Data Analysis in Astronomy and Physics

Lecture 13: Singular Value Decomposition

M. Röllig

Introduction

A singular value decomposition of a $m \times n$ matrix $\mathbb{A}$ of rank r is a decomposition such that:

Out[ ]//TraditionalForm=

$$\boxed{\mathbb{A}_{m\times n} = \mathbb{U}_{m\times m}\, \Sigma_{m\times n}\, (\mathbb{V}_{n\times n})^{\mathsf{T}}}$$

The columns of $\mathbb{U}$ and $\mathbb{V}$ are orthonormal and Σ is a diagonal matrix.
$\mathbb{U}$ is a $m \times n$ unitary matrix (unitary: $\mathbb{U}^{\dagger} = \mathbb{U}^{-1}$ conjugate transpose = inverse, for real matrices unitary=orthogonal, i.e. $\mathbb{U}\,\mathbb{U}^{\mathsf{T}}=\mathbb{I}$)
$\mathbb{V}$ is an adjunct unitary $n \times n$ matrix (adjunct: $(\mathbb{U}^{*})^{\mathsf{T}}=(\mathbb{U}^{\mathsf{T}})^{*}$)

For Σ we find:

$$\Sigma = \left(\begin{array}{ccc|ccc} \sigma_1 & & 0 & & \vdots & \\ & \ddots & & \cdots & 0 & \cdots \\ 0 & & \sigma_r & & \vdots & \\ \hline & \vdots & & & \vdots & \\ \cdots & 0 & \cdots & \cdots & 0 & \cdots \\ & \vdots & & & \vdots & \end{array}\right) \quad \text{with} \quad \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$$

The σ_i are the singular values of $\mathbb{A}$

Introduction

In summation notation we can write:

Out[∘]//TraditionalForm=

$$a_{\mathrm{ij}} = \sum_{k=1}^{n} u_{\mathrm{ik}}\, \sigma_k\, v_{\mathrm{jk}}$$

The sum of the squares of the singular values should be equal to the total variance in $\mathbb{A}$. Thus, the size of each tells you how much of the total variance is accounted for by each singular vector. You can create a truncated SVD containing, for instance, 99% of the variance:

Out[∘]//TraditionalForm=

$$a_{\mathrm{ij}} \approx \sum_{k=1}^{p} u_{\mathrm{ik}}\, \sigma_k\, v_{\mathrm{jk}}$$

where $p < n$ is the number of singular values that we’ve decided to keep.

Algorithm

$\mathbb{A}$ is a matrix with m rows and n columns. We assume $n \leq m$

1. Form $\mathbb{B} = \mathbb{A}^{\mathsf{T}}.\mathbb{A}$ This is a $n \times n$ matrix.
2. Compute the eigenvalues of $\mathbb{B}$. Sort them in decreasing order:
   $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k > \lambda_{k+1} \geq \ldots \lambda_n = 0$ (k is the rank of $\mathbb{A}$ and $\mathbb{B}$).
3. Form an orthonormal basis $\vec{v_1}, \ldots, \vec{v_n}$ of $\mathbb{R}^n$. Here $\vec{v_i}$ is eigenvector to the eigenvalue λ_i. $\mathbb{V}=\left[\vec{v_1}, \ldots, \vec{v_n}\right]$ is orthogonal $\mathbb{V}^{\mathsf{T}}=\mathbb{V}^{-1}$
4. The singular values of $\mathbb{A}$ are defined as $s_i = \sqrt{\lambda_i}$. The matrix Σ is a diagonal matrix with $\sigma_{\text{ii}} = s_i$ and $\sigma_{\text{ij}} = 0$ if $i \neq j$. Σ has the same dimensions as $\mathbb{A}$ (m rows, n columns)
5. For $i \leq k$ define the vectors $\vec{u_i} = \frac{1}{\sqrt{\lambda_i}} \mathbb{A}.\vec{v_i}$. Those form an orthonormal system. Use these vectors to form an orthonormal basis $\vec{u_1}, \ldots, \vec{u_m}$ in $\mathbb{R}^m$ and collect them in the matrix $\mathbb{U} = \left[\vec{u_1}, \ldots, \vec{u_m}\right]$.
6. The singular value decomposition (SVD) of $\mathbb{A}$ is
   $\mathbb{A}=\mathbb{U}.\Sigma.\mathbb{V}^{\mathsf{T}}$

Properties of SVD

Out[∘]//TraditionalForm=

$$\boxed{\mathbb{A} = \mathbb{U}\,\Sigma\,\mathbb{V}^{\mathsf{T}} \quad \Leftrightarrow \quad \mathbb{A}^{\mathsf{T}} = \mathbb{V}\,\Sigma^{\mathsf{T}}\,\mathbb{U}^{\mathsf{T}}}$$

If $\mathbb{A}$ is invertible, then $\mathbb{A}^{-1} = \mathbb{U}\,\Sigma^{-1}\,\mathbb{V}^{\mathsf{T}}$ (note that $\mathbb{U} = \mathbb{U}^{\mathsf{T}}$ and $\mathbb{V}^{-1} = \mathbb{V}^{\mathsf{T}}$)

Example

Singular value decomposition of $\mathbb{A}=\begin{pmatrix}1 & 0\\ 2 & 1\\ 0 & 1\end{pmatrix}$

1. n=2<m=3, so we can start directly: $\mathbb{B}=\mathbb{A}^{T}.\mathbb{A}$

$$\mathbb{B}=\begin{pmatrix}1 & 2 & 0\\ 0 & 1 & 1\end{pmatrix}.\begin{pmatrix}1 & 0\\ 2 & 1\\ 0 & 1\end{pmatrix}=\begin{pmatrix}5 & 2\\ 2 & 2\end{pmatrix}$$

2. Eigenvalues from $|\,\mathbb{B}-\lambda\,\mathbb{1}\,|=0=\left|\begin{pmatrix}5 & 2\\ 2 & 2\end{pmatrix}-\lambda\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\right]=p(\lambda)=\lambda^2-7\,\lambda+6=(\lambda-6)\,(\lambda-1)$

λ_1=6, $\lambda_2=1$

```
In[ ]:= Det[{{5, 2}, {2, 2}} - λ * IdentityMatrix[2]] == 0
Out[ ]= 6 - 7 λ + λ^2 == 0
```

```
Eigenvalues[{{5, 2}, {2, 2}}]
{6, 1}
```

Example

3. Eigenvector to $\lambda_1 = 6$: $(\mathbb{B} - 6\,\mathbb{1}) = \begin{pmatrix} -1 & 2 \\ 2 & -4 \end{pmatrix} \vec{v_1} = 0$, so $\vec{v_1} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.

Analogous: $\vec{v_2} = \begin{pmatrix} 1 \\ -2 \end{pmatrix}$.

We need to normalize the $\vec{v_i}$ to form $\mathbb{V} = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$

```
Eigenvectors[{{5, 2}, {2, 2}}]
```

```
{{2, 1}, {-1, 2}}
```

4. Σ contains the $\sqrt{\lambda_i}$ as diagonal elements. Pad rows with 0: $\Sigma = \begin{pmatrix} \sqrt{6} & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$

```
Append[DiagonalMatrix[√{6, 1}], {0, 0}]
```

```
{{√6, 0}, {0, 1}, {0, 0}}
```

Example

5. $\vec{u_1} = \frac{1}{\sqrt{\lambda_1}}\mathbb{A}.\vec{v_1} = \frac{1}{\sqrt{6}}\begin{pmatrix}1 & 0\\ 2 & 1\\ 0 & 1\end{pmatrix}.\frac{1}{\sqrt{5}}\{2, 1\} = \left\{\sqrt{\frac{2}{15}}, \sqrt{\frac{5}{6}}, \frac{1}{\sqrt{30}}\right\}$

$\vec{u_2} = \frac{1}{\sqrt{\lambda_2}}\mathbb{A}.\vec{v_2} = \frac{1}{\sqrt{1}}\begin{pmatrix}1 & 0\\ 2 & 1\\ 0 & 1\end{pmatrix}.\frac{1}{\sqrt{5}}\{1, -2\} = \left\{\frac{1}{\sqrt{5}}, 0, -\frac{2}{\sqrt{5}}\right\}$

We form an ON-basis of $\mathbb{R}^3$: by using Gram – Schmidt or $\vec{v_3} = \vec{v_1} \times \vec{v_2}$

MatrixForm /@ Orthogonalize$\left[\left\{\left\{\sqrt{\frac{2}{15}}, \sqrt{\frac{5}{6}}, \frac{1}{\sqrt{30}}\right\}, \left\{\frac{1}{\sqrt{5}}, 0, -\frac{2}{\sqrt{5}}\right\}, \{0, 1, 0\}\right\}\right]$

$\left\{\begin{pmatrix}\sqrt{\frac{2}{15}}\\ \sqrt{\frac{5}{6}}\\ \frac{1}{\sqrt{30}}\end{pmatrix}, \begin{pmatrix}\frac{1}{\sqrt{5}}\\ 0\\ -\frac{2}{\sqrt{5}}\end{pmatrix}, \begin{pmatrix}-\sqrt{\frac{2}{3}}\\ \frac{1}{\sqrt{6}}\\ -\frac{1}{\sqrt{6}}\end{pmatrix}\right\}$

Example

6. The singular value decomposition of $\mathbb{A}$ is:

$$\mathbb{A}=\mathbb{U}\,\Sigma\,\mathbb{V}^{T}=\begin{pmatrix} \sqrt{\frac{2}{15}} & \frac{1}{\sqrt{5}} & -\sqrt{\frac{2}{3}} \\ \sqrt{\frac{5}{6}} & 0 & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{30}} & -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{6}} \end{pmatrix} \begin{pmatrix} \sqrt{6} & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{pmatrix}$$

There is a thin version where the elements that lead to zero are left away:

$$\mathbb{A}=\mathbb{U}\,\Sigma\,\mathbb{V}^{T}=\begin{pmatrix} \sqrt{\frac{2}{15}} & \frac{1}{\sqrt{5}} & * \\ \sqrt{\frac{5}{6}} & 0 & * \\ \frac{1}{\sqrt{30}} & -\frac{2}{\sqrt{5}} & * \end{pmatrix} \begin{pmatrix} \sqrt{6} & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{pmatrix}$$

So effectively we have $\mathbb{U}$ as a m×r matrix, and Σ a r×r matrix:

$$\mathbb{A}=\mathbb{U}\,\Sigma\,\mathbb{V}^{T}=\begin{pmatrix} \sqrt{\frac{2}{15}} & \frac{1}{\sqrt{5}} \\ \sqrt{\frac{5}{6}} & 0 \\ \frac{1}{\sqrt{30}} & -\frac{2}{\sqrt{5}} \end{pmatrix} \begin{pmatrix} \sqrt{6} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{pmatrix}$$

Example

Singular value decomposition of $\mathbb{A}=\begin{pmatrix} 2 & 2 & 1 \end{pmatrix}$

The matrix is wider than high, so we have to decompose $\mathbb{A}_1 = \mathbb{A}^T$ first

1. We see that $\mathbb{B}=\mathbb{A}_1{}^T \mathbb{A}_1 = \mathbb{A}\mathbb{A}^T = \begin{pmatrix} 9 \end{pmatrix}$
2. $\lambda_1 = 9$ is the only eigenvalue of this 1×1 matrix
3. $\vec{v_1} = \begin{pmatrix} 1 \end{pmatrix}$ is a ONB of $\mathbb{R}^1$, $\mathbb{V}_1 = (1)$
4. $\Sigma=\begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix}$

Example

5. $\vec{u_1} = \frac{1}{3}\mathbb{A}_1\,\vec{v_1} = \frac{1}{3}\begin{pmatrix}2\\2\\1\end{pmatrix}$

to create an ON-basis of $\mathbb{R}^3$ we use the cross product. $\vec{w_2} = \begin{pmatrix}1\\-1\\0\end{pmatrix}$ is definitely perpendicular to $\vec{u_1}$ and $\vec{w_3} = \begin{pmatrix}2\\2\\1\end{pmatrix} \times \begin{pmatrix}1\\-1\\0\end{pmatrix} = \begin{pmatrix}1\\1\\-4\end{pmatrix}$ is perpendicular to $\vec{u_1}$ and $\vec{w_2}$. Normalizing both gives

$$\mathbb{U}_1 = \left(\vec{u_1}\ \ \vec{u_2}\ \ \vec{u_2}\right) = \begin{pmatrix}\frac{2}{3} & \frac{1}{\sqrt{2}} & \frac{1}{3\sqrt{2}}\\ \frac{2}{3} & \frac{-1}{\sqrt{2}} & \frac{1}{3\sqrt{2}}\\ \frac{1}{3} & 0 & -\frac{4}{3\sqrt{2}}\end{pmatrix}$$

Example

6. So $\mathbb{A}_1 = \mathbb{U}_1 \Sigma_1 \mathbb{V}_1^T = \begin{pmatrix} \frac{2}{3} & \frac{1}{\sqrt{2}} & \frac{1}{3\sqrt{2}} \\ \frac{2}{3} & \frac{-1}{\sqrt{2}} & \frac{1}{3\sqrt{2}} \\ \frac{1}{3} & 0 & -\frac{4}{3\sqrt{2}} \end{pmatrix} \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix} (1)$

   and

$$\mathbb{A}_1 = \mathbb{A}_1^T = \mathbb{V}_1 \Sigma_1^T \mathbb{U}_1^T = (1) \begin{pmatrix} 3 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{3\sqrt{2}} & \frac{1}{3\sqrt{2}} & -\frac{4}{3\sqrt{2}} \end{pmatrix}$$

7. Of course we can also compute the SVD of $\mathbb{A}$ directly starting with

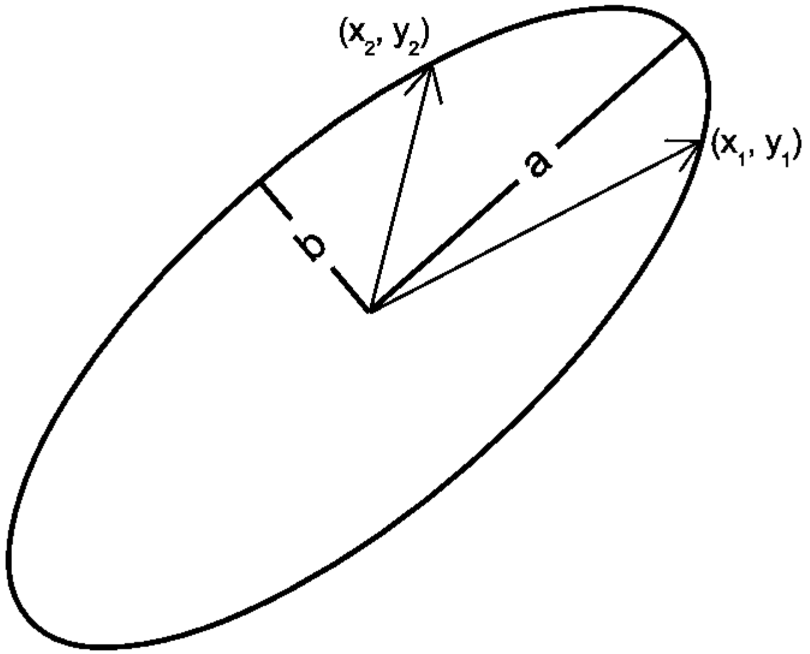
   $\mathbb{B} = \mathbb{A}^T.\mathbb{A} = \begin{pmatrix} 4 & 4 & 2 \\ 4 & 4 & 2 \\ 2 & 2 & 1 \end{pmatrix}$.

   We see that this requires more computational steps because of the eigenvalue and eigenvector determination.

Understanding SVD

To get an intuitive feel for the SVD let’s look at the simplest example in two dimensions.

Suppose, we have two, two-dimensional vectors $\vec{x_1}$ = {x1, y1} and $\vec{x_2}$ = {x2, y2}. We can fit an ellipse with major axis, *a*, and minor axis, *b*, to these two vectors as shown in the figure. But to make things easier on ourselves and save typing, we write out the equations using matrix algebra.


(x_2, y_2)
(x_1, y_1)
a
b

Understanding SVD

We can construct an ellipse of any size and orientation by stretching and rotating a unit circle. Let $\vec{x'} = \{x', y'\}$, be the transformed coordinates:

Out[•]//TraditionalForm=

$$x' = x\,R\,M^{-1}$$

where R is a rotation matrix:

Out[•]//TraditionalForm=

$$R = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

and M is a diagonal matrix containing the major and minor axes:

Out[•]//TraditionalForm=

$$M = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$$

Understanding SVD

Lets write this out term-by-term, both for the general case:

Out[ ]//TraditionalForm=

$$x_j' = \frac{1}{m_j} \sum_i x_i\, r_{\text{ij}}$$

where m_i is the i-th diagonal of the matrix, M, and for the two-dimension case:

Out[ ]=

$$\begin{aligned} x' &= \frac{x \cos(\theta) - y \sin(\theta)}{a} \\ y' &= \frac{x \sin(\theta) - y \cos(\theta)}{b} \end{aligned}$$

Note that the rotation is clockwise, opposite the usual sense because we are going from the untransformed to the transformed coordinate system rather than the other way around. For the resulting ellipse, the angle will be in the usual, counter-clockwise sense.

Understanding SVD

The equation for a unit circle is

Out[ ]//TraditionalForm=

$$x' \cdot x' = 1$$

Out[ ]//TraditionalForm=

$$\left(M^{-1} R^T x\right) \cdot \left(x R M^{-1}\right) = 1$$

We wish to fit a set of x’s, which we collect as the rows of a matrix X:

Out[ ]//TraditionalForm=

$$X = \begin{pmatrix} \vec{x_1} \\ \vec{x_2} \\ \vec{x_3} \\ \ldots \end{pmatrix} = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \ldots & \ldots \end{pmatrix}$$

The resulting matrix equation is given:

Out[ ]//TraditionalForm=

$$M^{-1} R^T X^T X R M^{-1} = 1$$

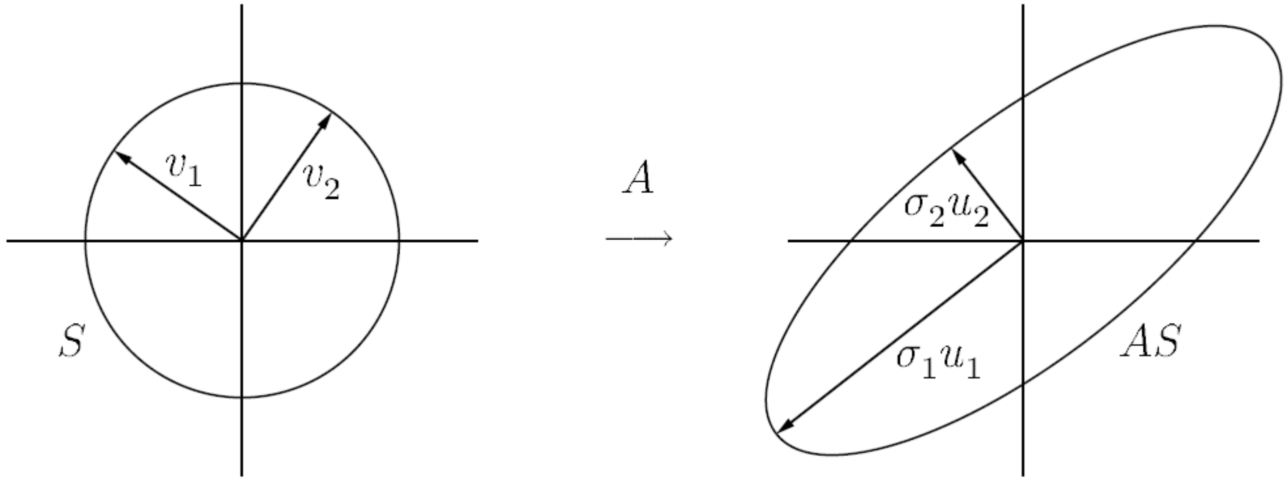
This is just a rearrangement of equation $\mathbb{A}$=$\mathbb{U}$.Σ.$\mathbb{V}^{\mathsf{T}}$ which can be rearranged as:

Out[ ]//TraditionalForm=

$$\mathbb{A}^{\mathsf{T}} \mathbb{A} \mathbb{V} = \mathbb{V} \Sigma^2$$

Understanding SVD

If we regard $\mathbb{A}$ as a collection of points, then the singular values are the axes of a least squares fitted ellipsoid while $\mathbb{V}$ is its orientation. The matrix $\mathbb{U}$ is the projection of each of the points in $\mathbb{A}$ onto the axes.



$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \left[\begin{array}{c|c|c|c} v_1 & v_2 & \cdots & v_n \end{array}\right] = \left[\begin{array}{c|c|c|c} u_1 & u_2 & \cdots & u_n \end{array}\right] \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix}$$

Application of the SVD

Pseudoinverse

The singular value decomposition can be used for computing the pseudoinverse of a matrix.
If $\mathbb{A}=\mathbb{U}\,\Sigma\,\mathbb{V}^T$ then the Pseudoinverse (or Moore-Penrose-Inverse) $\mathbb{A}^+$ of $\mathbb{A}$ is defined by: $\mathbb{A}^+=\mathbb{V}\,\Sigma^+\,\mathbb{U}^T$ where Σ^+ is computed from Σ by transposing it and replacing the non-zero diagonal elements s_{ii} by their reciprocal value $\frac{1}{s_{ii}}$.

The method of least squares is a way of “solving” an overdetermined system of linear equations

Out[ ]//TraditionalForm=

$$\mathbb{A}.\vec{x}=\vec{b}$$

i.e., a system in which $\mathbb{A}$ is a rectangular m×n matrix with more equations than unknowns (when m>n)

Application of the SVD

Least squares

As a concrete illustration, suppose that we observe the motion of a small object, assimilated to a point, in the plane. From our observations, we suspect that this point moves along a straight line, say of equation $y = dx + c$.

Suppose that we observed the moving point at three different locations (x1, y1), (x2, y2), and (x3, y3). Then, we should have

$$c + d\,x_1 = y_1$$
$$c + d\,x_2 = y_2$$
$$c + d\,x_2 = y_2$$

If there were no errors in our measurements, these equations would be compatible, and c and d would be determined by only two of the equations.

However, in the presence of errors, the system may be inconsistent. Yet, we would like to find c and d!

Least squares

The idea of the method of least squares is to determine (c, d) so that it minimizes the sum of squares of the errors, namely

$$(c + d\,x_1 - y_1)^2 + (c + d\,x_2 - y_2)^2 + (c + d\,x_3 - y_3)^2$$

In general, for an overdetermined m×n system $\mathbb{A}$x = b, what Gauss and Legendre discovered is that there are solutions x minimizing

$$\left\|\mathbb{A}\,\vec{x} = \vec{b}\right\|$$

and that these solutions are given by the square n × n system

$$\mathbb{A}^{\mathsf{T}}\,\mathbb{A}\,\vec{x} = \mathbb{A}^{\mathsf{T}}\,\vec{b}$$

Furthermore, when the columns of $\mathbb{A}$ are linearly independent, it turns out that $\mathbb{A}^{\mathsf{T}}\,\mathbb{A}$ is invertible, and so x is unique and given by

$$\vec{x} = \left(\mathbb{A}^{\mathsf{T}}\,\mathbb{A}\right)^{-1}\mathbb{A}^{\mathsf{T}}\,\vec{b}$$

Least squares

It turns out that the pseudo-inverse provides the optimal solution to the least-squares problem

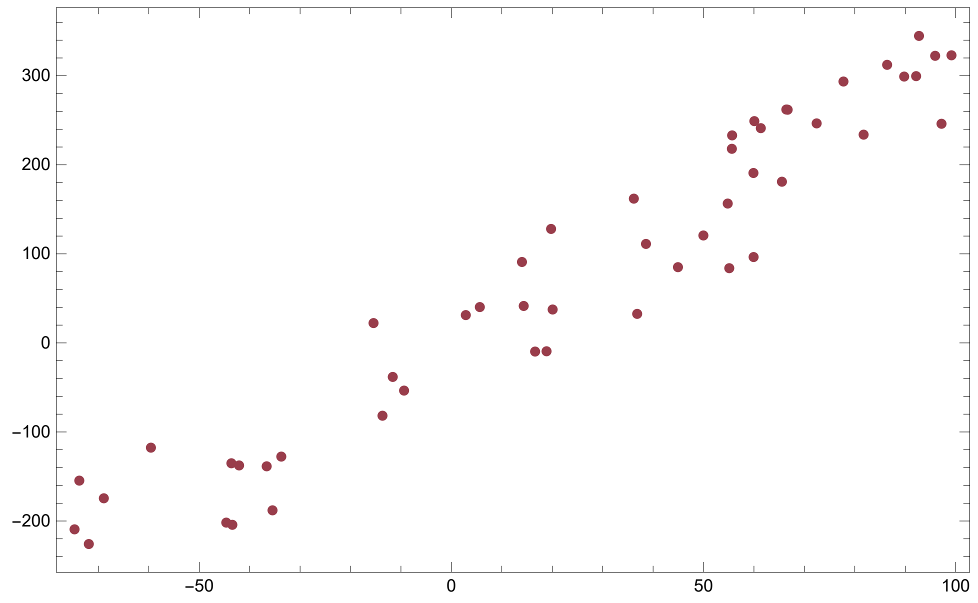
Out[ ]//TraditionalForm=

$$\vec{x}^{+} = \left(\mathbb{A}^{\mathsf{T}}\,\mathbb{A}\right)^{-1}\mathbb{A}^{\mathsf{T}}\,\vec{b} = \mathbb{A}^{+}\,\vec{b} = \mathbb{U}\,\Sigma^{+}\,\mathbb{V}^{\mathsf{T}}\,\vec{b}$$

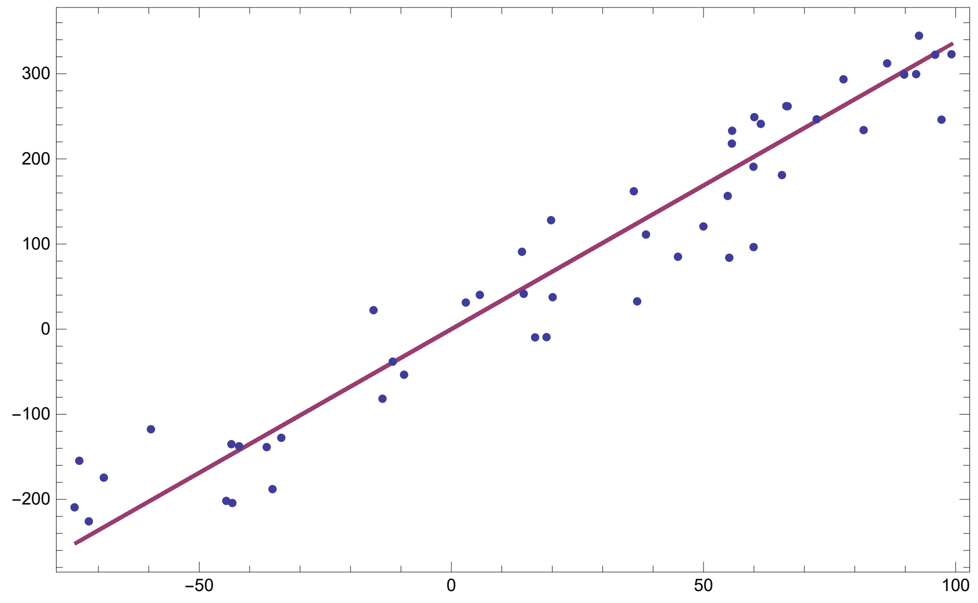
Linear Equation Fitting

```
In[ ]:= (*Generate some points around a line*)
intercept = -10;
slope = 3;
npts = 50;
noise = 80;
xs = 10 + RandomReal[{-1, 1}, npts] * 90;
ys = slope * xs + intercept + RandomReal[{-1, 1}, npts] * noise;
ListPlot[Transpose[{xs, ys}]]
(*Fit these points to a line*)
A = Transpose[{xs, ys, -ConstantArray[1, npts]}];
{U, S, V} = SingularValueDecomposition[A];
fit = V[[All, -2]];
(*Get the coefficients a,b,c in ax+by+c=0*)
a = fit[[1]];
b = fit[[2]];
c = fit[[3]];
(* Compute slope m and intercept i for y=mx+i*)
slopeEst = -a / b;
interceptEst = c / b;
ysEst = slopeEst * xs + interceptEst;
ListPlot[{Transpose[{xs, ys}], Transpose[{xs, ysEst}]}, Joined → {False, True}]
```

Out[•]=



Out[•]=



Compression

The SVD can be used to compress images, but there are some better algorithms of course.

Out[∘]=

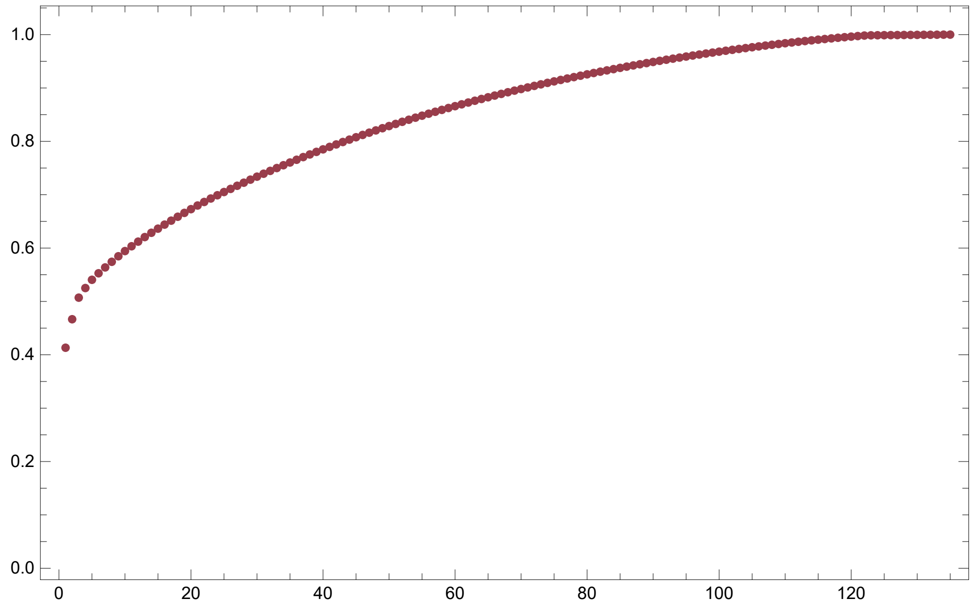
```
In[∘]:= {u, w, v} = SingularValueDecomposition[ImageData@img];

In[∘]:= Flatten[w] // Total

Out[∘]= 170.694
```

Compression

Percentage of the total information content per SV

In[ ]:=
```
ListPlot[Accumulate@ DeleteCases[Diagonal[w], 0] / (Flatten[w] // Total)]
```

Out[ ]=



In[ ]:=
```
Dimensions[w]
```

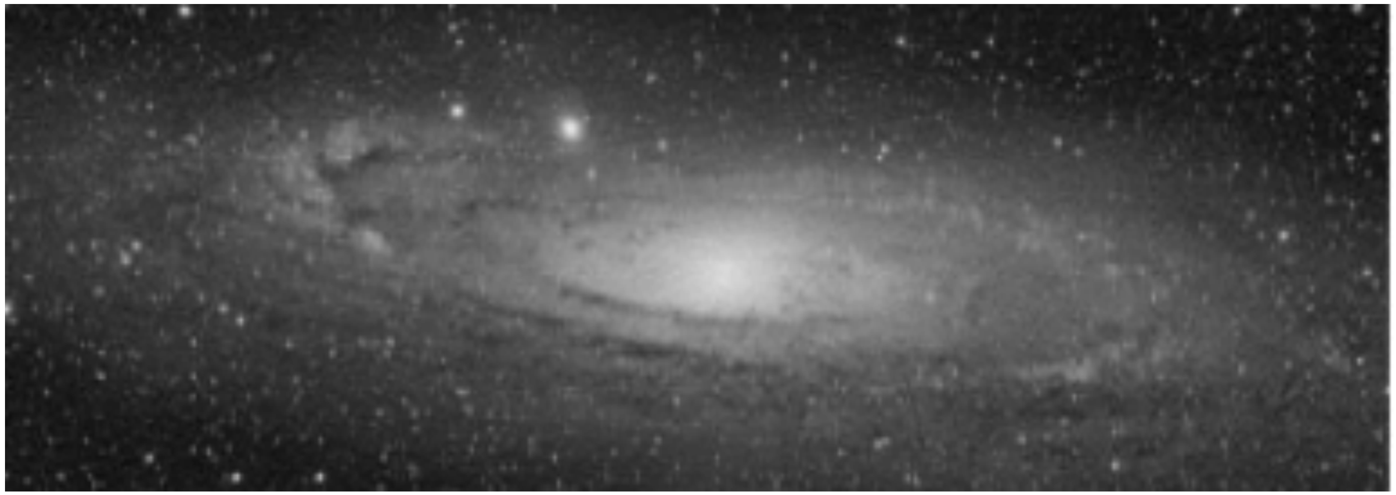
Out[ ]=
```
{135, 385}
```

Compression

Using only the first 50 singular values (setting everything else to 0).

```
In[]:= w[[50 ;; -1]] = ConstantArray[0, 385];
```

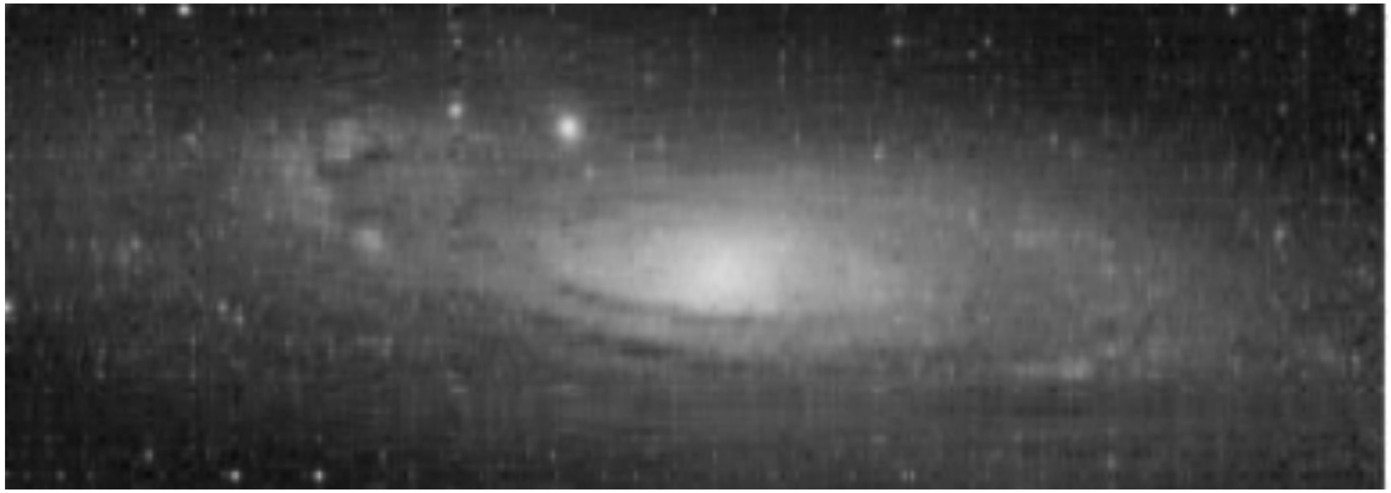
```
In[]:= Image[u.w.Transpose[v]]
```

Out[]=


Compression

Using even fewer, only the first 20 singular values (setting everything else to 0) gives a lower quality representations or the original matrix.

In[ ]:=
```
w[[20 ;; -1]] = ConstantArray[0, 385];
Image[u.w.Transpose[v]]
```

Out[ ]=


While you wouldn’t use the SVD to compress images because there are more efficient algorithms (jpg) the example shows how good the SVD is to **find a low-rank approximation to a given matrix**.

Using the first 50 singular values we compressed a 127×385 matrix into a 50×127 matrix (for U), 50 singular values and a 385×50 matrix (for V), i.e. 127*50+50+50*385=25650 bytes in the reduced image compared to 127×385=48895 bytes in the original image.

Web Search Engine

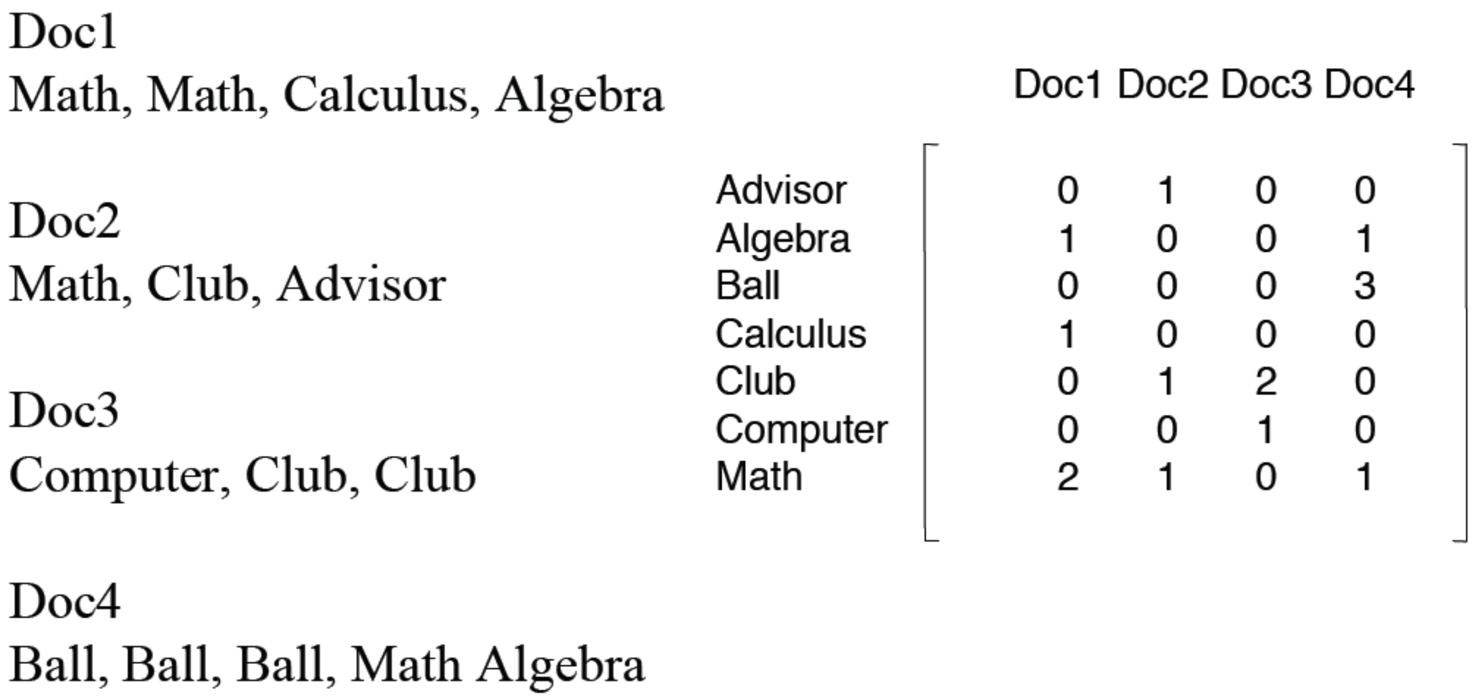
In this example we will learn how to create a web search engine using a vector space model.

Central for the discussion to follow are the term-by-document matrices.

A term-by-document matrix $A = [a_{i,j}]$ is constructed by letting the entries $a_{i,j}$ represent the frequency of the term i in document j.

We are interested in the situation when the documents are web pages, however, the method described can be applied to other documents as well, such as letters, books, phone calls, etc.

Web Search Engine



Doc1
Math, Math, Calculus, Algebra

Doc2
Math, Club, Advisor

Doc3
Computer, Club, Club

Doc4
Ball, Ball, Ball, Math Algebra

$$
\begin{array}{l|cccc|}
 & \text{Doc1} & \text{Doc2} & \text{Doc3} & \text{Doc4} \\
\text{Advisor} & 0 & 1 & 0 & 0 \\
\text{Algebra} & 1 & 0 & 0 & 1 \\
\text{Ball} & 0 & 0 & 0 & 3 \\
\text{Calculus} & 1 & 0 & 0 & 0 \\
\text{Club} & 0 & 1 & 2 & 0 \\
\text{Computer} & 0 & 0 & 1 & 0 \\
\text{Math} & 2 & 1 & 0 & 1
\end{array}
$$

We would like to search for Club using the term-by-document matrix of the Figure above.

The row corresponding to Club in this matrix is [0 1 2 0]. This shows that documents 2 and 3 contain the word Club.

Web Search Engine

Consider the matrix A from the figure and a query vector for Club,

Out[∘]=
$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix} \qquad q = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad \longleftarrow \qquad \begin{pmatrix} \text{Advisor} \\ \text{Algebra} \\ \text{Ball} \\ \text{Calculus} \\ \text{Club} \\ \text{Computer} \\ \text{Math} \end{pmatrix}$$

If a document does not contain the term “Club”, e.g. document 4, then multiplying q with the corresponding column of A gives:

```
In[∘]:= {0, 1, 3, 0, 0, 0, 1}.{0, 0, 0, 0, 1, 0, 0}
Out[∘]= 0
```

i.e. “Club” does not occur in document 4.

Web Search Engine

Taking the 3rd document on the other hand:

```
In[ ]:= {0, 0, 0, 0, 2, 1, 0}.{0, 0, 0, 0, 1, 0, 0}
Out[ ]= 2
```

shows that “Club” occur in document 3. The angle θ_j between the column j A[:,j] of a term-by-document matrix and the query vector q can be computed by

Out[ ]//TraditionalForm=

$$\cos(\theta_j) = \frac{q^{\mathsf{T}}.\mathrm{A}[\![:, j]\!]}{\|\mathrm{A}[\![:, j]\!]\| \, \|q\|}$$

```
In[ ]:= {0, 0, 0, 0, 2, 1, 0}.{0, 0, 0, 0, 1, 0, 0} / (Norm@{0, 0, 0, 0, 2, 1, 0} * Norm@{0, 0, 0, 0, 1, 0, 0})
```

Out[ ]= $\frac{2}{\sqrt{5}}$

Notice that when q is identical with the j-th column of A, the angle between q and this column vanishes and the cosine of this angle is one. Thus, a large value of cos(θ_j) suggests that document j may be relevant.

Web Search Engine

In particular, for the term-by-document matrix A and query vector q for Club, we have

```
In[]:= A = {{0, 1, 0, 0}, {1, 0, 0, 1}, {0, 0, 0, 3}, {1, 0, 0, 0}, {0, 1, 2, 0}, {0, 0, 1, 0}, {2, 1, 0, 1}};
       q = {0, 0, 0, 0, 1, 0, 0};
       fkt[{a_, b_}] := equationNoBox[a == b]
                 q.#
       N@ ─────────────────── & /@ Transpose[A];
          Norm[q] Norm[#]
       Row[Riffle[fkt /@ MapIndexed[{Cos[θFirst@#2], #1} &, %], " , "]]
Out[]= cos(θ1) = 0. , cos(θ2) = 0.57735 , cos(θ3) = 0.894427 , cos(θ4) = 0.
```

This shows that document 3 is the best match.

In applications of the SVD to document retrieval, the columns u_j of U commonly are referred to as term vectors and the columns v_j of V as document vectors.

Web Search Engine

Remember, that

Out[ ]//TraditionalForm=

$$\mathbb{A} = \mathbb{U}\,\Sigma\,\mathbb{V}^{\mathsf{T}}$$

By ignoring the last $n - k$ terms we have:

Out[ ]//TraditionalForm=

$$\mathbb{A}_k = \mathbb{U}_k\,\Sigma_k\,(\mathbb{V}_k)^{\mathsf{T}}$$

The projection of the scaled documents onto a k - dimensional space is given by

Out[ ]//TraditionalForm=

$$\Sigma_k\,\mathbb{V}_k = \mathbb{A}^{\mathsf{T}}\,\mathbb{U}_k$$

and, similarly, the projection of the scaled terms onto a k-dimensional space can be expressed as

Out[ ]//TraditionalForm=

$$\mathbb{U}_k\,\Sigma_k = \mathbb{A}\,\mathbb{V}_k$$

Web Search Engine

```
{u, Σ, v} = SingularValueDecomposition[A, 2] // N;
MatrixForm /@ %
```

Out[•]=

$$\left\{\begin{pmatrix} 0.0535112 & -0.205275 \\ 0.369305 & 0.00394598 \\ 0.733978 & 0.363347 \\ 0.124646 & -0.11717 \\ 0.0811402 & -0.790577 \\ 0.0138145 & -0.292651 \\ 0.547462 & -0.318499 \end{pmatrix}, \begin{pmatrix} 3.57031 & 0. \\ 0. & 2.53039 \end{pmatrix}, \begin{pmatrix} 0.445025 & -0.296485 \\ 0.191052 & -0.519426 \\ 0.049322 & -0.74052 \\ 0.87351 & 0.306469 \end{pmatrix}\right\}$$

Let $e_j = [0, \ldots, 0, 1, 0, \ldots, 0\}^T$ denote the j-th axis vector. The point

Out[•]//TraditionalForm=

$$(e_5)^\mathsf{T}\, U_2\, \Sigma_2 = (\sigma_1 u_{1,5}\ ,\ \sigma_2 u_{2,5}) = \{-0.289696,\ 2.00047\}$$

is the projection of the 5th term (Club), where $u_{j,5}$ denotes the 5th entry of the vector u_j.

Web Search Engine

Similarly

Out[∘]//TraditionalForm=

$$(e_3)^{\mathsf{T}}\, U_2\, \Sigma_2 = (\sigma_1 u_{1,3}\ ,\ \sigma_2 u_{2,3}) = \{-0.176095,\ 1.8738\}$$

Out[∘]//TableForm=

	x-coordinates	y-coordinates
Terms	$\sigma_1 u_1$	$\sigma_2 u_2$
Documents	$\sigma_1 v_1$	$\sigma_2 v_2$

Out[∘]=

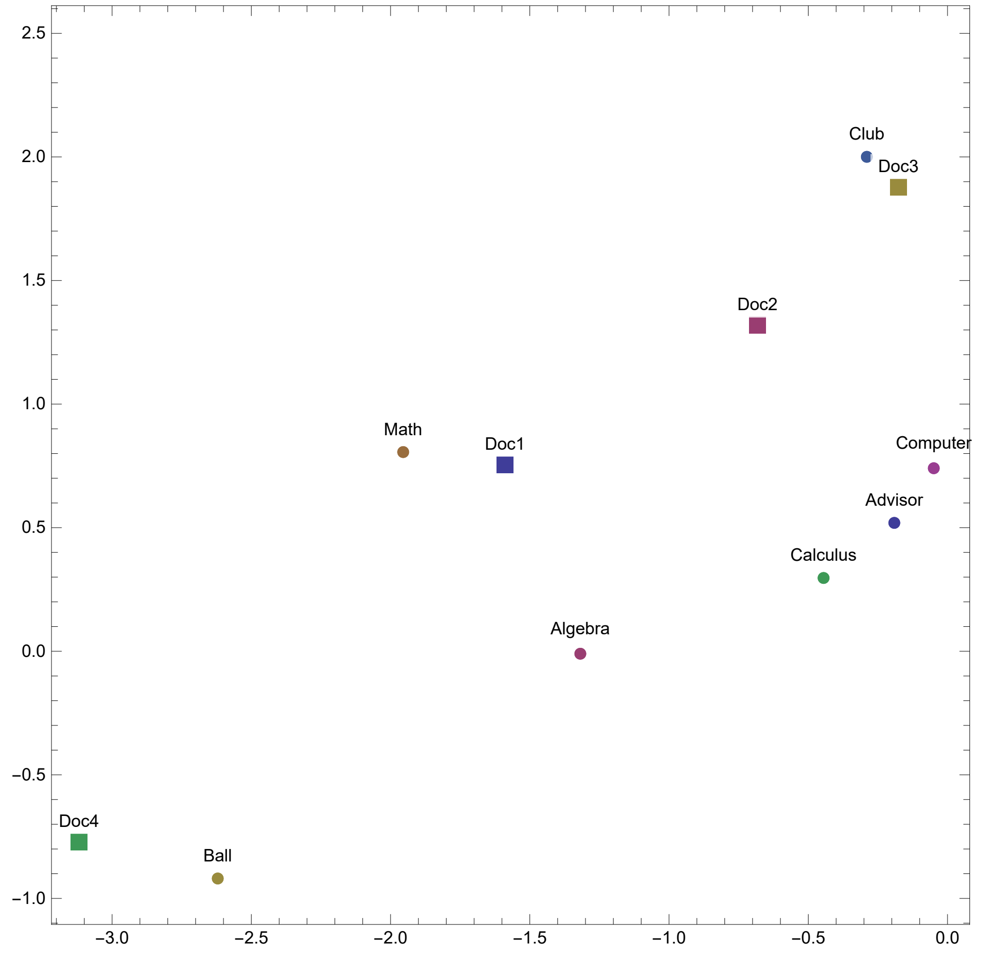
$\sigma_1 u_1$	$\sigma_2 u_2$
-0.191052	0.519426
-1.31853	-0.00998485
-2.62053	-0.919408
-0.445025	0.296485
-0.289696	2.00047
-0.049322	0.74052
-1.95461	0.805926

$\sigma_1 v_1$	$\sigma_2 v_2$
-1.58888	0.750221
-0.682114	1.31435
-0.176095	1.8738
-3.1187	-0.775487

Web Search Engine

The Figure shows the two-dimensional projection of the terms and documents in the Example

Out[ ]=



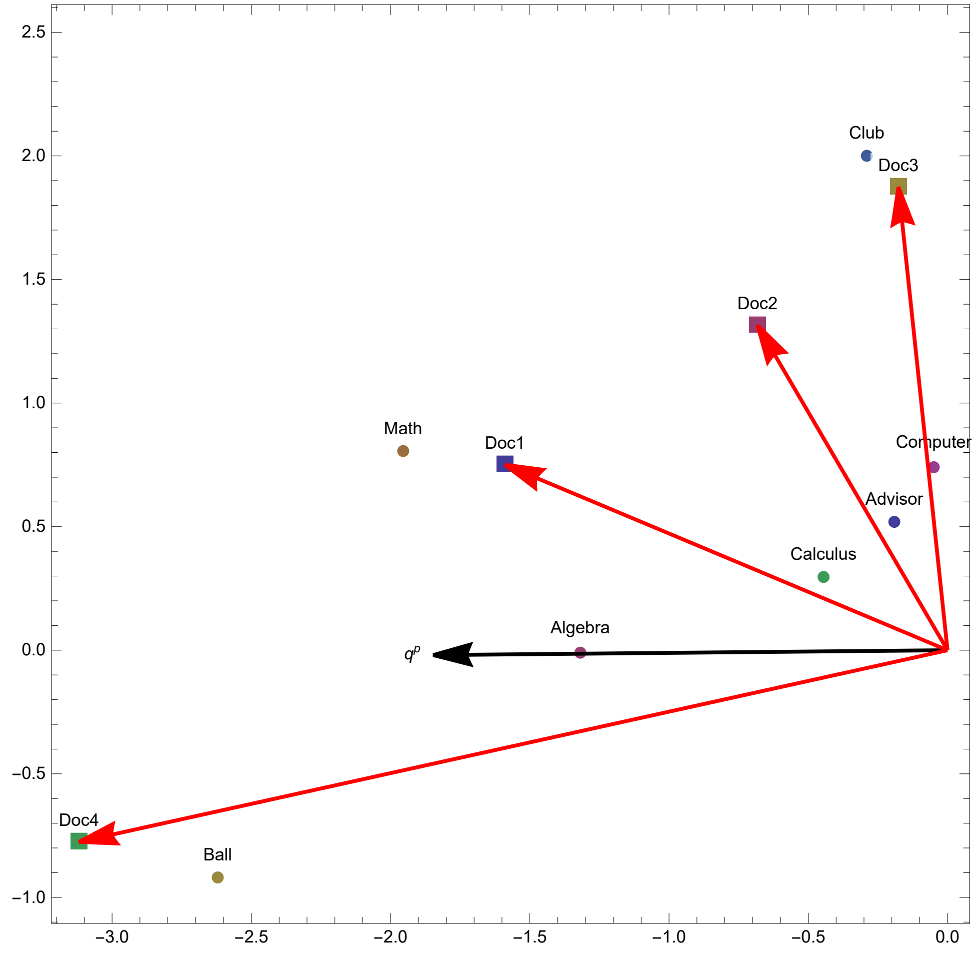
The projections of the 5th (Club) term and 3rd document are close, which indicates that Doc3 is likely to be a relevant document when we search for Club, in agreement with the discussion above.

In[ ]:=
```
-{0, 1, 0, 0, 0, 0, 0}.u
```

```
Out[ ]= {-0.36930527765711485`, -0.0039459754863706105`}
```

Web Search Engine

Out[ ]=

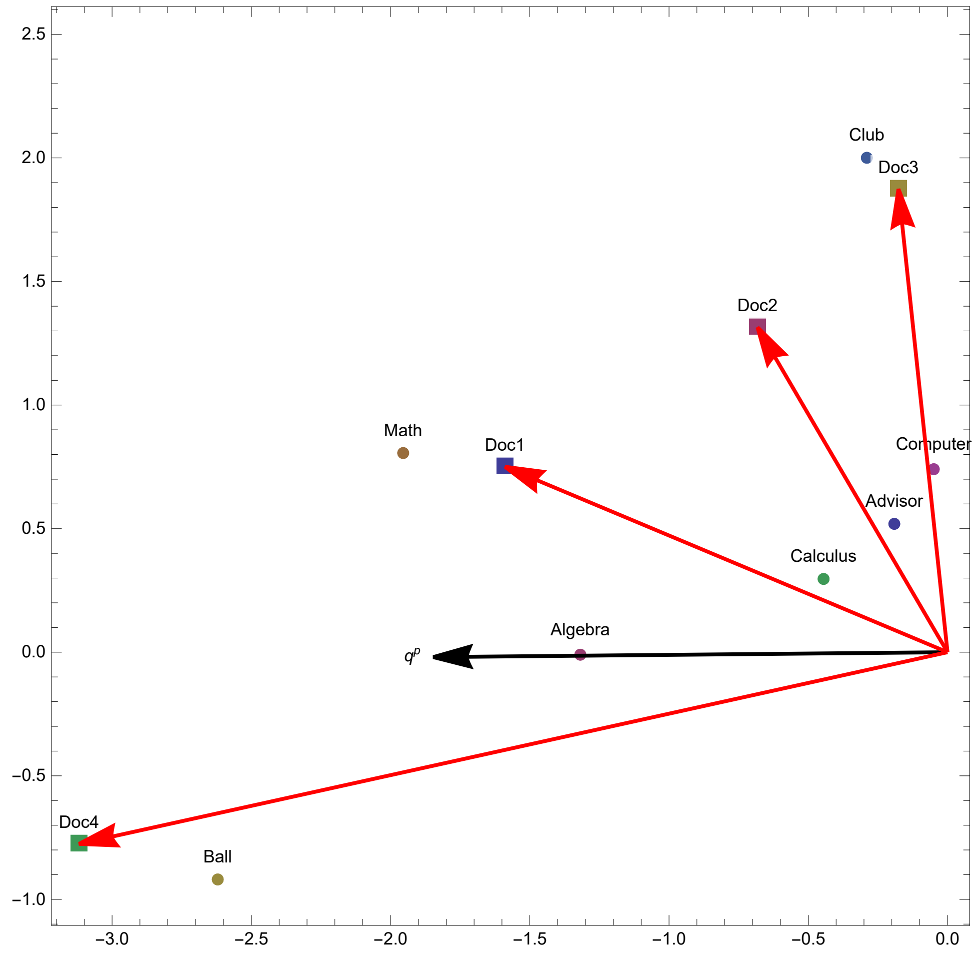


Note that Algebra does not appear in Doc2, however, the Figure shows the angle between q^p and the projection of Doc2 to be smaller than 90°.This indicates that Doc2 may be somewhat relevant to Algebra.

Doc2 contains the words Club, Math, and Advisor, and Doc4 and Doc1 contain the words Math and Algebra. All three documents contain the word Math, and therefore the documents may be relevant to the query. Hence, Doc2 could be relevant.

Web Search Engine

Out[∘]=



Also, notice that Doc3 does not contain the words Math and Algebra, which indicates that this document is unrelated to the query. In fact, the angle between the query vector for Algebra and Doc3 in the Figure is almost 90°.

Literally matching terms in documents with a query does not always yield clear answers. A better approach is to match a query with the meaning of a document. Latent Semantic Indexing (LSI) does just that. It overcomes the problems of lexical matching by using statistically derived quantities instead of individual words for matching.

Latent Semantic Indexing

Latent semantic indexing examines the whole document collection to determine which documents contain similar words. LSI considers documents that have words in common to be semantically close, and documents with few words in common to be semantically distant.

Let A_k be the rank k approximation of A. Working with A_k instead of A has some advantages: memory efficiency, removal of unimportant information, smaller search space, ...

The value of k needs to be determined empirically. Even for a very large number of documents, $k = 100$ often gives acceptable results.

Latent Semantic Indexing

LSI with A_k instead of the matrix A. Specifically, one determines the angle between the query vector q and every column of A_k. These angles can be computed

Out[ ]= $$\cos(\theta_j) = \frac{\vec{q}^{\,T} A_k \vec{e}_j}{\|\vec{q}\| \left\|A_k \vec{e}_j\right\|} = \frac{\left(\vec{q}^{\,T} \mathbb{U}_k\right) \left(\Sigma_k \mathbb{V}^T \vec{e}_j\right)}{\|\vec{q}\| \left\|\Sigma_k \mathbb{V}^T \vec{e}_j\right\|} \qquad j = 1, 2, \ldots, k$$

Where e_j denotes the j-th axis vector. Note that, $\vec{q}^{\,T} \mathbb{U}_k$ and $\Sigma_k \mathbb{V}^T$ are projections of the query vector and document vectors into k-dimensional spaces

Latent Semantic Indexing

Using k = 2 for LSI, we obtain the following results. For Club, we have

```
Out[•]= cos(Θ1) = 0.410924 , cos(Θ2) = 0.739083 , cos(Θ3) = 0.7947 , cos(Θ4) = -0.112031
```

$\cos(\Theta_1) = 0.410924$, $\cos(\Theta_2) = 0.739083$, $\cos(\Theta_3) = 0.7947$, $\cos(\Theta_4) = -0.112031$

which gives the ordering Doc3, Doc2, Doc1, and Doc4 of the documents, with Doc3 being the most relevant and Doc4 the least relevant document.

Latent Semantic Indexing

The query vector for Algebra yields

```
Out[ ]= cos(θ1) = 0.332266 , cos(θ2) = 0.166613 , cos(θ3) = 0.0306254 , cos(θ4) = 0.359344
```

which determines the ordering Doc4, Doc1, Doc2, and Doc3. Notice that the cosine of the angle between Algebra and Doc3 is close to zero. This indicates that Doc3 is irrelevant for the query. The angles reveal that Doc2 has some connection to Algebra.

Finally, we note that weighting of the terms in a term-by-document matrix can affect query results significantly. Documents that have many words or repeat the same word many times are erroneously given a high ranking without term weighting.

● Init