$\mathbf{U}(\mathbf{x}_1)$

Remember…

These approximations work on PSD matrices

S does not provide information about **U** directly.

$$\begin{aligned}\mathbf{X} &= \mathbf{U}\Lambda\mathbf{V}^\top \\ \mathbf{Q} &= \mathbf{X}\mathbf{X}^\top \\ &= \mathbf{U}\Lambda^2\mathbf{U}^\top \\ &= \begin{bmatrix}\mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22}\end{bmatrix}\end{aligned}$$

Lots of approximations

After some reasonable algebra…

Quantity of interest	Label	Approximations
$\mathbf{V}$	$\mathbf{V}_{nys}$	$L(\mathbf{S})\mathbf{V}(\mathbf{S}_{11})\Lambda(\mathbf{S}_{11})^{\dagger}$
	$\mathbf{V}_{cs}$	$\mathrm{U}(L(\mathbf{S}))$
$\mathbf{U}$	$\mathbf{U}_{nys}$	$L(\mathbf{Q})\mathbf{V}(\mathbf{Q}_{11})\Lambda(\mathbf{Q}_{11})^{\dagger}$
	$\mathbf{U}_{cs}$	$\mathrm{U}(L(\mathbf{Q}))$
	$\widehat{\mathbf{U}}_{nys}$	$\mathbf{X}\mathbf{V}_{nys}\Lambda_{nys}^{\dagger/2}$
	$\widehat{\mathbf{U}}_{cs}$	$\mathbf{X}\mathbf{V}_{cs}\Lambda_{cs}^{\dagger/2}$
	$\widehat{\mathbf{U}}$	$\mathrm{U}(\mathbf{x}_1)$

Sampling observations

Q is $n \times n$.

Approximations using **Q** mean sampling the rows of **X**.

That is, throwing away observations.

This seems odd. Usually imagine the features may be redundant, but not often the observations.[[There are some exceptions to this…]]

Lots of approximations

After some reasonable algebra…

Quantity of interest	Label	Approximations
$\mathbf{V}$	$\mathbf{V}_{nys}$	$L(\mathbf{S})\mathbf{V}(\mathbf{S}_{11})\Lambda(\mathbf{S}_{11})^{\dagger}$
	$\mathbf{V}_{cs}$	$\mathbf{U}(L(\mathbf{S}))$
$\mathbf{U}$	$\mathbf{U}_{nys}$	$L(\mathbf{Q})\mathbf{V}(\mathbf{Q}_{11})\Lambda(\mathbf{Q}_{11})^{\dagger}$
	$\mathbf{U}_{cs}$	$\mathbf{U}(L(\mathbf{Q}))$
	$\widehat{\mathbf{U}}_{nys}$	$\mathbf{X}\mathbf{V}_{nys}\Lambda_{nys}^{\dagger/2}$
	$\widehat{\mathbf{U}}_{cs}$	$\mathbf{X}\mathbf{V}_{cs}\Lambda_{cs}^{\dagger/2}$
	$\widehat{\mathbf{U}}$	$\mathbf{U}(\mathbf{x}_1)$

Lots of approximations

After some reasonable algebra…

Quantity of interest	Label	Approximations
$\mathbf{V}$	$\mathbf{V}_{nys}$	$L(\mathbf{S})\mathbf{V}(\mathbf{S}_{11})\Lambda(\mathbf{S}_{11})^{\dagger}$
	$\mathbf{V}_{cs}$	$\mathbf{U}(L(\mathbf{S}))$
$\mathbf{U}$	$\mathbf{U}_{nys}$	$L(\mathbf{Q})\mathbf{V}(\mathbf{Q}_{11})\Lambda(\mathbf{Q}_{11})^{\dagger}$
	$\mathbf{U}_{cs}$	$\mathbf{U}(L(\mathbf{Q}))$
	$\widehat{\mathbf{U}}_{nys}$	$\mathbf{X}\mathbf{V}_{nys}\Lambda_{nys}^{\dagger/2}$
	$\widehat{\mathbf{U}}_{cs}$	$\mathbf{X}\mathbf{V}_{cs}\Lambda_{cs}^{\dagger/2}$
	$\widehat{\mathbf{U}}$	$\mathbf{U}(\mathbf{x}_1)$

NOTES OF INTEREST

	Complexity:	
Method	Computational	Storage
Standard	$O(n^2 p + pn^2)$	$O(np)$
Nyström	$O(nl^2 + l^3)$	$O(l^2)$ [$O(nl)$]
Column sampling	$O(lnp + p^2 l)$	$O(pl)$

- Column sampling results in orthogonal **U** and **V**, Nyström doesn't
- $\widehat{\mathbf{U}} = U(\mathbf{x}_1)$ where $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2\end{bmatrix}$ seems reasonable if we think that only the first l selected covariates matter (gets used in supervised PCA)[2]

[2] Bair, Hastie, Paul, and Tibshirani, JASA, 2006.

Results

So what do we use?

Well…

It depends. We did some theory which gives a way to calculate how far off your approximation might be. You could use these bounds if you like to use your data and make a choice.

Did some simulations too.

Simulations

- Draw $\widetilde{X}_i \overset{iid}{\sim} \mathrm{N}_p(0, \Sigma)$ for $n = 5000,\ p = 3000$.

- 4 conditions: $\texttt{Random}_{0.001}$, $\texttt{Random}_{0.01}$, $\texttt{Random}_{0.1}$, and `Band`.

  - $\texttt{Random}_x$ — with probability x and for $i < j$, $\Sigma_{ij}^{-1} = 1$ and 0 otherwise. Diagonal elements are always equal to 1. And symmetrize.

  - `Band` — $\Sigma_{ij}^{-1} = 1$ if $|i - j| \leq 50$ and 0 otherwise.

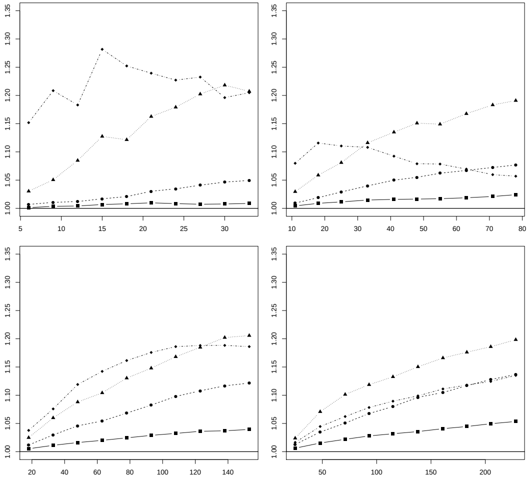
- The graphical models have approximately $\binom{p}{2} \cdot x$ edges for $\texttt{Random}_x$ and exactly $25(2p - 1 - 50)$ edges for `Band`.

What we report

In the **V** case

$$\frac{||\mathbf{V}_{nys,d}(\mathbf{V}_{nys,d}{}^{\top}\mathbf{V}_{nys,d})^{-1}\mathbf{V}_{nys,d}{}^{\top} - \mathbf{V}_{d}\mathbf{V}_{d}{}^{\top}||_F}{||\mathbf{V}_{cs,d}\mathbf{V}_{cs,d}{}^{\top} - \mathbf{V}_{d}\mathbf{V}_{d}{}^{\top}||_F}.$$

Approximations to **V**



- y-axis, relative performance Nyström to column sampling.
- $< 1 \rightarrow$ better; $> 1 \rightarrow$ worse.
- $d \in \{2,\ 5,\ 10,\ 15\}$.
- x-axis, approximation parameter $l \in [3d/2,\ 15d]$
- $\texttt{Random}_{0.001}$ (solid, square)
  $\texttt{Random}_{0.01}$ (dashed, circle)
  $\texttt{Random}_{0.1}$ (dotted, triangle)
  `Band` (dot-dash, diamond)
- d small, similar time
  d large, Nyström is 10–15%
- `Band` theory implies parabolic in l

What we report

In the **U** case

$$\frac{\text{Projection error of estimator}}{\text{Projection error of } \mathbf{U}_{cs,d}}$$

Recall:

$$\begin{aligned}
\mathbf{U}_{nys} &= L(\mathbf{Q})\mathbf{V}(\mathbf{Q}_{11})\Lambda(\mathbf{Q}_{11})^{\dagger} \\
\mathbf{U}_{cs} &= \mathbf{U}(L(\mathbf{Q})) \\
\widehat{\mathbf{U}}_{nys} &= \mathbf{X}\mathbf{V}_{nys}\Lambda_{nys}^{\dagger/2} \\
\widehat{\mathbf{U}}_{cs} &= \mathbf{X}\mathbf{V}_{cs}\Lambda_{cs}^{\dagger/2} \\
\widehat{\mathbf{U}} &= \mathbf{U}(\mathbf{x}_1)
\end{aligned}$$

WHAT WE REPORT

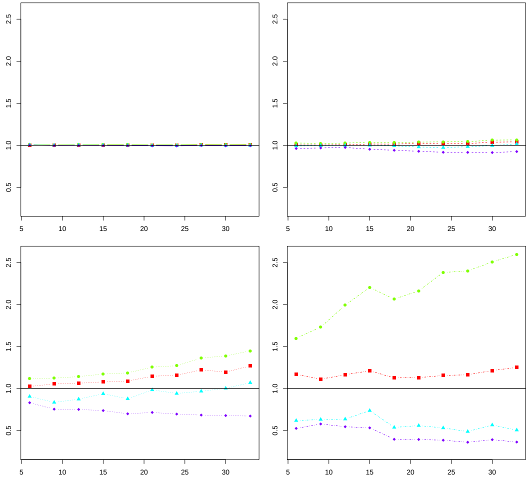
In the **U** case

$$\frac{\text{Projection error of estimator}}{\text{Projection error of } \widehat{\mathbf{U}}_{cs,d}}$$

Recall:

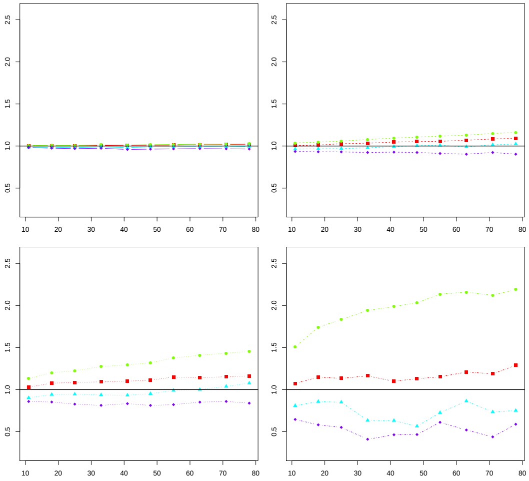
$$\begin{aligned}
\mathbf{U}_{nys} &= L(\mathbf{Q})\mathbf{V}(\mathbf{Q}_{11})\Lambda(\mathbf{Q}_{11})^{\dagger} \\
\mathbf{U}_{cs} &= \mathbf{U}(L(\mathbf{Q})) \\
\widehat{\mathbf{U}}_{nys} &= \mathbf{X}\mathbf{V}_{nys}\Lambda_{nys}^{\dagger/2} \\
\widehat{\mathbf{U}}_{cs} &= \mathbf{X}\mathbf{V}_{cs}\Lambda_{cs}^{\dagger/2} \\
\widehat{\mathbf{U}} &= \mathbf{U}(\mathbf{x}_1)
\end{aligned}$$

APPROXIMATIONS TO $\mathbf{U}$



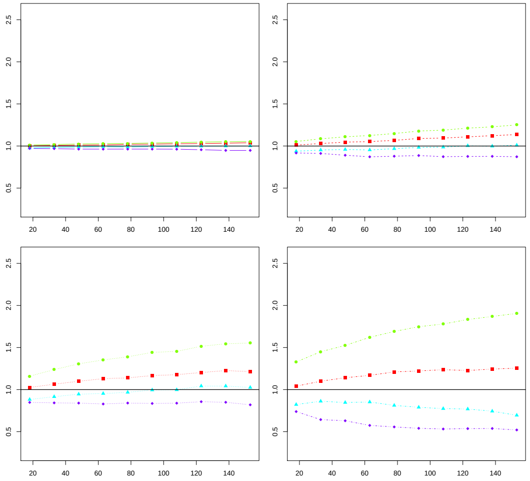
- y-axis, performance relative to $\mathbf{U}_{cs}$
- x-axis, approximation parameter l
- $\mathbf{U}_{nys}$, $\widehat{\mathbf{U}}_{nys}$, $\widehat{\mathbf{U}}_{cs}$, $\widehat{\mathbf{U}}$
- Random$_{0.001}$
  Random$_{0.01}$
  Random$_{0.1}$
  Band
- $d = 2$

APPROXIMATIONS TO $\mathbf{U}$



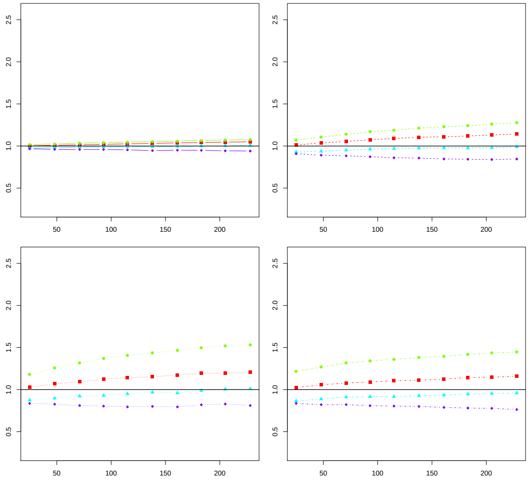
- y-axis, performance relative to $\mathbf{U}_{cs}$
- x-axis, approximation parameter l
- $\mathbf{U}_{nys}$, $\widehat{\mathbf{U}}_{nys}$, $\widehat{\mathbf{U}}_{cs}$, $\widehat{\mathbf{U}}$
- $\texttt{Random}_{0.001}$
  $\texttt{Random}_{0.01}$
  $\texttt{Random}_{0.1}$
  `Band`
- $d = 5$

APPROXIMATIONS TO $\mathbf{U}$



- y-axis, performance relative to $\mathbf{U}_{cs}$
- x-axis, approximation parameter l
- $\mathbf{U}_{nys}$, $\widehat{\mathbf{U}}_{nys}$, $\widehat{\mathbf{U}}_{cs}$, $\widehat{\mathbf{U}}$
- $\texttt{Random}_{0.001}$
  $\texttt{Random}_{0.01}$
  $\texttt{Random}_{0.1}$
  $\texttt{Band}$
- $d = 10$

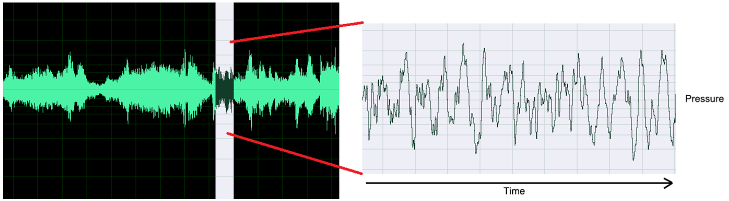
APPROXIMATIONS TO $\mathbf{U}$



- y-axis, performance relative to $\mathbf{U}_{cs}$
- x-axis, approximation parameter l
- $\mathbf{U}_{nys}$, $\widehat{\mathbf{U}}_{nys}$, $\widehat{\mathbf{U}}_{cs}$, $\widehat{\mathbf{U}}$
- $\texttt{Random}_{0.001}$
  $\texttt{Random}_{0.01}$
  $\texttt{Random}_{0.1}$
  `Band`
- $d = 15$

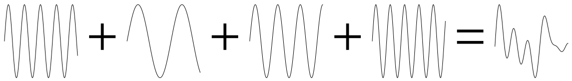
The fun part

AUDIO



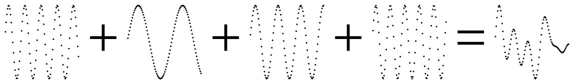
- Sound is created by moving objects, like a vibrating tuning fork
- The motion of the tuning fork creates changes in air pressure
- This time-varying air pressure propagates through the air in all directions
- If it reaches some other object (like your eardrum) it will cause that object to move in the same way
- Thus, we can think of audible sound as time-varying pressure

Sines and Cosines



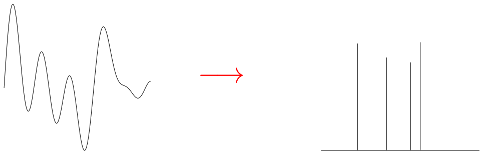
- We can represent any function using an infinite sum of basis functions.
- In particular, for sound, we use ‘sine’ or ‘cosine’
- One reason is that pure tones are single sine waves
- If you ever tuned to A=440, that’s $\sin(2\pi \times 440t)$.
- So, if we decompose audio into sines at different frequencies and amplitudes, then we know how loud each individual pitch is

SINES AND COSINES



- We don’t get continuous curves
- Rather a digital recording device ‘samples’ at a particular rate
- Typical audio (uncompressed) is 44.1 kHz, telephone is ∼8 kHz
- Can’t differentiate frequencies above sample rate/2
- Human ear goes up to about 20 kHz, so 44.1 is “enough”

FFT



- Use FFT to transform audio to frequencies/amplitudes
- If sampled at n points, get $n/2$ frequencies
- If an instrument plays three ascending notes or three descending notes at the same speed $\rightarrow$ same FFT
- Solution: perform many ‘windowed’ FFTs, called STFT
- Actually, we do this with overlap

Our little project

Suppose we apply some sort of “filter” to a recording.

Can we use the remaining information combined with other recordings to recover what we removed?

The application here is “de-soloing”, removing one instrument or voice from a recording.

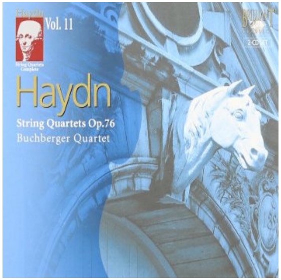
The problem is that if you delete bins in the STFT, you also remove information about the remaining instruments.

This is a pilot exercise toward the “de-soloing” problem.

Pilot exercise

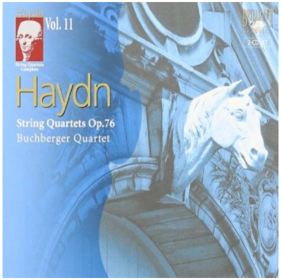
Can I reconstruct the first movement of Haydn's Op. 76 String Quartet No. 4?

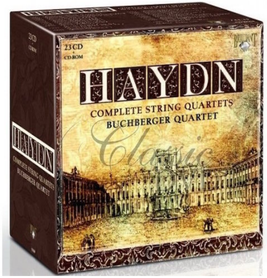
- Each frame is 4096 samples with a 50/50 overlap
- That's about .1 seconds long
- Database of all Complete Haydn Quartets
- Buchberger Quartet
- Test is 3000 frames (about 1 minute)
- Database has 176,000 frames



PROCEDURE

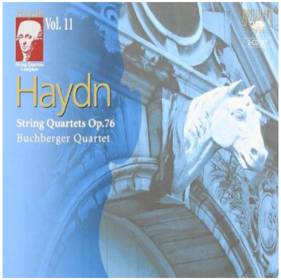
1. The database has $n = 176000$ frames, $p = 4096$ frequencies (at which the amplitudes are measured)
2. Perform PCA on the database
3. Reduce to 100 dimensions (95% explained variation)
4. Express target frames in same coordinates
5. Find 40 nearest neighbors in Database
6. Reconstruction is weighted average of the neighbors

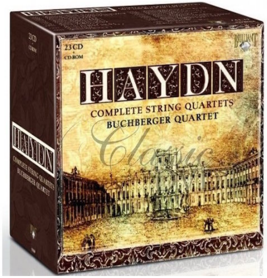




PROCEDURE

1. The database has $n = 176000$ frames, $p = 4096$ frequencies (at which the amplitudes are measured)
2. Perform PCA on the database
3. Reduce to 100 dimensions (95% explained variation)
4. Express target frames in same coordinates
5. Find 40 nearest neighbors in Database
6. Reconstruction is weighted average of the neighbors





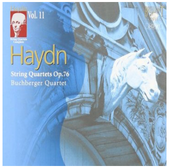
Experiment

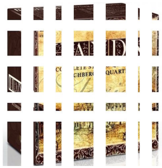
Here, we can compute the SVD on the database.

For real applications, we won't be able to.

Compare the reconstruction with PCA on the full dataset to that using $\widehat{\mathbf{U}}_{cs}$ where we use only 500 (rather than 4096) random columns

Measures of the reconstruction error show only slight differences.





Conclusions

- For computing $\mathbf{V}$, CS beats Nyström in terms of accuracy, but is much slower for similar choices of the approximation parameter and d large.
- For computing $\mathbf{U}$, the naïve methods are bad, better to approximate $\mathbf{V}$ and multiply, so see above
- $\widehat{\mathbf{U}}$ really stinks. But it's most obvious. This also gets used for supervised PCA (stringent assumptions make it work in their paper)
- Other choices of Φ
- Thanks NSF for funding us.
- For more info, see the paper. Contains boring theory, more extensive simulations, and application to Enron email dataset.