SVD - Low Rank Approximations

Low Rank Approximations

We now consider applications of the Singular Value Decomposition (SVD).

> SVD is “the Swiss Army Knife of Numerical Linear Algebra.”

Dianne O’Leary, MMDS ’06 (Workshop on Algorithms for Modern Massive Data Sets)

Applications

We will see how the SVD is used for

Singular Vectors and Values

For

$$A \in \mathbb{R}^{m \times n} \text{ with } m > n \text{ and rank } k,$$

there exists a set of orthogonal vectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ with $\mathbf{v}_i \in \mathbb{R}^n$

and a set of orthogonal vectors $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n\}$ with $\mathbf{u}_i \in \mathbb{R}^m$

such that

$$A\mathbf{v}_1 = \sigma_1 \mathbf{u}_1 \quad \cdots \quad A\mathbf{v}_k = \sigma_k \mathbf{u}_k, \quad A\mathbf{v}_{k+1} = 0 \quad \cdots \quad A\mathbf{v}_n = 0.$$

where $\sigma_1, \sigma_2, \dots, \sigma_k$ are the *singular values* of A.

Singular Value Decomposition

We can collect the vectors $\mathbf{v}_i$ into a matrix V and the vectors $\mathbf{u}_i$ into a matrix U.

$$A \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_m \\ | & | & & | \end{bmatrix} \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & \mathbf{0} \\ 0 & \cdots & \sigma_k & \\ \hline & \mathbf{0} & & \mathbf{0} \end{array}\right].$$

We call the $\mathbf{v}_i$ the **right singular vectors** and the $\mathbf{u}_i$ the **left singular vectors**.

Singular Value Decomposition

And because V is an orthogonal matrix, we have $VV^T = I$, so we can right multiply both sides by V^T to get

$$A = \left[\begin{array}{cccc} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_m \\ | & | & & | \end{array}\right] \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & \mathbf{0} \\ 0 & \cdots & \sigma_k & \\ \hline & \mathbf{0} & & \mathbf{0} \end{array}\right] \left[\begin{array}{cccc} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \\ | & | & & | \end{array}\right]^T.$$

We can write this as

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

Singular Value Decomposition

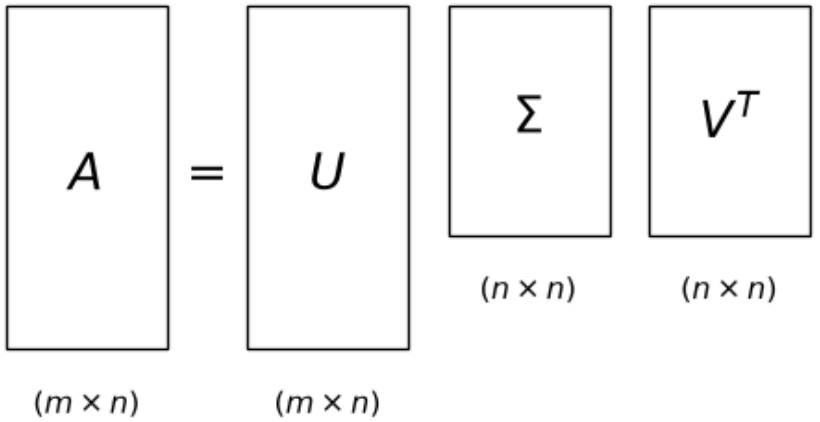
The SVD of a matrix $A \in \mathbb{R}^{m \times n}$ (where $m > n$) is

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

where

SVD Matrix Shapes

▶ Code



SVD Properties

- The SVD of a matrix always exists.
  - The existence of the SVD was proven in 1936 by Carl Eckart and Gale Young.
- The singular values are uniquely determined.
- The left and right singular vectors are uniquely determined up to ± 1.

Outer Products

The SVD can also be represented as a sum of outer products

$$A = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T,$$

where $\mathbf{u}_i$, $\mathbf{v}_i$ are the i-th columns of U and V, respectively.

Outer Products, continued

An outer product of a $m \times 1$ vector and a $1 \times n$ vector is a $m \times n$ matrix.

$$\mathbf{u}_i \mathbf{v}_i^T = \begin{bmatrix} u_{i1} \\ u_{i2} \\ \vdots \\ u_{im} \end{bmatrix} \begin{bmatrix} v_{i1} & v_{i2} & \cdots & v_{in} \end{bmatrix} = \begin{bmatrix} u_{i1}v_{i1} & u_{i1}v_{i2} & \cdots & u_{i1}v_{in} \\ u_{i2}v_{i1} & u_{i2}v_{i2} & \cdots & u_{i2}v_{in} \\ \vdots & \vdots & \ddots & \vdots \\ u_{im}v_{i1} & u_{im}v_{i2} & \cdots & u_{im}v_{in} \end{bmatrix}$$

It is a rank-1 matrix. How can you tell?

Lecture Organization

In this lecture we first discuss:

- Theoretical properties of the SVD related to
  - matrix rank
  - determining the best low rank approximations to a matrix

SVD Properties

Matrix Rank

Let $A \in \mathbb{R}^{m \times n}$ be a real matrix such that with $m > n$.

Matrix Rank and Column Space

The dimension of the column space of A is the **smallest number of vectors that suffice to construct the columns of A.**

And it is equal to the rank of the matrix.

To store a matrix $A \in \mathbb{R}^{m \times n}$ we need to store mn values.

However, if A has rank k, it can be factored as $A = CR$,

$$A = \begin{bmatrix} | & | & & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \dots & \mathbf{c}_k \\ | & | & & | \end{bmatrix} \begin{bmatrix} | & | & & | \\ \mathbf{r}_1 & \mathbf{r}_2 & \dots & \mathbf{r}_n \\ | & | & & | \end{bmatrix}$$

where $C \in \mathbb{R}^{m \times k}$ and $R \in \mathbb{R}^{k \times n}$.

This requires $k(m+n)$ values, which could be much smaller than mn when k is small, e.g. $k < \frac{mn}{m+n}$.

Low Effective Rank

In many situations we want to **approximate** a matrix A with a low-rank matrix $A^{(k)}$.

To quantify when one matrix is *close* to another, we define the distance function:

$$\operatorname{dist}(A, B) = \|A - B\|_F.$$

This can be viewed as Euclidean distance between mn-dimensional vectors.

We define the optimal **rank-k approximation** to A as

$$A^{(k)} = \underset{\{B \mid \operatorname{Rank} B = k\}}{\arg\min} \|A - B\|_F.$$

In other words, $A^{(k)}$ is the closest rank-k matrix to A.

Finding Rank-k Approximations

How can we find the optimal rank-k approximation to a matrix A?

To form the best rank-k approximation to using the SVD you calculate

$$A^{(k)} = U'\Sigma'(V')^T,$$

where

Full SVD:

$$A = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_m \\ | & | & & | \end{bmatrix} \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & \mathbf{0} \\ 0 & \cdots & \sigma_k & \\ \hline & \mathbf{0} & & \mathbf{0} \end{array}\right] \begin{bmatrix} - & \mathbf{v}_1^T & - \\ - & \mathbf{v}_2^T & - \\ & \vdots & \\ - & \mathbf{v}_n^T & - \end{bmatrix}$$

Rank-k SVD:

$$A = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_k \\ | & | & & | \end{bmatrix} \left[\begin{array}{ccc|c} \sigma_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & \mathbf{0} \\ 0 & \cdots & \sigma_k & \\ \hline & \mathbf{0} & & \mathbf{0} \end{array}\right] \begin{bmatrix} - & \mathbf{v}_1^T & - \\ - & \mathbf{v}_2^T & - \\ & \vdots & \\ - & \mathbf{v}_k^T & - \end{bmatrix}$$

For a matrix A of rank n, we can prove that

$$\|A - A^{(k)}\|_F^2 = \sum_{i=k+1}^{n} \sigma_i^2.$$

This means that the distance (in Frobenius norm) of the best rank-k approximation $A^{(k)}$ from A is equal to $\sqrt{\sum_{i=k+1}^{n} \sigma_i^2}$.

Low Rank Approximations in Practice

Models are simplifications

William of Ockham

William of Ockham (c. 1300 AD) said:

Source

This has come to be known as “Occam’s razor.”

Occam’s Razor

William was saying that it is more common for a set of observations to be determined by a simple process than a complex process.

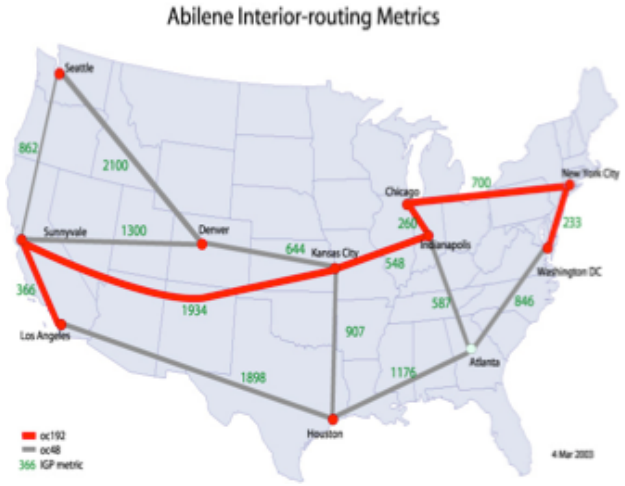
Low Effective Rank of Data Matrices

In general, a data matrix $A \in \mathbb{R}^{m \times n}$ is usually **full rank**, meaning that $\mathrm{Rank}(A) \equiv p = \min(m, n)$.

Traffic Data

Let’s see how this theory can be used in practice and investigate some real data.

We’ll look at data traffic on the Abilene network:



Source: Internet2, circa 2005

▶ Code

	ATLA-ATLA	ATLA-CHIN	ATLA-DNVR	ATLA-HSTN	ATLA-IPLS	ATLA-KSC
2003-09-01 00:00:00	8466132.0	29346537.0	15792104.0	3646187.0	21756443.0	10792818.
2003-09-01 00:10:00	20524567.0	28726106.0	8030109.0	4175817.0	24497174.0	8623734.0
2003-09-01 00:20:00	12864863.0	27630217.0	7417228.0	5337471.0	23254392.0	7882377.0
2003-09-01 00:30:00	10856263.0	32243146.0	7136130.0	3695059.0	28747761.0	9102603.0

4 rows × 121 columns

```
1 Atraf.shape
```

```
(1008, 121)
```

As we would expect, our traffic matrix has rank 121:

```
1 np.linalg.matrix_rank(Atraf)
```

```
np.int64(121)
```

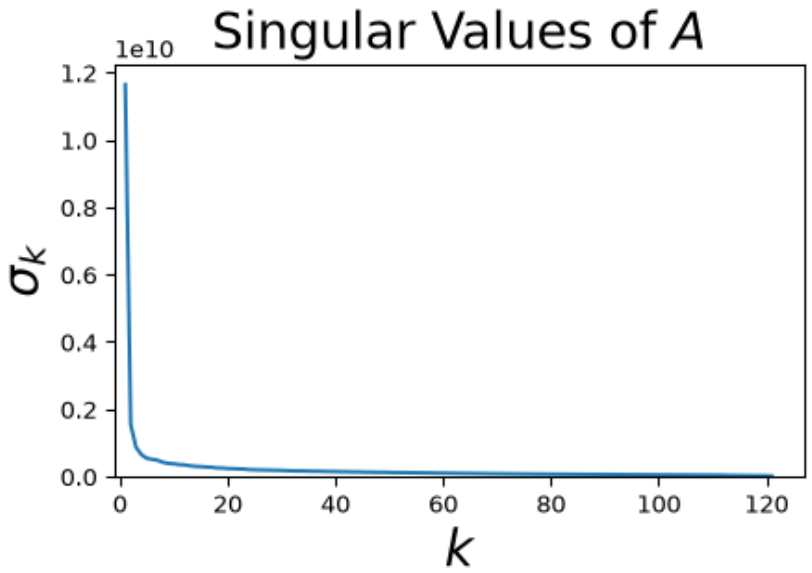
However – perhaps it has low **effective** rank.

The `numpy` routine for computing the SVD is `np.linalg.svd`:

```
1 u, s, vt = np.linalg.svd(Atraf)
```

Now let's look at the singular values of `Atraf` to see if it can be usefully approximated as a low-rank matrix:

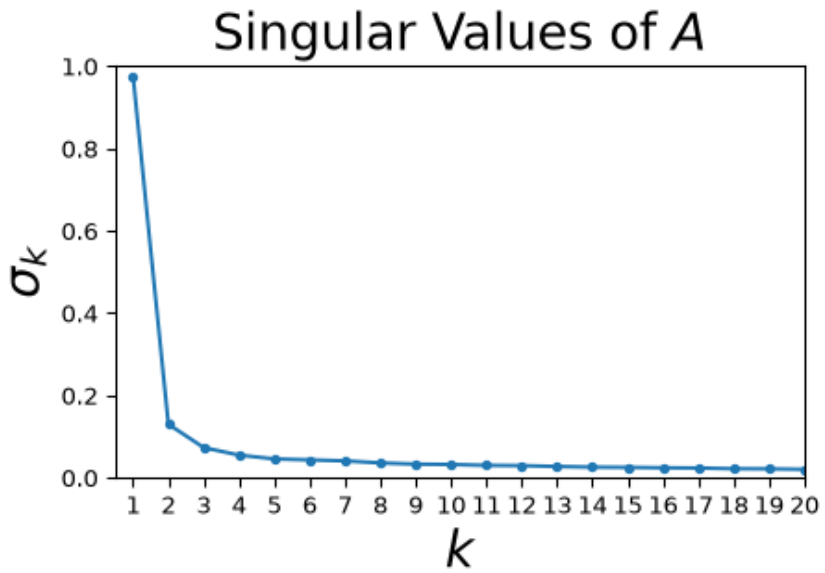
▶ Code



This classic, sharp-elbow tells us that a few singular values are very large, and most singular values are quite small.

Zooming in for just small k values, we can see that the elbow is around 4 - 6 singular values:

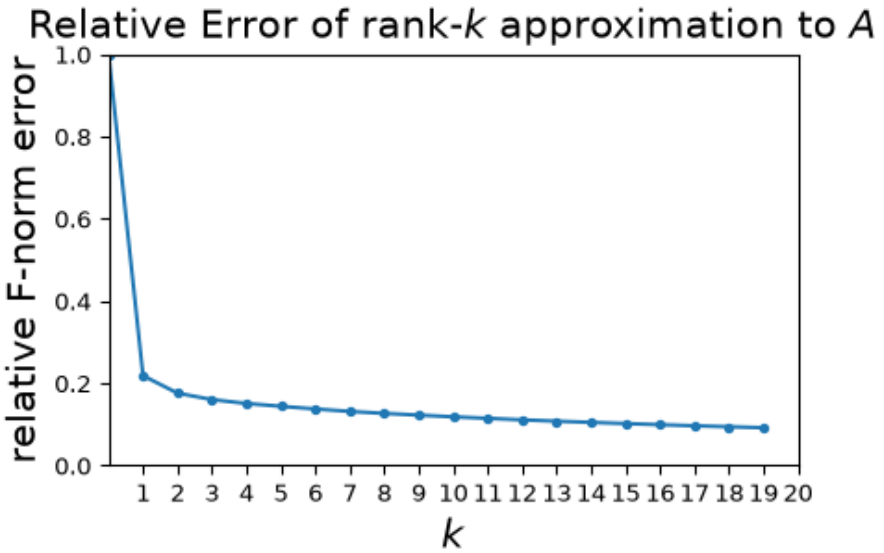
▶ Code



This pattern of singular values suggests **low effective rank.**

Let's use the formula above to compute the relative error of a rank-k approximation to A:

▶ Code



Remarkably, we are down to 9% relative error using only a rank 20 approximation to A.

So instead of storing

- $mn = (1008 \cdot 121) = 121{,}968$ values,

we only need to store

- $k(m+n) = 20 \cdot (1008 + 121) = 22{,}580$ values,

which is an 81% reduction in size.

Low Effective Rank is Common

In practice **many** datasets have low effective rank.

We consider the following examples:

Likes on Facebook

Here, the matrices are

Social Media Activity

Here, the matrices are

Netflix

Example: the Netflix prize worked with partially-observed (sparse) rating matrices like this:

$$\begin{bmatrix} & & & \vdots & & & \\ & & 3 & 2 & & 1 & \\ & 1 & & 1 & & & \\ \dots & & 2 & & 4 & & \dots \\ & 5 & 5 & & 4 & & \\ & 1 & & & 1 & 5 & \\ & & & \vdots & & & \end{bmatrix},$$

Images

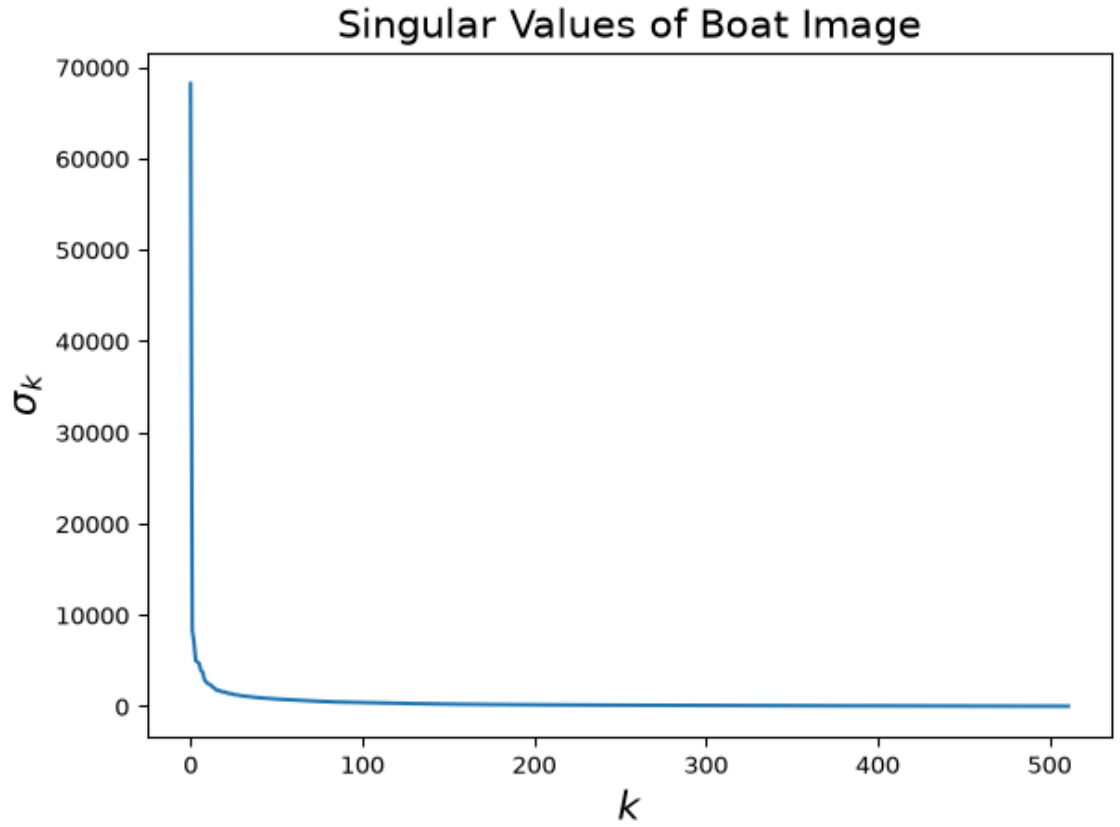
Image data often shows low effective rank.

For example, here is an original 512×512 photo:

▶ Code

Treat the image as a matrix and then compute the singular values.

▶ Code



This image is 512×512. As a matrix, it has rank of 512.

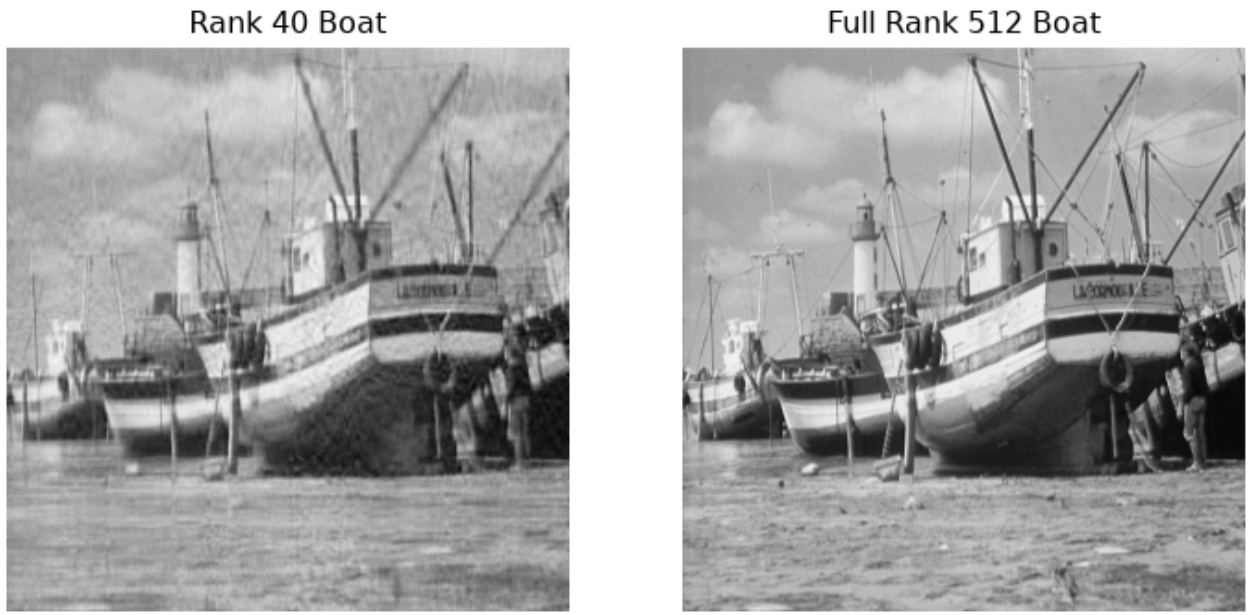
But its *effective* rank is low.

Based on the previous plot, its effective rank is perhaps 40.

Let's find the closest rank-40 matrix and view it.

```
u, s, vt = np.linalg.svd(boat, full_matrices=False)
s[40:] = 0
boatApprox = u @ np.diag(s) @ vt
```

▶ Code



Interpretations of Low Effective Rank

How can we understand the low-effective-rank phenomenon in general?

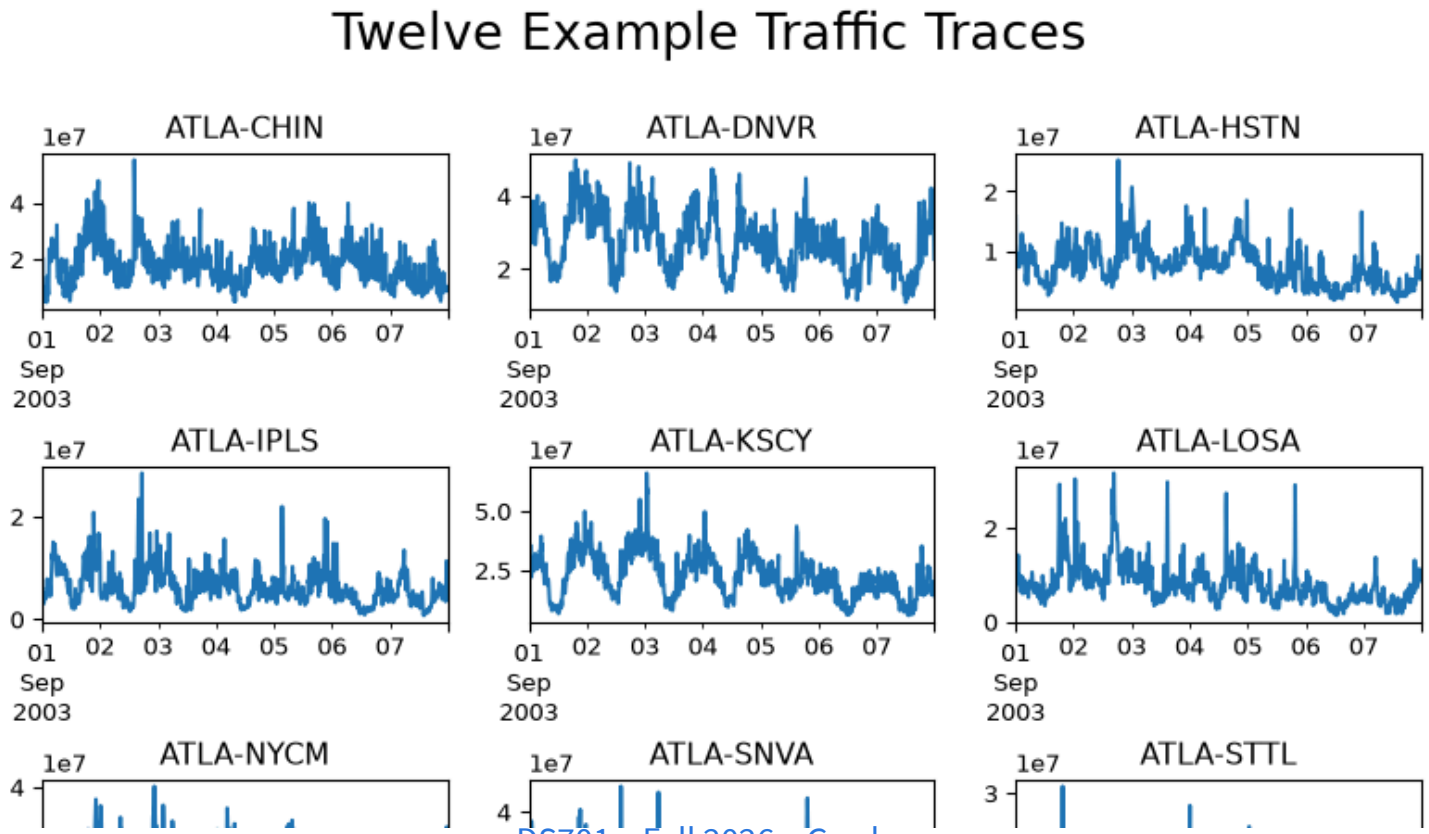
Low Rank Implies Common Patterns

The first interpretation of low-rank behavior is in answering the question:

Common Patterns: Traffic Example

Consider the set of traffic traces. There are clearly some common patterns. How can we find them?

▶ Code



Let's use as our example $\mathbf{a}_1$, the first column of A.

This happens to be the ATLA-CHIN flow.

The earlier SVD equation tells us that

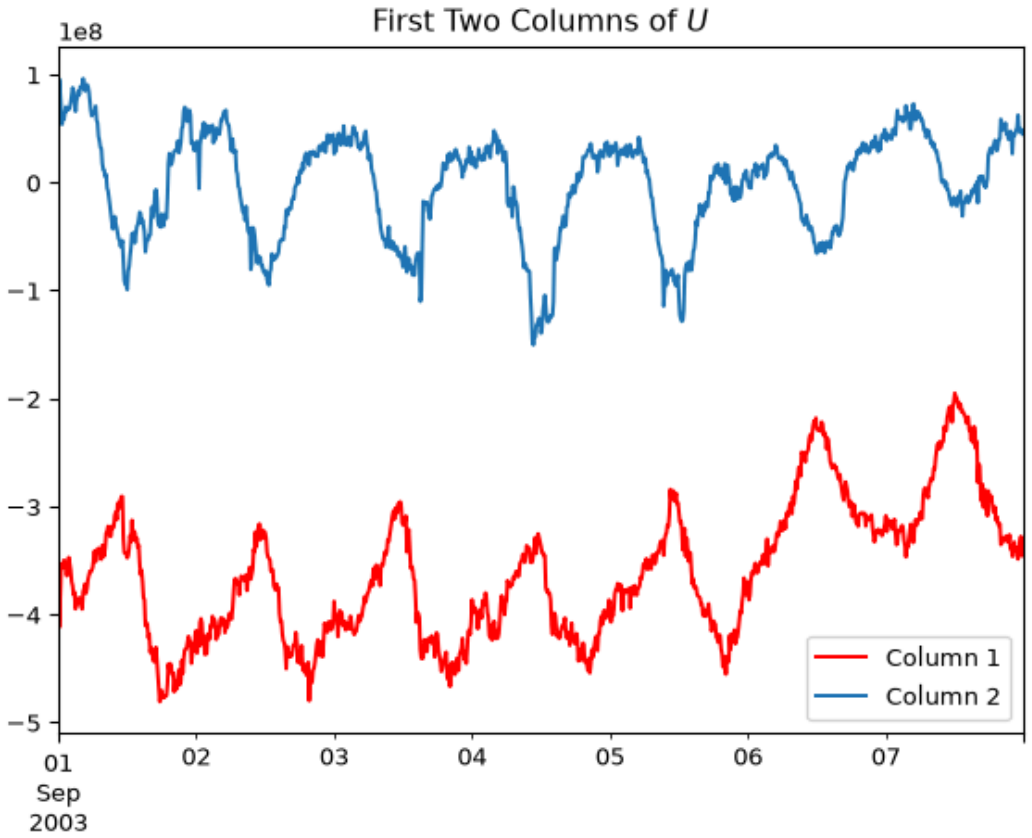
$$\mathbf{a}_1 \approx v_{11}\sigma_1\mathbf{u}_1 + v_{12}\sigma_2\mathbf{u}_2 + \cdots + v_{1k}\sigma_k\mathbf{u}_k.$$

In other words, $\mathbf{u}_1$ (the first column of U) is the "strongest" pattern occurring in A, and its strength is measured by σ_1.

Here is a view of the first 2 columns of $U\Sigma$ for the traffic matrix data.

These are the strongest patterns occurring across all of the 121 traces.

▶ Code



Low Rank Defines Latent Factors

The next interpretation of low-rank behavior is that it exposes “latent factors” that describe the data.

Latent Factors: Netflix example

Recall the structure from earlier:

$$\begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ \vdots & \vdots & & \vdots \end{bmatrix} \approx \underbrace{\begin{bmatrix} \vdots & & \vdots \\ \sigma_1 \mathbf{u}_1 & & \sigma_k \mathbf{u}_k \\ \vdots & & \vdots \end{bmatrix}}_{\tilde{U} \in \mathbb{R}^{m \times k}} \underbrace{\begin{bmatrix} \cdots & \mathbf{v}_1^T & \cdots \\ & & \\ \cdots & \mathbf{v}_k^T & \cdots \end{bmatrix}}_{\tilde{V} \in \mathbb{R}^{k \times n}},$$

where the rows of A are the users and the columns are movie ratings.

Then the rating that a user gives a movie is the inner product of a k-element vector that corresponds to the user, and a k-element vector that corresponds to the movie.

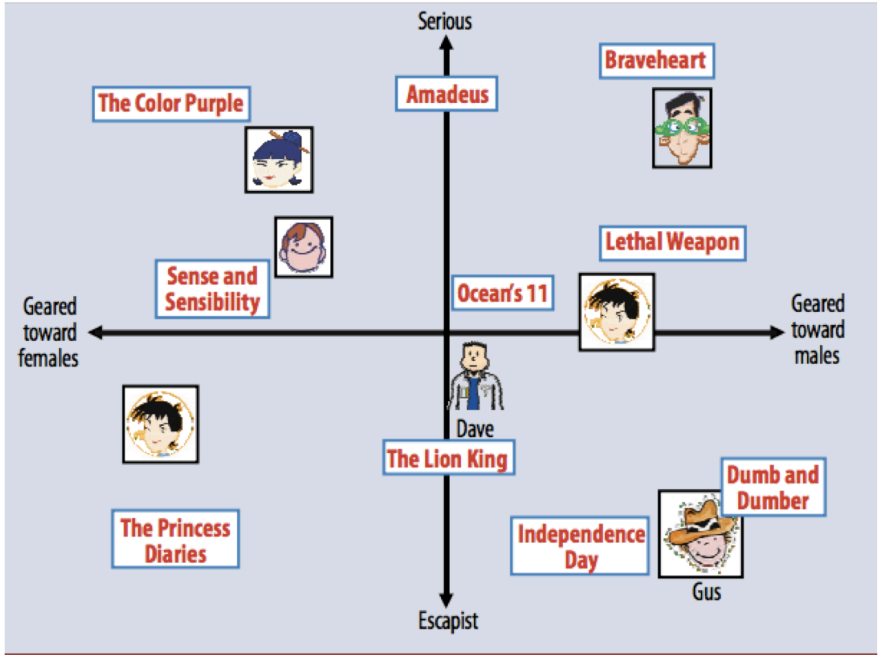
In other words we have:

$$a_{ij} = \sum_{p=1}^{k} \tilde{U}_{ip} \tilde{V}_{pj}.$$

We can think of user i's preferences as being captured by row i of $\tilde{U}$, which is a point in $\mathbb{R}^k$.

Here is a figure from the paper that described the winning strategy in the Netflix prize.

It shows a hypothetical **latent space** in which each user, and each movie, is represented by a latent vector.

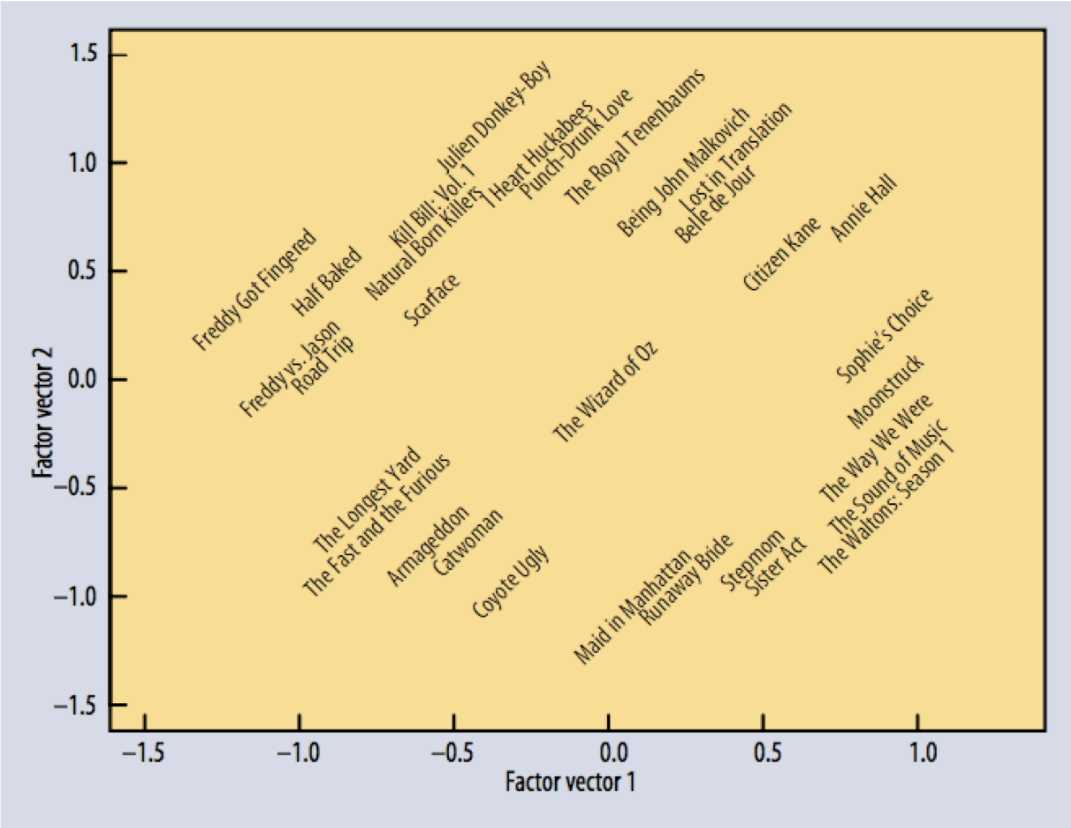


Source: Koren et al, IEEE Computer, 2009

In practice, this is perhaps a 20- or 40-dimensional space.

Here are some representations of movies in that space (reduced to 2-D).

Notice how the space seems to capture similarity among movies!



Source: Koren et al, IEEE Computer, 2009

Summary