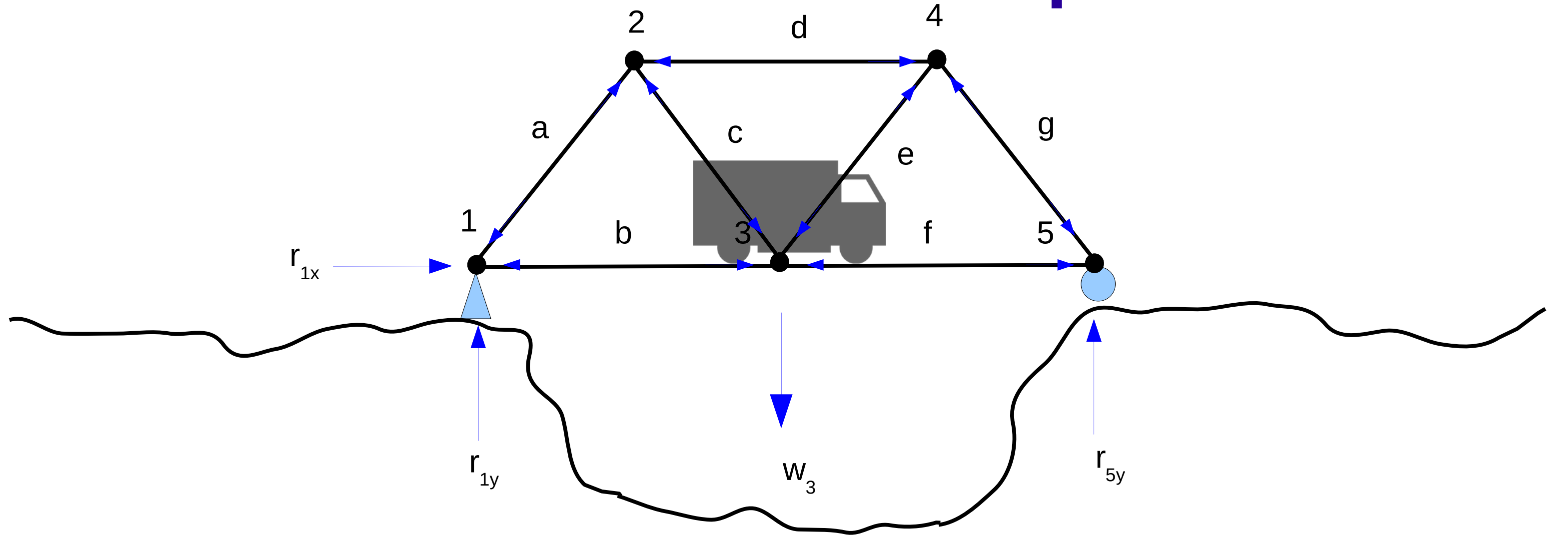
Homework: Truss problem



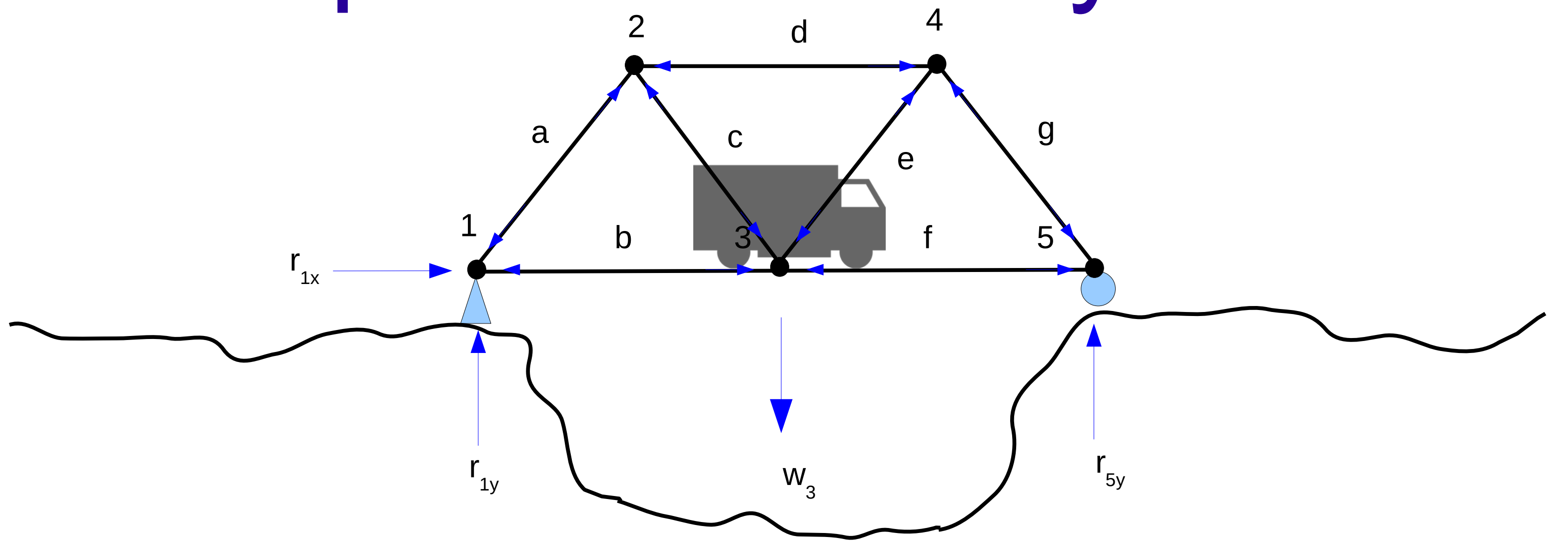
- In equilibrium, sum of forces at each joint is zero.
- Must resolve forces into x and y (horiz and vert).
- Three “reaction” forces act on whole bridge.
- Blue arrows indicate forces pulling on ends of beams.
- Equilibrium equation relates known, external forces to unknown, internal forces.

Equilibrium equation – sparse 10x10

$$
\begin{array}{cc}
 & \begin{array}{cccccccccc} \text{fa} & \text{fb} & \text{fc} & \text{fd} & \text{fe} & \text{ff} & \text{fg} & \text{r1x} & \text{r1y} & \text{r5y} \end{array} \\
\begin{array}{c} \text{1x} \\ \text{1y} \\ \text{2x} \\ \text{2y} \\ \text{3x} \\ \text{3y} \\ \text{4x} \\ \text{4y} \\ \text{5x} \\ \text{5y} \end{array} &
\begin{bmatrix}
-c & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
-s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
c & 0 & -c & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
s & 0 & s & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & c & 0 & -c & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & -s & 0 & -s & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & c & 0 & -c & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & s & 0 & s & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & c & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -s & 0 & 0 & 1
\end{bmatrix}
\end{array}
\begin{bmatrix} f_a \\ f_b \\ f_c \\ f_d \\ f_e \\ f_f \\ f_g \\ r_{1x} \\ r_{1y} \\ r_{5y} \end{bmatrix}
= -
\begin{array}{cc}
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -15 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} &
\begin{array}{c} \text{1x} \\ \text{1y} \\ \text{2x} \\ \text{2y} \\ \text{3x} \\ \text{3y} \\ \text{4x} \\ \text{4y} \\ \text{5x} \\ \text{5y} \end{array}
\end{array}
$$

$$A f_{internal} = -w_{external}$$

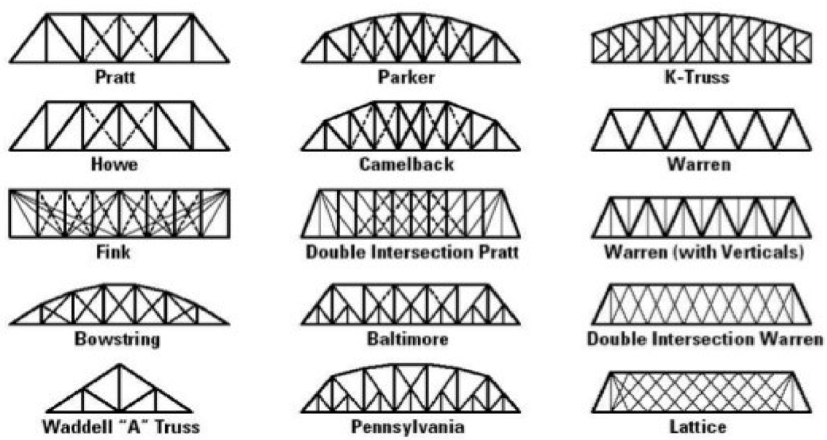
Truss problem: sanity checks

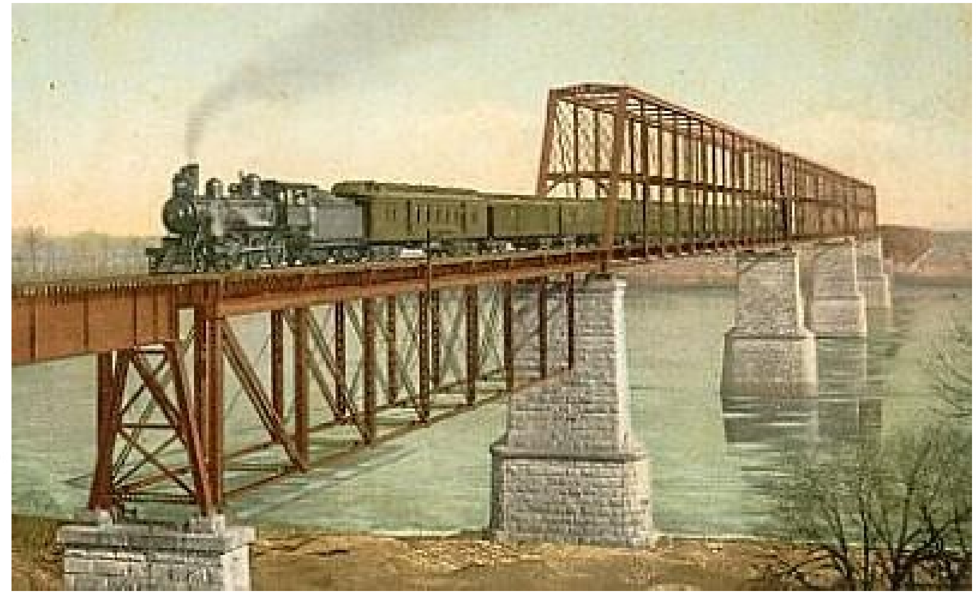


- Vertical reaction forces should sum to weight of bridge + truck.
- Horizontal reaction force should be zero.
- Simulate bridge sag in your head. Do floor members stretch our or compress when truck weight is added?
- Think about symmetry of problem. Are your computed forces symmetric?

Tobin Bridge

BRIDGE TRUSS TYPE





The SVD: Applications

Fun video about computing SVD

- Video from 1975:

  https://www.youtube.com/watch?v=R9UoFyqJca8

Summary: Triangle of concepts

SVD

$$A = U \Sigma V^T$$

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 & \\ & \sigma_2 & & \dots \\ 0 & & \sigma_3 & \end{pmatrix}$$

Matrix norm (induced norm)

$$\|A\| = max\left(\frac{\|A x\|}{\|x\|} : x \in K^n\right)$$

$$= \sigma_{max}$$

Matrix condition number

$$k = \frac{\sigma_{max}}{\sigma_{min}}$$

$$= \|A\| \cdot \|A^{-1}\|$$

Singular values and eigenvalues

- Eigenvalue decomposition: Square matrix

$$A = Q \Lambda Q^{-1}$$

$$\boxed{A} = \boxed{Q}\,\boxed{\Lambda}\,\boxed{Q^{-1}}$$

$$\Lambda = \begin{pmatrix} \lambda_1 & & 0 & \\ & \lambda_2 & & \\ 0 & & \lambda_3 & \\ & & & \ddots \end{pmatrix}$$

- Singular value decomposition: Arbitrary rectangular matrix

$$A = U \Sigma V^T$$

$$\overset{n\times p}{\boxed{A}} = \overset{n\times n}{\boxed{U}}\,\overset{n\times p}{\boxed{\Sigma}}\,\overset{p\times p}{\boxed{V^T}}$$

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 & \\ & \sigma_2 & & \cdots \\ 0 & & \sigma_3 & \end{pmatrix}$$

Properties of the SVD

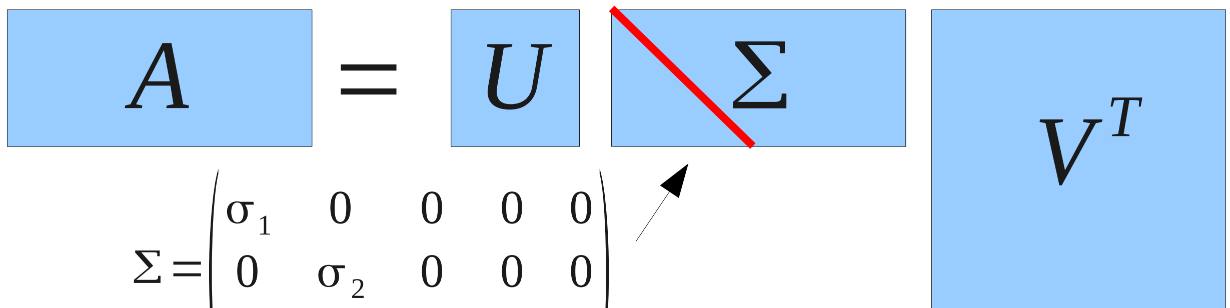
$$\overset{n\times p}{A} = \overset{n\times n}{U}\ \overset{n\times p}{\Sigma}\ \overset{p\times p}{V^T}$$

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & 0 & 0 & 0 & \\ 0 & \sigma_2 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \sigma_3 & 0 & 0 & \end{pmatrix}$$

- *U*, *V* are orthogonal matrices.
- Σ is diagonal matrix. Diagonal elements are the "singular values".
  - By convention, they are written in decreasing order, from largest to smallest.
  - Non diagonal entries are zero.

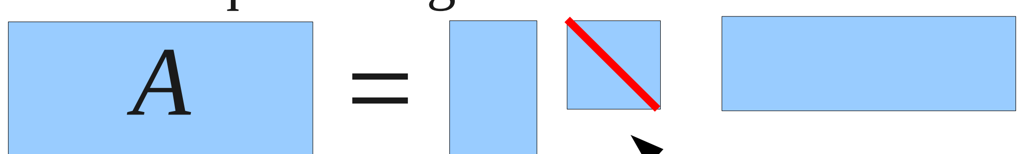
Full vs. reduced SVD

- **Full:** Matlab default
  U, V are square, orthogonal.
  Σ is rectangular. Zero cols (rows) correspond to nullspace of A.



$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & 0 \end{pmatrix}$$

- **Reduced:** Matlab: 'economy'
  U, V are rectangular, columns/rows orthogonal.
  Σ is square diagonal. No zero elements on diagonal.



$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix}$$

Alternate forms for writing SVD

- One can write A as sum of rank-1 matrices

$$A = U \Sigma V^T$$
$$= \sum_{i=1}^{r} u_i \sigma_i v_i^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

Outer product: result is matrix.

- This is just a different way of writing the SVD

$$A = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ u_1 & u_2 & u_3 & u_4 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \end{pmatrix} \begin{pmatrix} \cdots & v_1 & \cdots \\ \cdots & v_2 & \cdots \\ \cdots & v_3 & \cdots \\ \cdots & v_4 & \cdots \end{pmatrix}$$

Aside on vector-vector products

- Assume default vector is column vector.

$$\vec{u} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad \vec{v} = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$$

- "Inner" product (dot product):

$$\vec{u}^T \vec{v} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} = 32$$

Inner product nomenclature

Result is scalar

Outer product

$$\vec{u}\,\vec{v}^T = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 4 & 5 & 6 \end{pmatrix}$$

Outer product nomenclature

$$= \begin{pmatrix} 1\cdot 4 & 1\cdot 5 & 1\cdot 6 \\ 2\cdot 4 & 2\cdot 5 & 2\cdot 6 \\ 3\cdot 4 & 3\cdot 5 & 3\cdot 6 \end{pmatrix}$$

Result is matrix

$$= \begin{pmatrix} 4 & 5 & 6 \\ 8 & 10 & 12 \\ 12 & 15 & 18 \end{pmatrix}$$

Matlab demo

```
>> u = [1;2;3]

u =
     1
     2
     3

>> v = [4;5;6]

v =
     4
     5
     6
```

Inner product

```
>> u'*v

ans =

    32
```

Outer product

```
>> u*v'

ans =

     4     5     6
     8    10    12
    12    15    18
```

SVD as sum of outer products

$$\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix} \begin{pmatrix} * & . & . & . \\ . & * & . & . \\ . & . & * & . \end{pmatrix} \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}$$

$$A \quad = \quad U \quad \Sigma \quad V^T$$

$$= \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix} \left[\begin{pmatrix} \sigma_1 & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix} + \begin{pmatrix} . & . & . & . \\ . & \sigma_2 & . & . \\ . & . & . & . \end{pmatrix} + \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & \sigma_3 & . \end{pmatrix} \right] \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}$$

- Elements which participate in sum are *
- Elements which are zero or don’t participate in sum are .

Consider each singular value

$$\sigma_1 \qquad \begin{array}{cccc} & U & (\sigma_1) & V^T \\ & 3 & 4 & 4 \\ 3 & \begin{pmatrix} * & \cdot & \cdot \\ * & \cdot & \cdot \\ * & \cdot & \cdot \end{pmatrix} & \begin{pmatrix} * & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} & \begin{pmatrix} * & * & * & * \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \end{array} \quad 4$$

$$\begin{pmatrix} * & \cdot & \cdot \\ * & \cdot & \cdot \\ * & \cdot & \cdot \end{pmatrix} \begin{pmatrix} * & * & * & * \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} \qquad = \sigma_1 u_1 v_1^T$$

$$\sigma_2 \begin{pmatrix} \cdot & * & \cdot \\ \cdot & * & \cdot \\ \cdot & * & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & * & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ * & * & * & * \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\begin{pmatrix} \cdot & * & \cdot \\ \cdot & * & \cdot \\ \cdot & * & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ * & * & * & * \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} = \sigma_2 u_2 v_2^T$$

$$\sigma_3 \begin{pmatrix} \cdot & \cdot & * \\ \cdot & \cdot & * \\ \cdot & \cdot & * \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & * & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ * & * & * & * \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\begin{pmatrix} \cdot & \cdot & * \\ \cdot & \cdot & * \\ \cdot & \cdot & * \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ * & * & * & * \end{pmatrix}$$

$$\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} = \sigma_3 u_3 v_3^T$$

SVD as sum of outer products

$$\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix} \begin{pmatrix} * & & & \\ & * & & \\ & & * & \end{pmatrix} \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}$$

$$A \quad = \quad U \quad \Sigma \quad V^T$$

$$= \sum_{i=1}^{r} u_i \sigma_i v_i^T$$

Restatement of SVD

- Using SVD, one can decompose A into sum of rank-1 matrices

$$A = U \Sigma V^T$$

$$= \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ u_1 & u_2 & u_3 & u_4 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \end{pmatrix} \begin{pmatrix} \cdots & v_1 & \cdots \\ \cdots & v_2 & \cdots \\ \cdots & v_3 & \cdots \\ \cdots & v_4 & \cdots \end{pmatrix}$$

$$= \sum_{i=1}^{r} u_i \sigma_i v_i^T$$

Sum to rank r – sum is exact.
This is simple re-write of above decomposition.

Major theorem

Eckart-Young-Mirsky Theorem:

- Start with original matrix: A
- SVD decomposition: $A = \sum_{i=1}^{r} u_i \sigma_i v_i^T$
- Keep only p<r terms in sum: $A_p = \sum_{i=1}^{p} u_i \sigma_i v_i^T$

Theorem: A_p is the "closest" approximation to *A* in the L2 norm (largest singular value) and also the Frobenius norm.

Dimensionality reduction

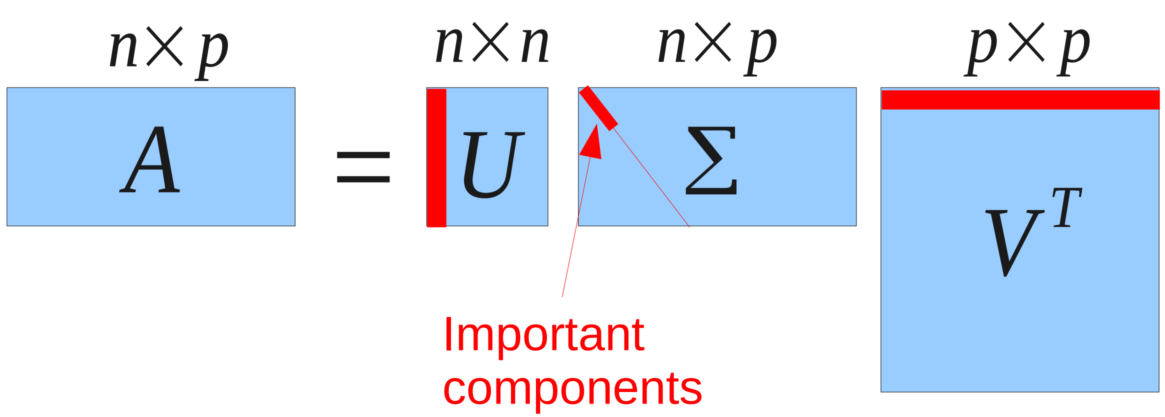
- Recall magnitude of singular values decreases with increasing *i*.....

$$A \approx \sum_{i=1}^{m} \sigma_i u_i v_i^T$$

- Suppose we only sum first few terms?
  - That is, throw away small contributions to the sum.
  - This is a way to get a good approximation to the matrix A.

Written in matrix form...

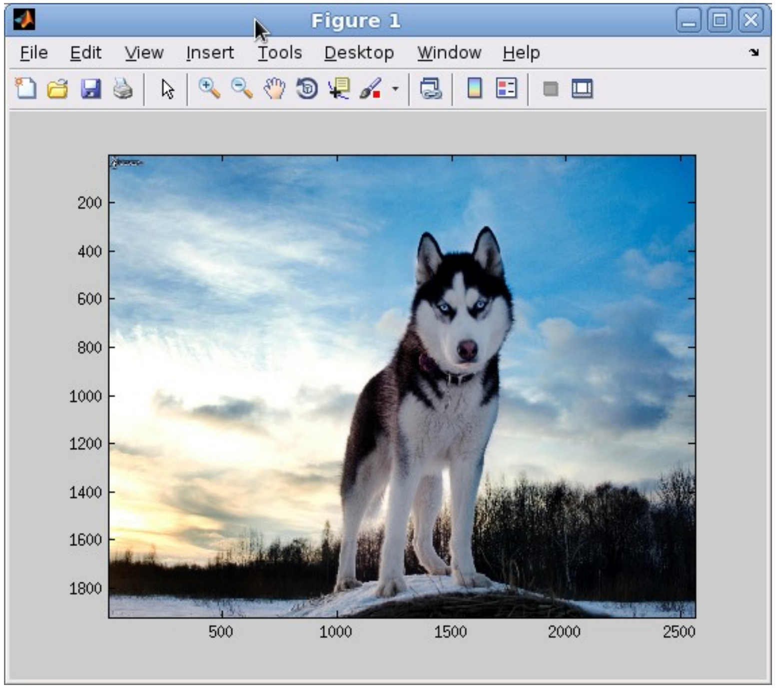
- SVD decomposes image into components (rows, cols) which are more important and those which are less important.
- Magnitude of each singular value is measure of how important each component is. (Recall σ_i are sorted from highest to lowest.)



$$A = \sum_{i=1}^{r} u_i \sigma_i v_i^T$$

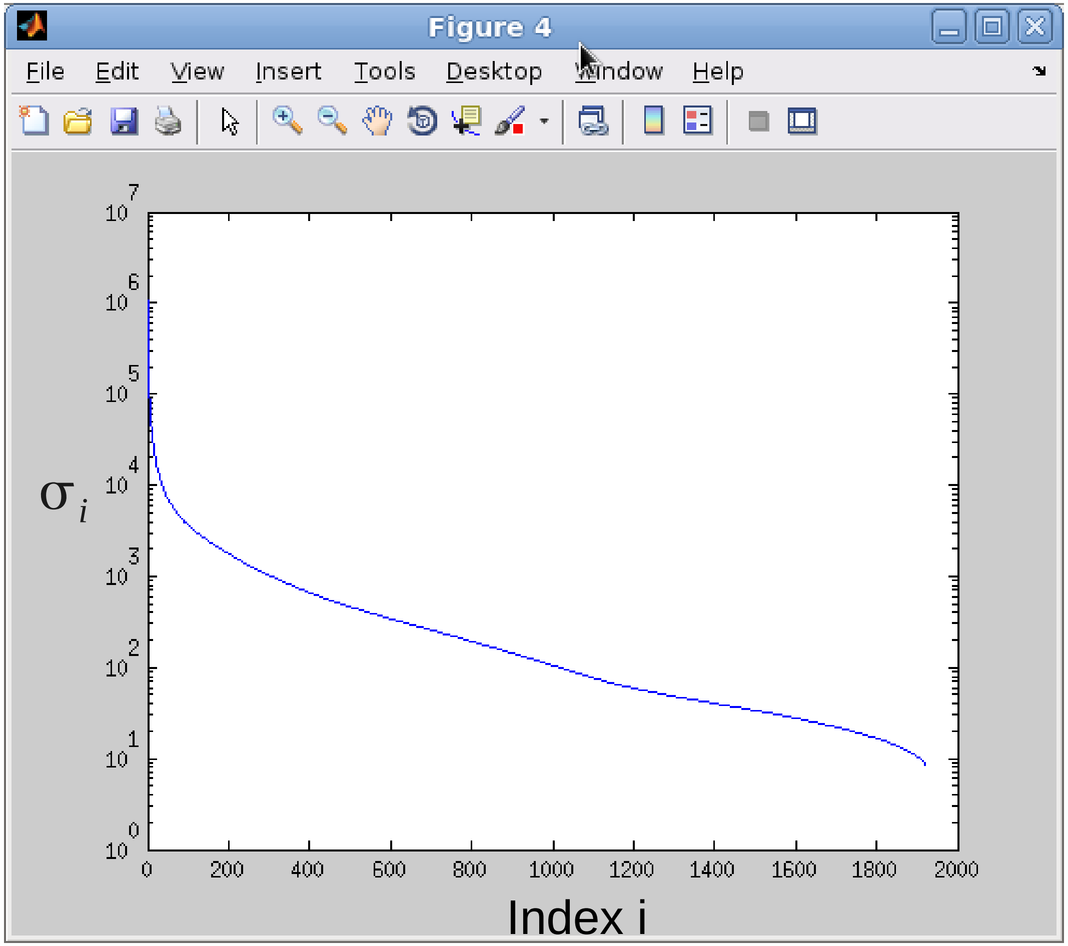
Dimensionality reduction corresponds to truncating sum

Singular values along diagonal



$$A = U \Sigma V^T$$

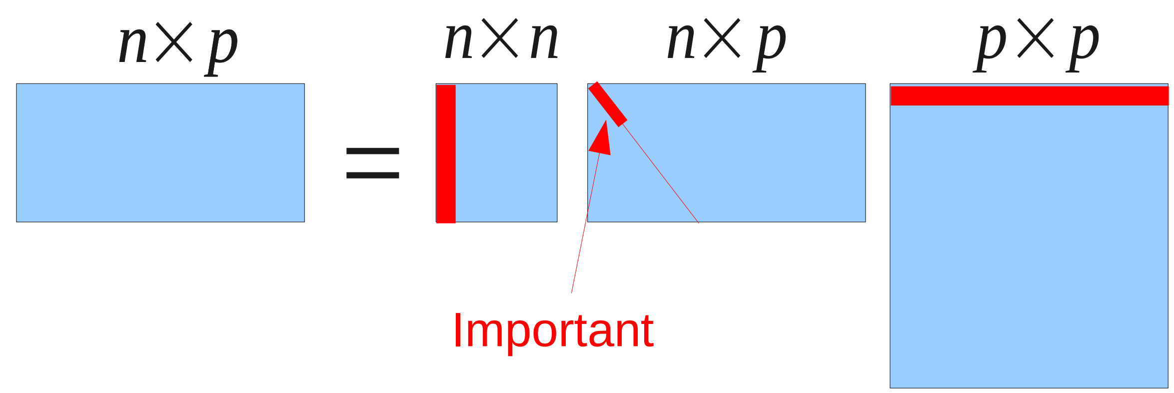
$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 & \\ & \sigma_2 & & \cdots \\ 0 & & \sigma_3 & \end{pmatrix}$$



Singular values from SVD of husky image

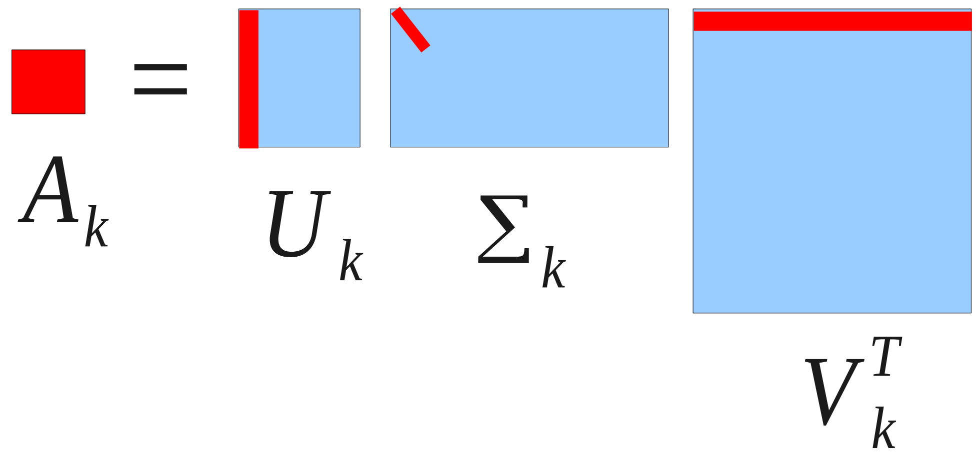
SVD and dimensionality reduction

- Consider an image as a matrix.
- SVD decomposes image into components (rows, cols) which are more important and those which are less important.
- Magnitude of each singular value is measure of how important each component is. (Recall S is sorted from highest to lowest.)

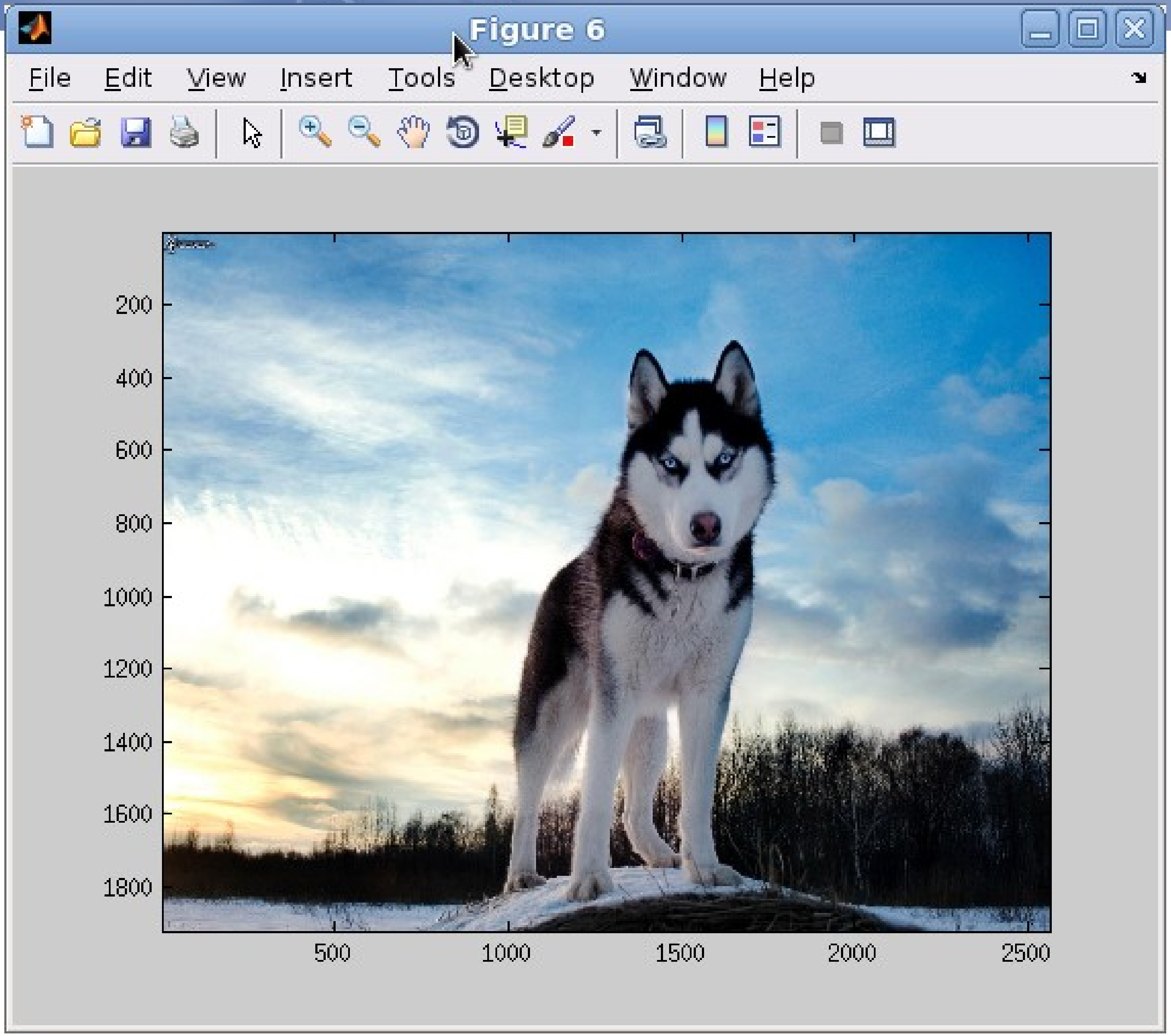


$$A = U \Sigma V^T$$

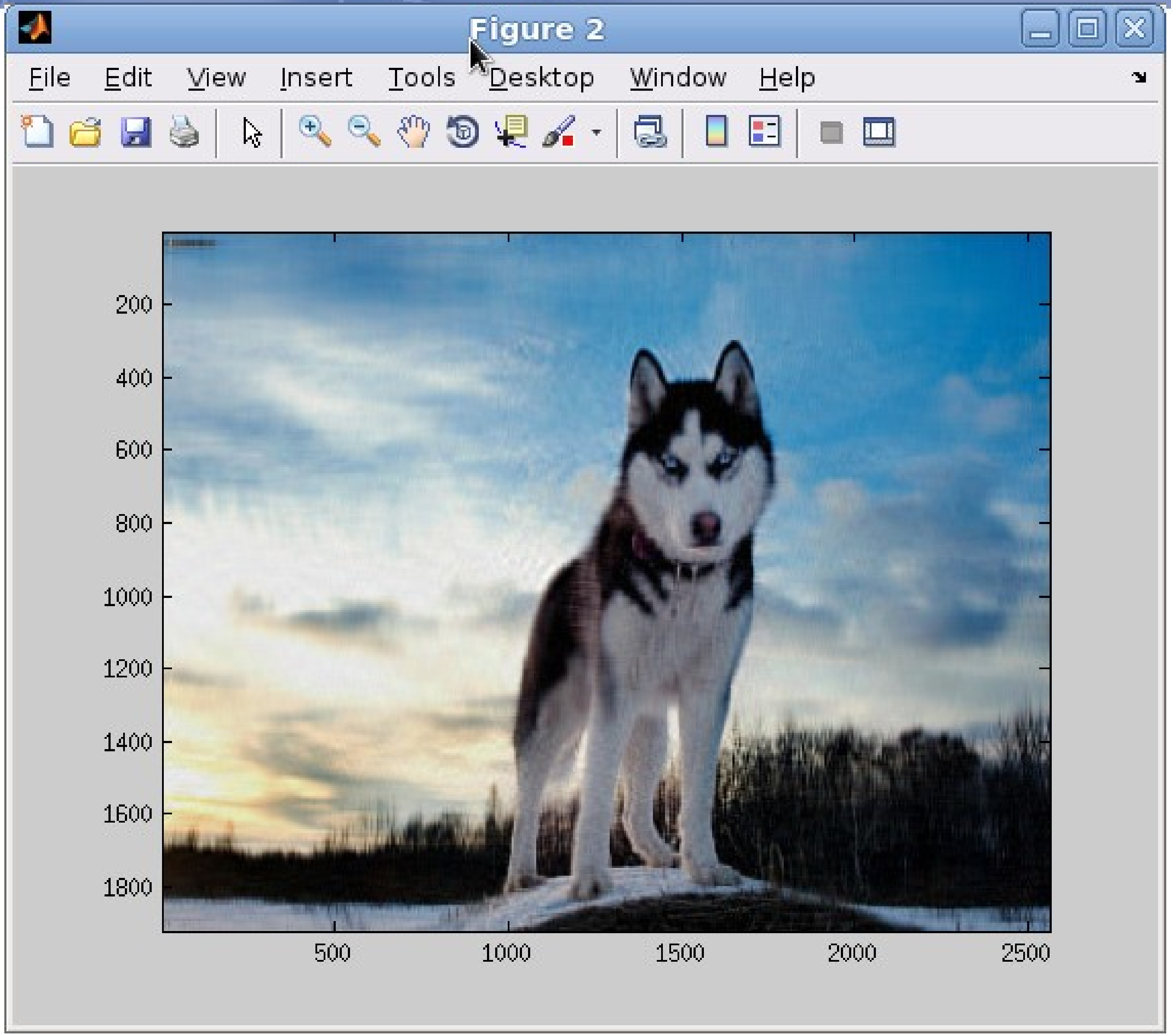
Idea: Compress image by discarding small singular values



- Perform SVD. $A \rightarrow U\,\Sigma\,V^T$
- Discard non-red components. $\Sigma_k = chop(\Sigma)$
- Recreate image by performing multiplication $A_k = U_k \Sigma_k V_k^T$

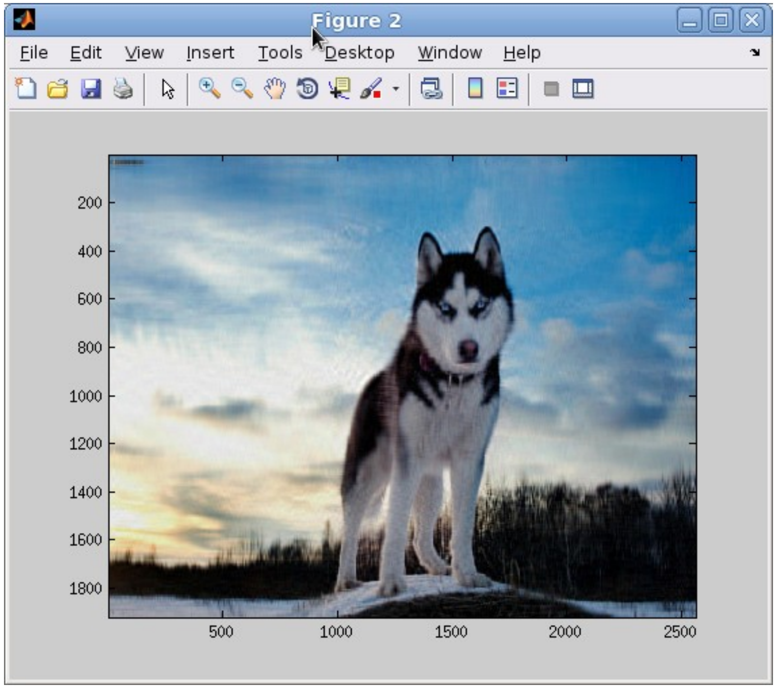


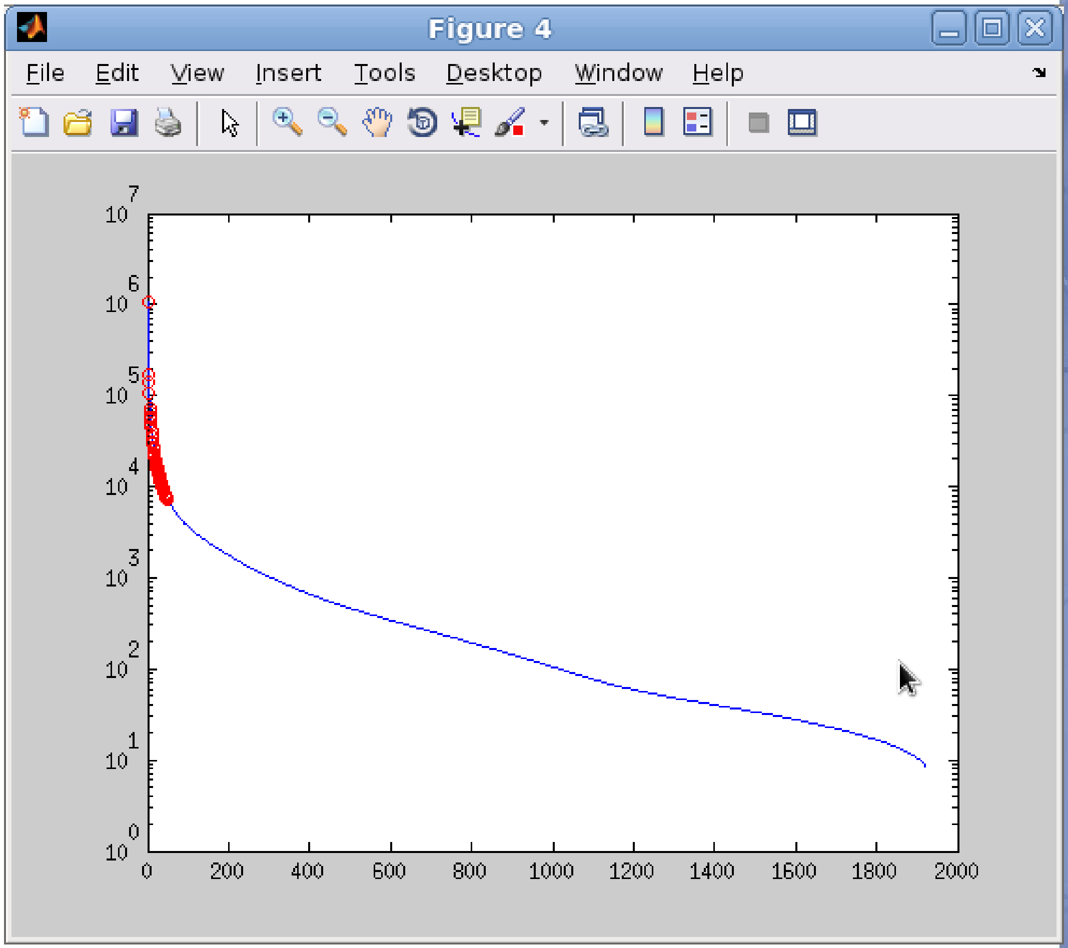
- Original image decomposed and recomposed – keeping all 1920 singular values.

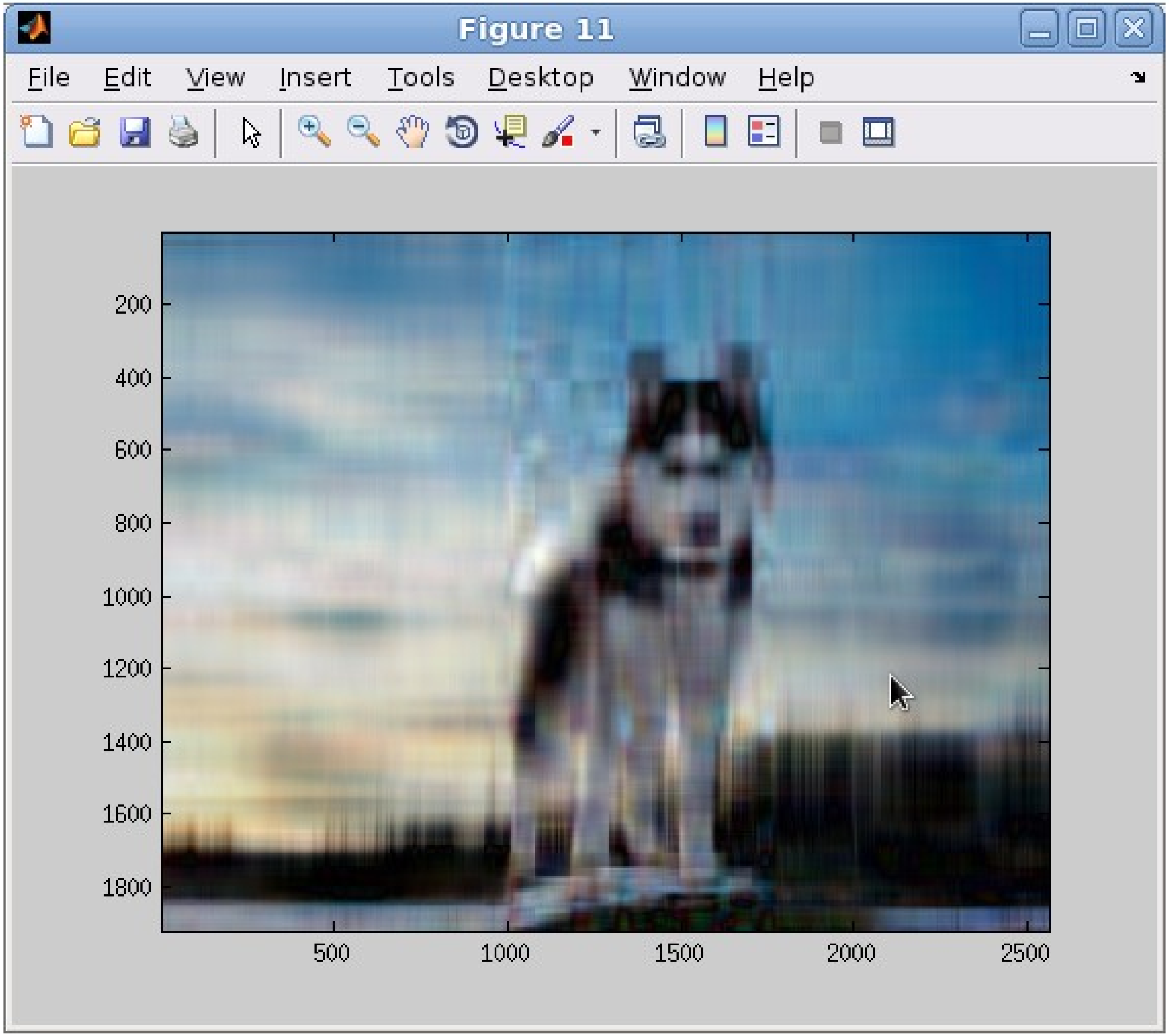


Keep first 50 singular values (zero out remaining 1870).

First 50 singular values

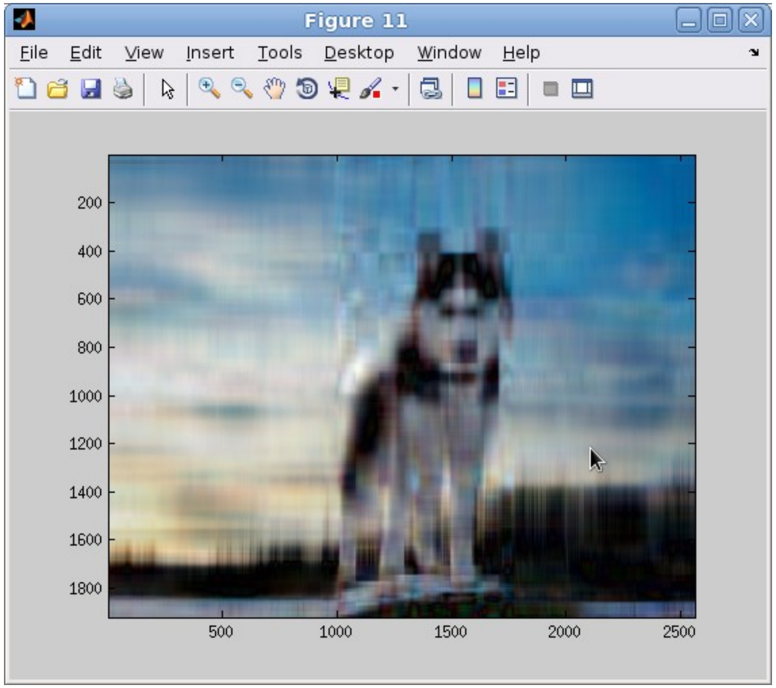


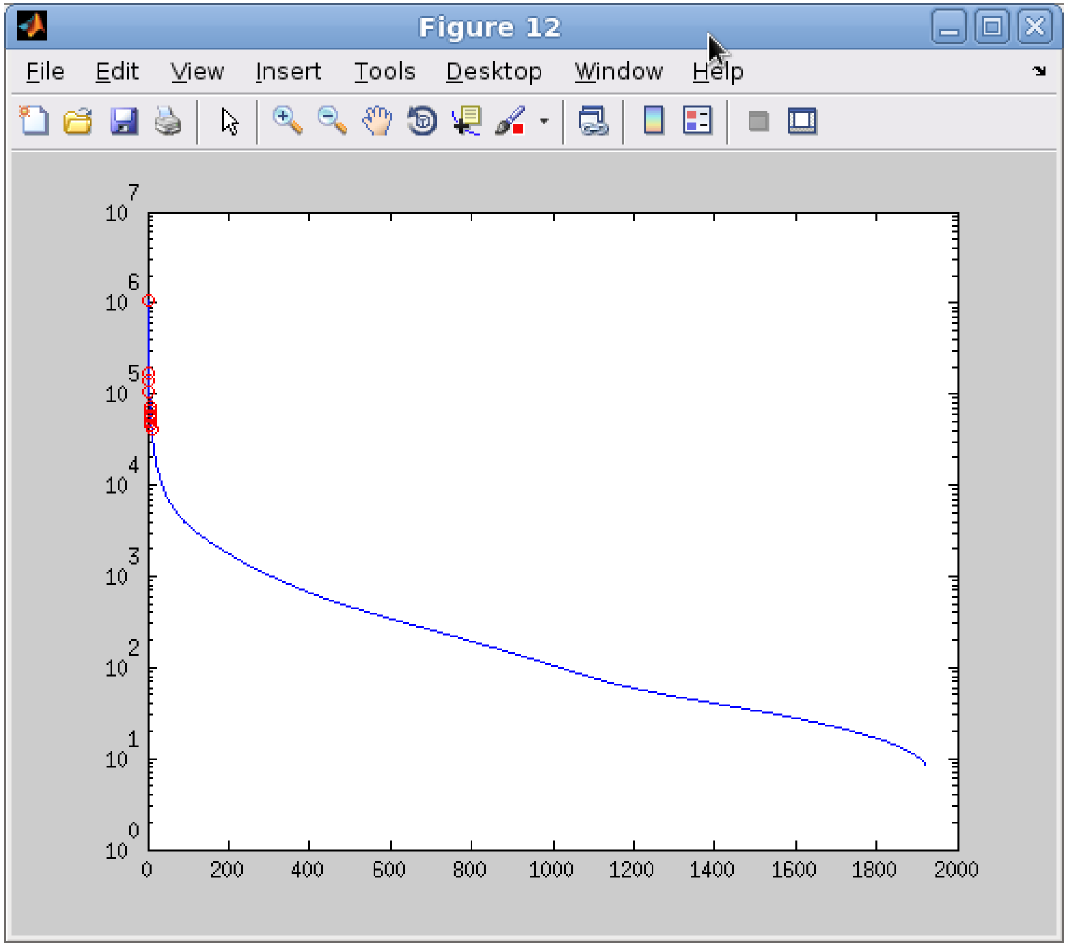




Keep first 10 singular values (zero out remaining 1910).

First 10 singular values





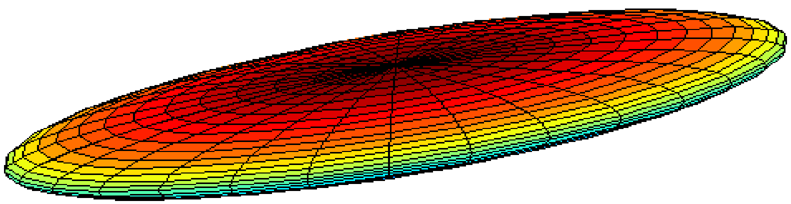
Remarks

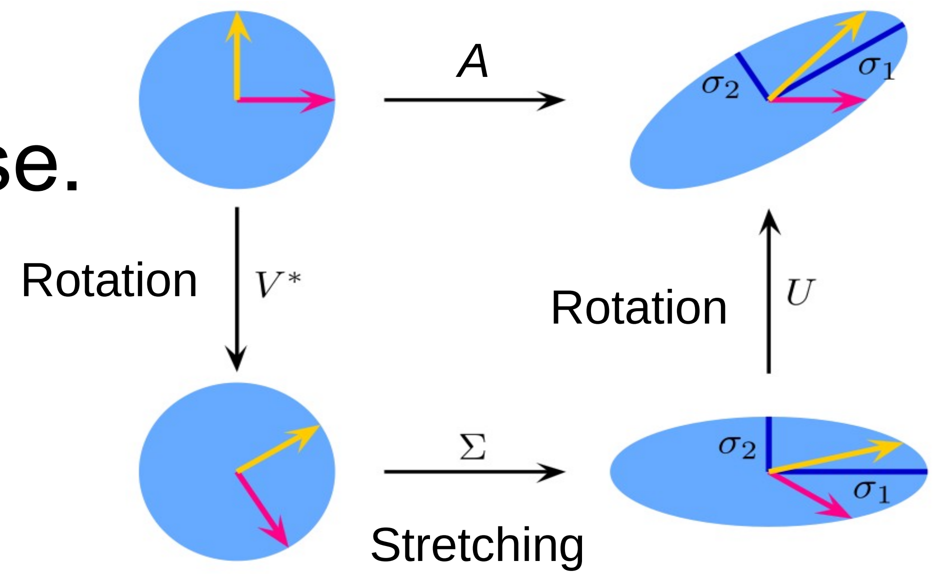
$$A \approx \sum_{i=1}^{m} \sigma_i u_i v_i^T \qquad m \ll rank(A)$$

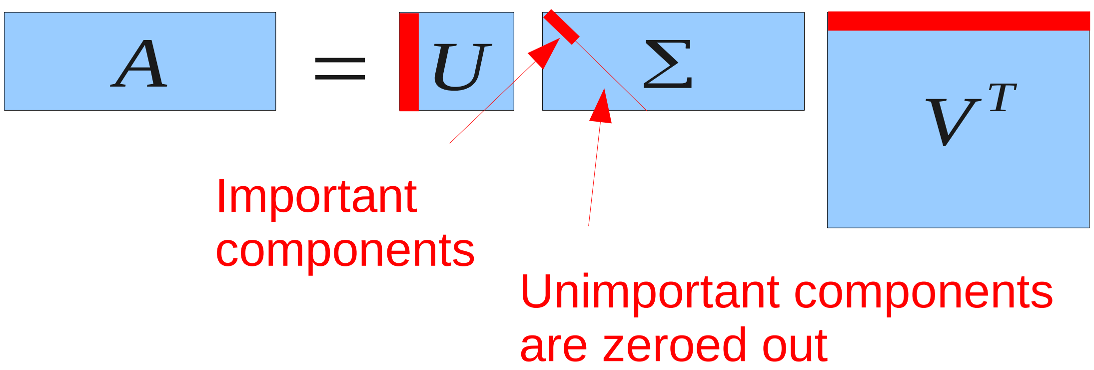
- This is called “dimensionality reduction”.
- This is the essence of LSI = latent semantic indexing (next example).
- Major point: I have created an approximation to matrix A which is close, but requires much less information to reproduce.
  - U, Σ, V much smaller than original *A*.

Dimensionality reduction -- Visualization

- Consider situation in 3D: almost flat ellipse.







- A good approximation is to simply treat the ellipse as flat.

Another application of SVD: Latent semantic indexing

- Algorithm used for information retrieval.
- Consider collection of documents (books, say).

d_1 d_2 d_3 d_4

- Books on same subject share similar words. Books on different subjects use different words.
- However, all books share common words like "the", "and", "or", etc.

Latent Semantic Indexing

- Latent = “hidden”
- Semantic = “meaning”
- Indexing = “analysis”

Problem to solve

- Given a search word, return all relevant books.
- Use this algorithm as a “recommender system”.
  - Example: I input the the phrase “non-Euclidian geometry”, it returns a list of math books about non-Euclidian geometry first, then some related math books next.
  - If I input the word “electrodynamics”, it returns a list of physics books about E&M first, then some related physics books next.
- The recommender system should return a list of books, sorted by their relevance to my search term.

Does this sound like Amazon's “you may also like”?

Word count for each book

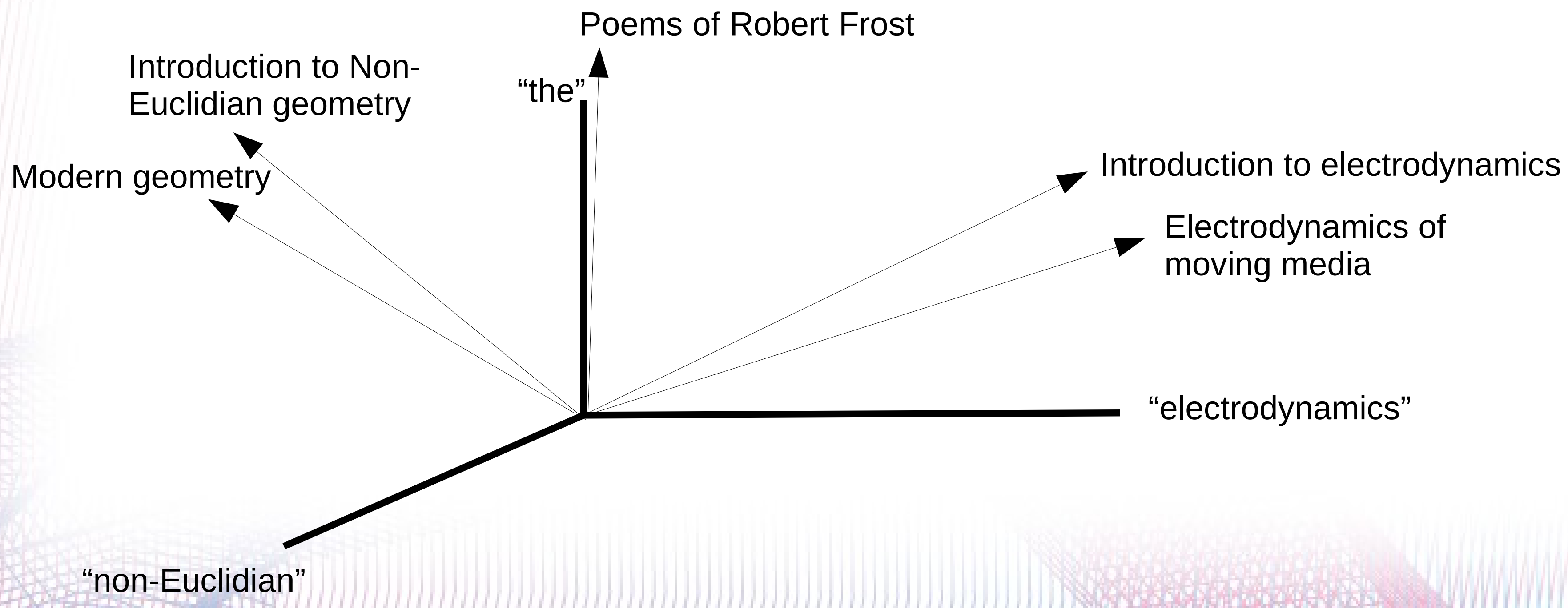
- You can distinguish different books via word count.
- Word count for each book (document) is a column vector.
- This construction is a “term-document” matrix.

Different documents in each column.

	A	B	C	D	E	F	G	H	I	J	K
1	numerical_analysis	1	2	1	1	0	0	0	0	0	0
2	is	1	2	2	1	2	1	2	1	1	2
3	fun	1	1	0	0	1	0	0	0	0	0
4	and	1	0	1	1	0	1	1	1	1	0
5	beautiful	1	0	0	0	0	0	0	1	0	1
6	some	1	0	0	0	0	0	0	0	0	0
7	people	1	0	0	0	0	0	0	0	0	0
8	find	1	1	0	0	0	0	0	0	0	0
9	it	1	1	0	0	0	0	1	1	0	1
10	difficult	1	1	0	0	1	0	0	0	0	0
11	can	0	1	0	0	0	0	0	0	0	0
12	be	0	1	0	0	0	0	0	0	0	0
13	however	0	1	0	0	0	0	0	0	0	0
14	important	0	1	0	1	0	0	2	0	0	0
15	you	0	1	0	0	0	0	0	0	0	0
16	may	0	1	0	0	0	0	0	0	0	0
17	that	0	1	0	0	0	0	0	0	0	0
18	also	0	1	0	0	0	0	1	0	0	0
19	a	0	0	1	0	1	0	0	1	0	1
20	branch	0	0	1	0	1	0	0	0	0	0

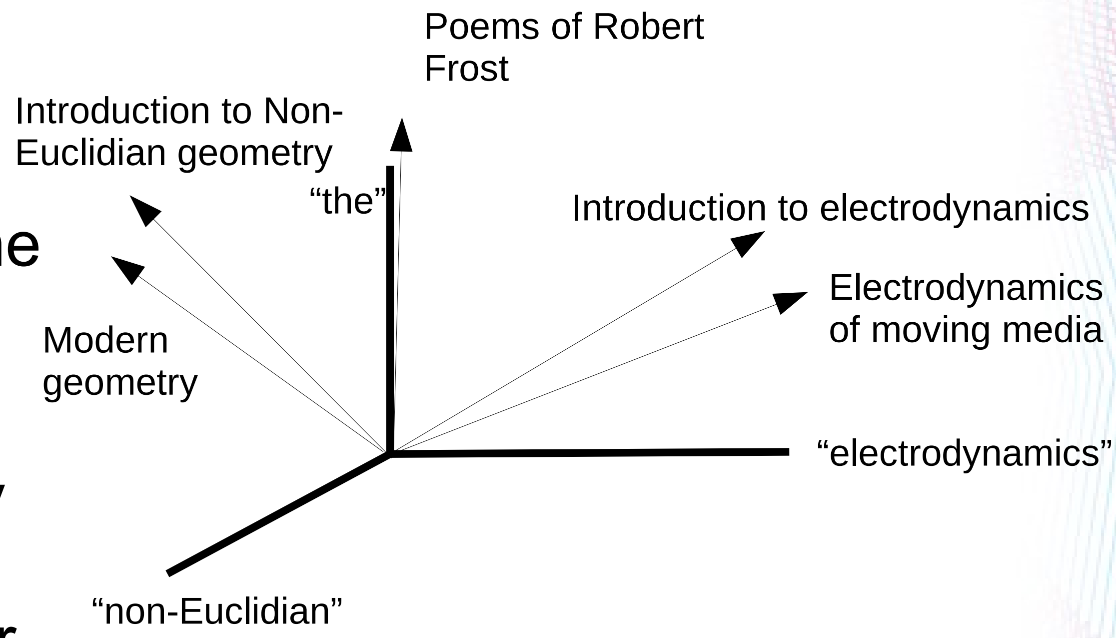
Vector space of documents

- The simplest answer is to just do a word count for each book, creating a word-count vector.
- Then view the recommender problem as one of finding closest vectors to your query term in a vector space whose basis vectors are words.



Vector space example

- Books on electrodynamics use many speciality words – their vectors lie close together in one part of the vector space.
- Books on non-Euclidian geometry also use many specialty words – their vectors lie close together in a different part of the vector space.

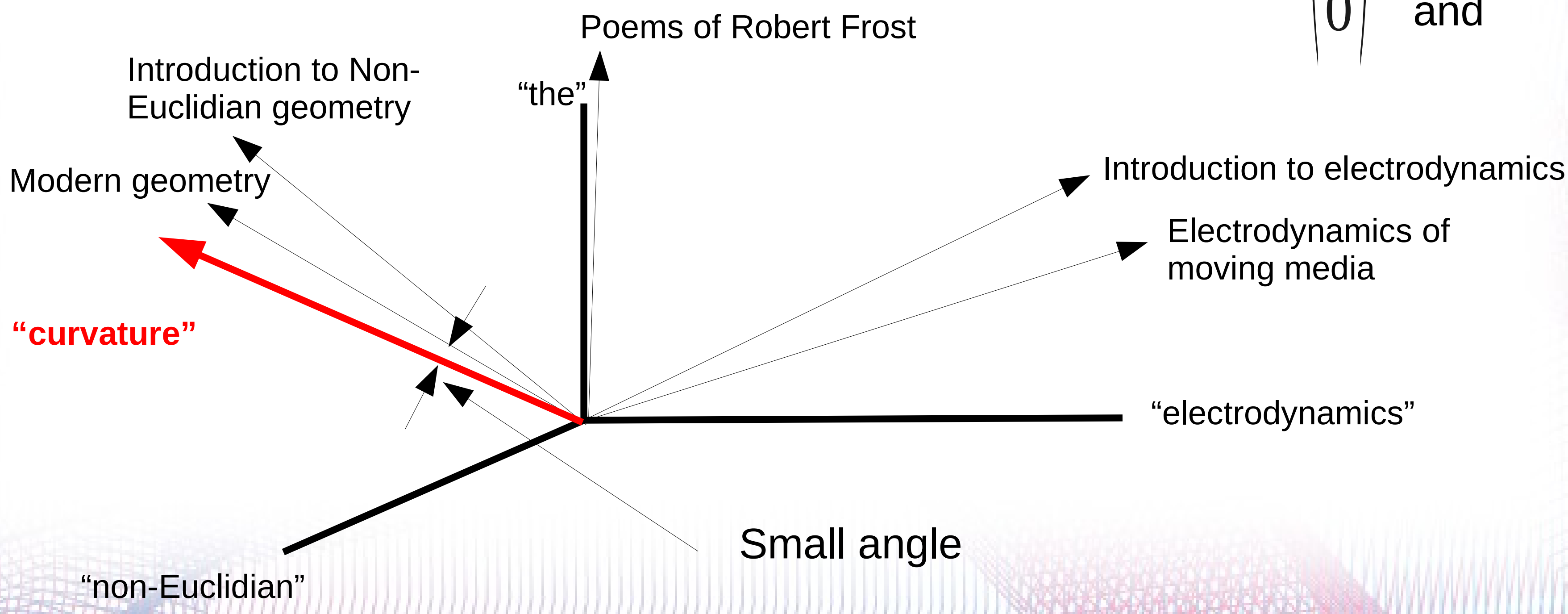


- Other books lie in different parts of this vector space.
- Common words like "and", "the" are unimportant dimensions in this space. That is, they don't help distinguish any books from one another.

How to match search term?

- Take query term as vector. Suppose term is "curvature"...
- Loop over all book vectors in the vector space, and compute at angle between query vector and book vector. Matching books have smallest angle.

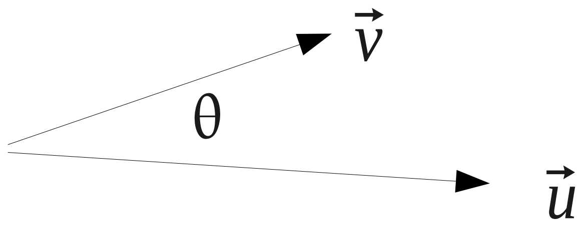
$$\vec{u} = \begin{array}{cl} \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{array}\right) & \begin{array}{l} \text{electron} \\ \text{matrix} \\ \text{vector} \\ \text{curvature} \\ \text{field} \\ \\ \text{and} \end{array} \end{array}$$



Computing angle in vector space

- In 2 and 3D, you can visualize the dot product:

$$\vec{u} \cdot \vec{v} = \|\vec{u}\| \|\vec{v}\| \cos(\theta)$$



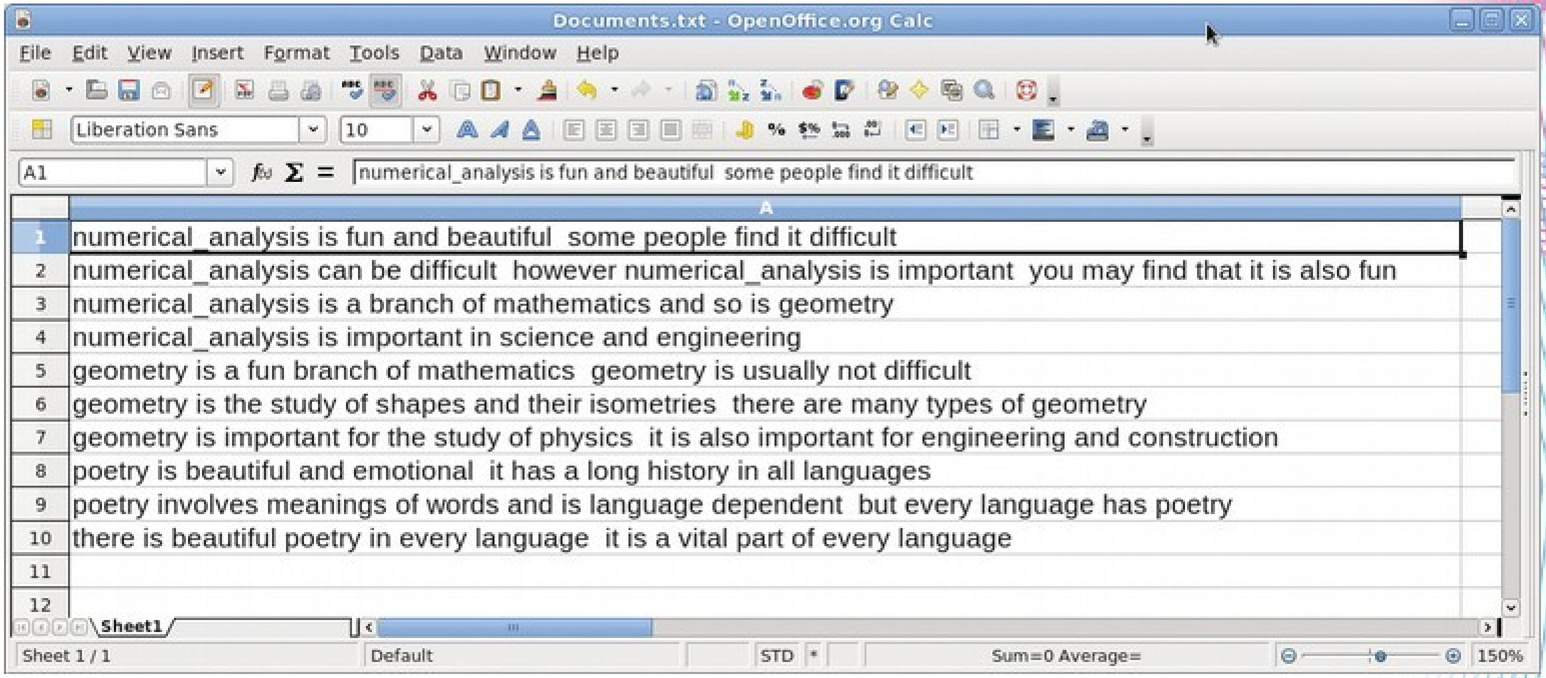
- This is also true in any dimension.
- However, instead of computing the angle, we can just compute the dot product of the vectors. This is a fast operation, and provides the same information as the angle (i.e. how close to parallel are the vectors).

Naive recommender algorithm

0. Prepare term-query matrix A from document collection beforehand.
1. Input query word as a vector.
2. Loop over all book (document) vectors.
3. Compute dot product. Append this dot product into a vector of dot products.
4. End of loop.
5. Sort dot product vector from highest to lowest.
6. Return index vector of sorted dot products. First few indices point to highly recommended books.

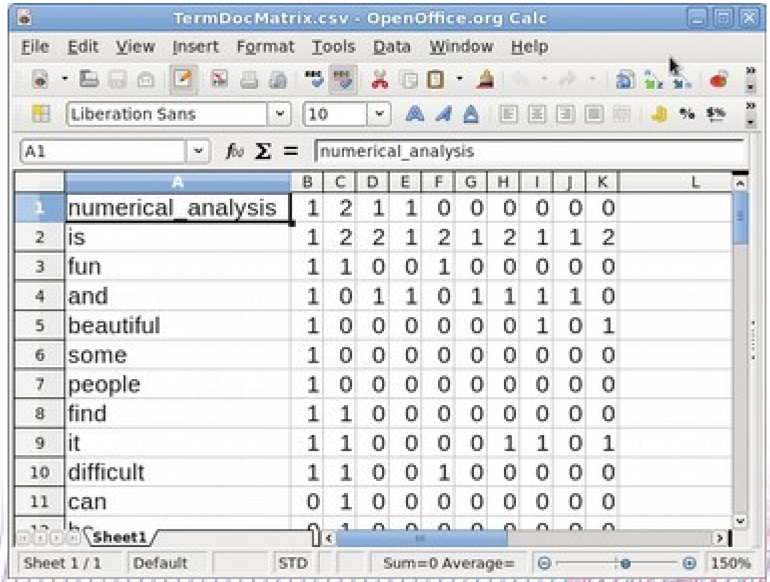
Example

Documents in rows



	A
1	numerical_analysis is fun and beautiful some people find it difficult
2	numerical_analysis can be difficult however numerical_analysis is important you may find that it is also fun
3	numerical_analysis is a branch of mathematics and so is geometry
4	numerical_analysis is important in science and engineering
5	geometry is a fun branch of mathematics geometry is usually not difficult
6	geometry is the study of shapes and their isometries there are many types of geometry
7	geometry is important for the study of physics it is also important for engineering and construction
8	poetry is beautiful and emotional it has a long history in all languages
9	poetry involves meanings of words and is language dependent but every language has poetry
10	there is beautiful poetry in every language it is a vital part of every language

Word count (term-document matrix)



	A	B	C	D	E	F	G	H	I	J	K
1	numerical_analysis	1	2	1	1	0	0	0	0	0	0
2	is	1	2	2	1	2	1	2	1	1	2
3	fun	1	1	0	0	1	0	0	0	0	0
4	and	1	0	1	1	0	1	1	1	1	0
5	beautiful	1	0	0	0	0	0	0	1	0	1
6	some	1	0	0	0	0	0	0	0	0	0
7	people	1	0	0	0	0	0	0	0	0	0
8	find	1	1	0	0	0	0	0	0	0	0
9	it	1	1	0	0	0	0	1	1	0	1
10	difficult	1	1	0	0	1	0	0	0	0	0
11	can	0	1	0	0	0	0	0	0	0	0

Results

```
>> test_find_matching_docs
================  Test A matrix  ===================
--------------------------------------------------
--->  Testing raw term-doc matrix A, search word = mathematics

Found document.  Docno = 3, Score = 1.000000.  Document:
numerical_analysis is a branch of mathematics and so is geometry

Found document.  Docno = 5, Score = 1.000000.  Document:
geometry is a fun branch of mathematics  geometry is usually not difficult

--------------------------------------------------
--->  Testing raw term-doc matrix A, search word = fun

Found document.  Docno = 1, Score = 1.000000.  Document:
numerical_analysis is fun and beautiful  some people find it difficult

Found document.  Docno = 2, Score = 1.000000.  Document:
numerical_analysis can be difficult  however numerical_analysis is important
you may find that it is also fun

Found document.  Docno = 5, Score = 1.000000.  Document:
geometry is a fun branch of mathematics  geometry is usually not difficult
```

Score is simply word count in each document.

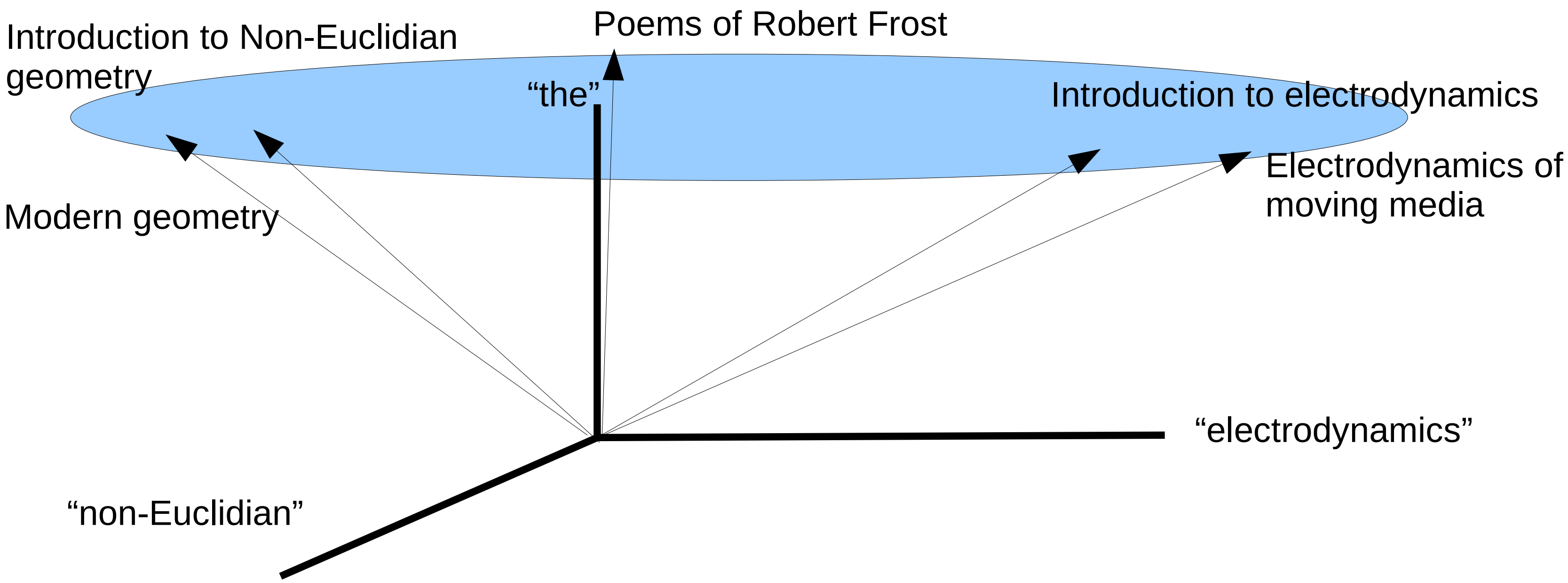
Remarks about naive word-count algorithm

- Term doc matrix is very large and very sparse.
- Algorithm grows with number of documents, and number of words.
  - For any reasonable problem, the vector space is gigantic – very high dimensionality.
- Many words are redundant or correlated (“geometry”, “mathematics”, “numerical_analysis”, etc.)
  - This means points (documents) in the vector space are clustered.
  - Potential to reduce dimensionality
- Use the SVD!

Latent semantic indexing idea

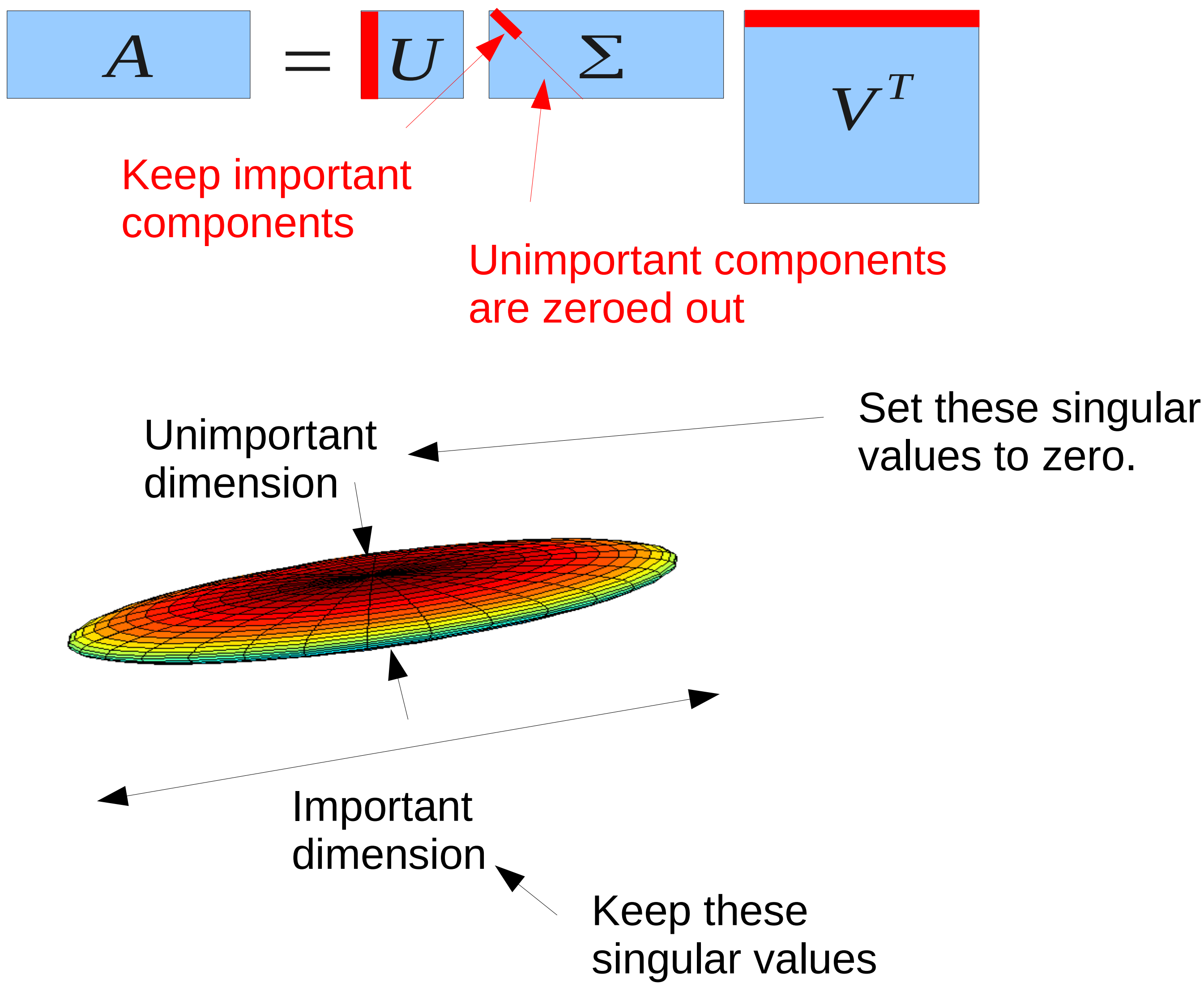
- Use SVD on term-doc matrix to reduce number of dimensions down to roughly the number of conceptual differences which exist across the document set (maybe a few hundred).
  - For us, we have 10 docs, maybe 5 or 6 concepts.
- Delete the small singular values, then recreate a smaller term-doc matrix.
  - Faster search over documents.
  - Faster computation of dot products.
  - Smaller memory footprint
  - Related “concepts” are clustered together

How does SVD eliminate unimportant dimensions?



- Identifies related "concepts" (specialist words appearing together in similar documents).
- Documents with similar "concepts" tend to lie close together in the big vector space.
- SVD reduces dimensionality so similar documents tend to lie in same hyperplane.

Dimensionality reduction by zeroing unimportant components



New query algorithm (LSI)

Preliminary work:

1. Create raw term-query matrix from document collection .
2. Compute SVD:

$$A = U \Sigma V^T$$

3. Zero out unwanted singular values, giving reduced matrices:

$$U_k, \Sigma_k, V_k^T$$

Notation means we keep *k* singular values

4. Store these matrices – they are inputs to the query algorithm.

New query algorithm (LSI)

Query work:

1. Load U_k, Σ_k, V_k^T
2. Input query word as a vector q
3. Project query word into new, reduced dim basis:

$$q_k = \Sigma_k U_k^T q$$

4. Loop over all row vectors in V_k
5. Compute dot product $\left| q_k \cdot V_k(\mathrm{row}) \right|$ Append this dot product into a vector of dot products.
6. End of loop.
7. Sort dot product vector from highest to lowest.
8. Return index vector of sorted dot products. First few indices point to highly recommended books.

```
function [doc, score] = find_matching_docs_svd(word, keywords, Uk, Sk, Vk)
  % First create column vector corresponding to word to find.
  q = zeros(length(keywords), 1);
  for idx = 1:length(keywords)
    if strcmp(word, keywords(idx))
      q(idx) = q(idx)+1;
    end
  end

  % Now transform query vector into the space spanned by the
  % reduced doc matrix Vk
  qk = inv(Sk)*Uk'*q;

  % Now iterate over documents (rows) in Vk and compute normalized
  % dot product for each.
  score = zeros(size(Vk, 1), 1);
  for didx = 1:size(Vk, 1)
    d = Vk(didx, :);
    score(didx) = dot(qk, d)/(norm(qk)*norm(d));
  end

  % Now sort scores in descending order.  Also get doc index,
  % in order of most relevant to least relevant.
  [score, doc] = sort(score, 'descend');

end
```

Query = “mathematics”

```
--->  Testing reduced term-doc matrix B, search word = mathematics

Found document.  Docno = 3, Score = 0.789235.  Document:
numerical_analysis is a branch of mathematics and so is geometry

Found document.  Docno = 5, Score = 0.708588.  Document:
geometry is a fun branch of mathematics  geometry is usually not difficult

Found document.  Docno = 1, Score = 0.349219.  Document:
numerical_analysis is fun and beautiful  some people find it difficult

Found document.  Docno = 6, Score = 0.336264.  Document:
geometry is the study of shapes and their isometries  there are many types of
geometry

Found document.  Docno = 2, Score = 0.182646.  Document:
numerical_analysis can be difficult  however numerical_analysis is important  you
may find that it is also fun

Found document.  Docno = 4, Score = 0.180679.  Document:
numerical_analysis is important in science and engineering

Found document.  Docno = 10, Score = 0.152065.  Document:
there is beautiful poetry in every language  it is a vital part of every language
```

Query = “fun”

```
--->  Testing reduced term-doc matrix B, search word = fun

Found document.  Docno = 4, Score = 0.766615.  Document:
numerical_analysis is important in science and engineering

Found document.  Docno = 1, Score = 0.735423.  Document:
numerical_analysis is fun and beautiful  some people find it difficult

Found document.  Docno = 5, Score = 0.501354.  Document:
geometry is a fun branch of mathematics  geometry is usually not difficult

Found document.  Docno = 2, Score = 0.372355.  Document:
numerical_analysis can be difficult  however numerical_analysis is important  you
may find that it is also fun

Found document.  Docno = 3, Score = 0.350525.  Document:
numerical_analysis is a branch of mathematics and so is geometry

Found document.  Docno = 10, Score = 0.154925.  Document:
there is beautiful poetry in every language  it is a vital part of every language

Found document.  Docno = 6, Score = 0.070816.  Document:
geometry is the study of shapes and their isometries  there are many types of
geometry

Found document.  Docno = 2, Score = 0.055225.  Document:
numerical_analysis can be difficult  however numerical_analysis is important  you
may find that it is also fun
```

Query = “poetry”

```
--->  Testing reduced term-doc matrix B, search word = poetry

Found document.  Docno = 9, Score = 0.894453.  Document:
poetry involves meanings of words and is language dependent  but every
language has poetry

Found document.  Docno = 8, Score = 0.340748.  Document:
poetry is beautiful and emotional  it has a long history in all languages

Found document.  Docno = 1, Score = 0.227054.  Document:
numerical_analysis is fun and beautiful  some people find it difficult

Found document.  Docno = 4, Score = 0.203128.  Document:
numerical_analysis is important in science and engineering

Found document.  Docno = 3, Score = 0.161908.  Document:
numerical_analysis is a branch of mathematics and so is geometry

Found document.  Docno = 2, Score = 0.124565.  Document:
numerical_analysis can be difficult  however numerical_analysis is important
you may find that it is also fun

Found document.  Docno = 6, Score = 0.120192.  Document:
geometry is the study of shapes and their isometries  there are many types of
geometry
```

Query = “beautiful”

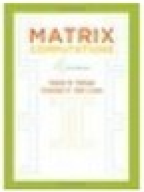
```
--->  Testing reduced term-doc matrix B, search word = beautiful

Found document.  Docno = 8, Score = 0.662326.  Document:
poetry is beautiful and emotional  it has a long history in all languages

Found document.  Docno = 1, Score = 0.625822.  Document:
numerical_analysis is fun and beautiful  some people find it difficult

Found document.  Docno = 3, Score = 0.448534.  Document:
numerical_analysis is a branch of mathematics and so is geometry

Found document.  Docno = 4, Score = 0.414169.  Document:
numerical_analysis is important in science and engineering

Found document.  Docno = 10, Score = 0.355392.  Document:
there is beautiful poetry in every language  it is a vital part of every language

Found document.  Docno = 9, Score = 0.320581.  Document:
poetry involves meanings of words and is language dependent  but every language
has poetry

Found document.  Docno = 6, Score = 0.073018.  Document:
geometry is the study of shapes and their isometries  there are many types of
geometry

Found document.  Docno = 2, Score = 0.055225.  Document:
numerical_analysis can be difficult  however numerical_analysis is important  you
may find that it is also fun
```

Remarks about LSI

- Our example is small, so dimensionality reduction only takes us from 50 terms to 6 (distinct concepts).
- However, consider 50K words in English language. Consider corpus of 100s or 1000s of documents. This is a big term-doc matrix.
- Typical real-world LSI systems reduce the matrix by keeping around 300 singular values.

Recommender systems

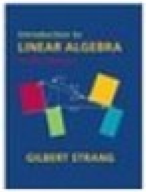
What Other Items Do Customers Buy After Viewing This Item?



Matrix Computations (Johns Hopkins Studies in the Mathematical Sciences) Hardcover
Gene H. Golub
17
$62.22 Prime

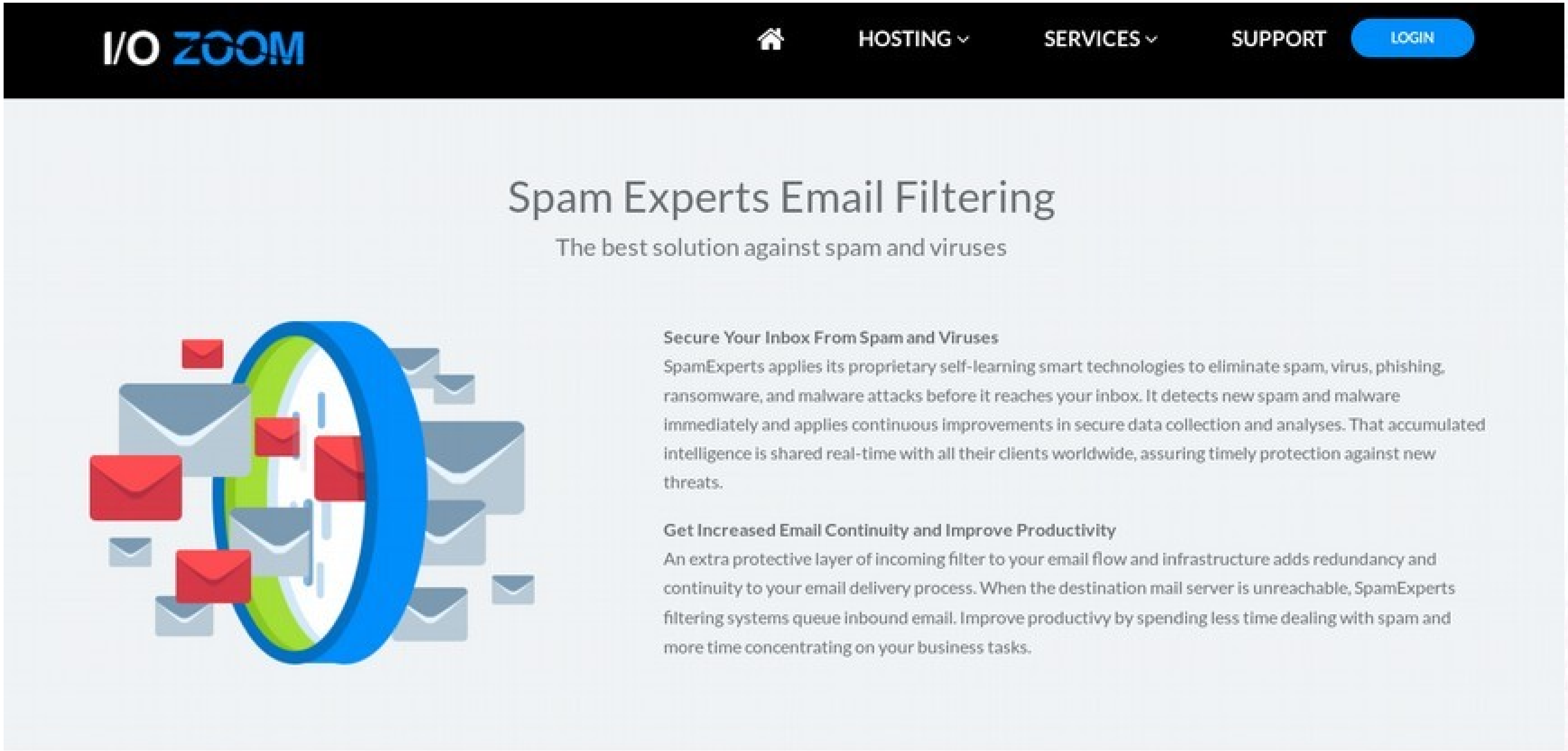
Fundamentals of Matrix Computations Hardcover
› David S. Watkins
8
$124.18 Prime

Applied Numerical Linear Algebra Paperback
James W. Demmel
9
$82.50 Prime



Introduction to Linear Algebra, Fourth Edition Hardcover
› Gilbert Strang
86

Document classification



- Spam filtering
- Web page filtering

Some applications of LSI

- Recommender systems:
  - Amazon: "Books you may like".
  - Online dating – recommend matches based on similar interests.
- Document classification systems
  - Spam filtering
- Finding relationships:
  - Facebook, LinkedIn -- "people you may know".
  - Automated systems to find terrorists.

Ideas presented in this session

- More properties of the SVD
  - Full vs. reduced SVD.
  - Outer product of two vectors.
- Dimensionality reduction
  - Eckart-Young-Mirsky theorem
  - Dramatic effect of removing small singular values from an image.
- Latent Semantic Indexing
  - Visualizing document word count as vector in vector space.
  - Recommender systems and document classification.