Applications of Singular-Value Decomposition (SVD)

Alkiviadis G. Akritas* and Gennadi I. Malaschonok†

Summer 2002

Abstract

Let A be an $m \times n$ matrix with $m \geq n$. Then one form of the singular-value decomposition of A is

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

where U and V are orthogonal and Σ is square diagonal. That is, $UU^T = I_{rank(A)}$, $VV^T = I_{rank(A)}$, U is $rank(A) \times m$, V is $rank(A) \times n$ and

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \sigma_{rank(A)-1} & 0 \\ 0 & 0 & \cdots & 0 & \sigma_{rank(A)} \end{pmatrix}$$

is a $rank(A) \times rank(A)$ diagonal matrix. In addition $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{rank(A)} > 0$. The σ_i's are called the *singular values* of A and their number is equal to the rank of A. The ratio $\frac{\sigma_1}{\sigma_{rank(A)}}$ can be regarded as a condition number of the matrix A.

It is easily verified that the singular-value decomposition can be also written as

$$A = U^T \Sigma V = \sum_{i=1}^{rank(A)} \sigma_i u_i^T v_i.$$

The matrix $u_i^T v_i$ is the *outer product* of the i-th row of U with the corresponding row of V. Note that each of these matrices can be stored using only $m+n$ locations rather than mn locations.

Using both forms presented above—and following Jerry Uhl's beautiful approach in the *Calculus and Mathematica* book series [1]—we show how SVD can be used as a tool for teaching Linear Algebra geometrically, and then apply it in solving least-squares problems and in data compression.

In this paper we used the Computer Algebra system *Mathematica* to present a purely numerical problem. In general, the use of Computer Algebra systems has greatly influenced the teaching of mathematics, allowing students to concentrate on the main ideas and to visualize them.

*University of Thessaly, Department of Computer and Communication Engineering, GR-38221 Volos, Greece

†Tambov University, Department of Mathematics, Tambov, Russia. This material is based on work supported in part by the Russian Ministry of Education Grant E02-2.0-98.

1 Introduction

In this section, we introduce the staple "ingredients" of any matrix, namely the hanger, stretcher and aligner matrices. As we will see, any matrix can be written as the product of a hanger, a stretcher and an aligner matrix.

We begin our discussion with the concept of a perpendicular frame in the two-dimensional space ($2D$). We restrict our attention to $2D$ spaces, and hence, to 2×2 matrices. Extensions to higher dimensions are easily made.

A $2D$ perpendicular frame consists of two perpendicular *unit* vectors which can be specified by an angle s. As an example, consider the unit vectors $perpframe[1] = \{Cos(s), Sin(s)\}$ and $perpframe[2] = \{Cos(s + \frac{\pi}{2}), Sin(s + \frac{\pi}{2})\}$ for which

$$perpframe[i] \cdot perpframe[i] = 1,$$

$(i = 1, 2)$ and

$$perpframe[1] \cdot perpframe[2] = 0.$$

Using these two perpendicular vectors as columns or rows we can define, respectively, the hanger or the aligner matrix.

1.1 Hanger matrices for hanging a curve on a perpendicular frame

Given the perpendicular frame $perpframe[1] = \{Cos(s), Sin(s)\}$,
$perpframe[2] = \{Cos(s + \frac{\pi}{2}), Sin(s + \frac{\pi}{2})\}$, the hanger matrix defined by it is:

$$hanger(s) = \begin{pmatrix} Cos(s) & Cos(s + \frac{\pi}{2}) \\ Sin(s) & Sin(s + \frac{\pi}{2}) \end{pmatrix},$$

where $perpframe[1]$ and $perpframe[2]$ are the two columns. To see its action on a curve, set the angle s for the hanger matrix to $s = \frac{\pi}{4}$ and consider the ellipse $\{x(t), y(t)\} = \{1.5Cos(t), 0.9Sin(t)\}$, in its parametric form (Figure 1).

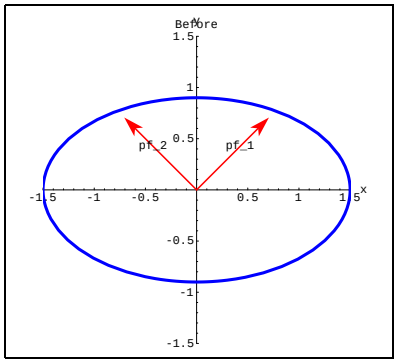

Figure 1: A perpendicular frame, $s = \frac{\pi}{4}$, together with an ellipse "hung" on the $x - y$ axes.

We mention in passing that this is a *right hand* perpendicular frame because if we mentally rotate it, pushing $perpframe[1]$ onto the x-axis, $perpframe[2]$ automatically aligns with the *positive* y-axis. If $perpframe[2]$ aligns with the *negative* y-axis, we are dealing with a *left hand* perpendicular frame .

Clearly, the ellipse is hung on the $x-y$ axes. The unit vectors $\{1,0\}$ and $\{0,1\}$ form a perpendicular frame and the vectors defined by the points on the ellipse are of the form:

$$\{x(t), y(t)\} = x(t)\{1,0\} + y(t)\{0,1\}.$$

This means that to get to a point $\{x(t), y(t)\}$ on the original curve, we advance $x(t)$ units in the direction of $\{1,0\}$ and then $y(t)$ units in the direction of $\{0,1\}$.

To hang the ellipse on the given perpendicular frame, all we have to do to the above expression is to simply replace $\{1,0\}$ by $perpframe[1]$ and $\{0,1\}$ by $perpframe[2]$. To wit, the vectors defined by the points on the ellipse now become:

$$hungellipse(t) = x(t)perpframe[1] + y(t)perpframe[2].$$

This means that to get to a point $\{x(t), y(t)\}$ on the new, hung ellipse, we advance $x(t)$ units in the direction of $perpframe[1]$ and then $y(t)$ units in the direction of $perpframe[2]$.

This can be explained in terms of matrix operations. Every point $\{x(t), y(t)\}$ of the original ellipse is multiplied by the hanger matrix, $hanger(s)$, resulting in the new hung ellipse

$$hungellipse(t) = \begin{pmatrix} Cos(s) & Cos(s+\frac{\pi}{2}) \\ Sin(s) & Sin(s+\frac{\pi}{2}) \end{pmatrix} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}.$$

The result is shown in Figure 2.

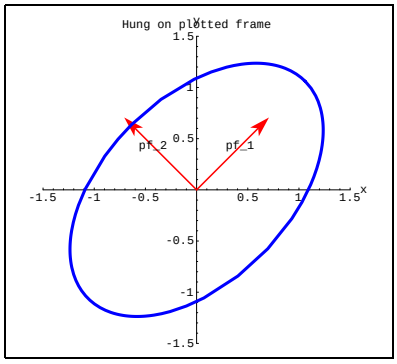

Figure 2: The ellipse "hung" on a perpendicular frame.

That is, $perpframe[1]$ is playing the former role of {1,0} and $perpframe[2]$ is playing the former role of {0,1}. In summary, multiplying the parametric form of a curve by a hanger matrix results in the curve being hung on the perpendicular frame defining this matrix.

Some thought also reveals that *rotation* matrices are hanger matrices coming from right hand perpendicular frames; by contrast, hanger matrices coming from left hand perpendicular frames are *not* rotations—they incorporate a flip.

Moreover, to *undo* the "hanging" of a curve on a perpendicular frame we have to multiply the new curve with the inverse of the hanger matrix. Pure algebraic operations reveal that the inverse of the hanger matrix is its transpose!

1.2 Aligner matrices for aligning a curve on the x-y axes

Given the perpendicular frame $perpframe[1] = \{Cos(s), Sin(s)\}$ and $perpframe[2] = \{Cos(s+\frac{\pi}{2}), Sin(s+\frac{\pi}{2})\}$, the aligner matrix defined by it is:

$$aligner(s) = \begin{pmatrix} Cos(s) & Sin(s) \\ Cos(s+\frac{\pi}{2}) & Sin(s+\frac{\pi}{2}) \end{pmatrix},$$

where $perpframe[1]$ and $perpframe[2]$ are the two rows. To see its action on a curve, suppose we are given the same perpendicular frame as above (i.e., $s = \frac{\pi}{4}$) and the ellipse hung on it as shown in Figure 2. The task is to align the ellipse to the $x - y$ axes.

We know that the parametric equation of the hung ellipse is

$$hungellipse(t) = x(t)perpframe[1] + y(t)perpframe[2].$$

Resolve now $\{x(t), y(t)\}$ into components in the directions of $perpframe[1]$ and $perp - frame[2]$ using the fact that the projection of any vector $\mathbf{x}$ on another vector $\mathbf{y}$ is $\frac{\mathbf{x}.\mathbf{y}}{\mathbf{y}.\mathbf{y}}\mathbf{y}$. Then, the parametric equation of the hung ellipse becomes:

$$\begin{aligned} hungellipse(t) = (\{x(t), y(t)\} \cdot perpframe[1])perpframe[1] \\ +(\{x(t), y(t)\} \cdot perpframe[2])perpframe[2]. \end{aligned}$$

Replacing now the last $perpframe[1]$ by $\{1, 0\}$ and the last $perpframe[2]$ by $\{0, 1\}$, we obtain the ellipse alligned to the $x - y$ axes:

$$\begin{aligned} alignedellipse(t) &= (\{x(t), y(t)\} \cdot perpframe[1])\{1, 0\} + (\{x(t), y(t)\} \cdot perpframe[2])\{0, 1\} \\ &= \{\{x(t), y(t)\} \cdot perpframe[1], \{x(t), y(t)\} \cdot perpframe[2]\}. \end{aligned}$$

The last expression is equivalent to :

$$alignedellipse(t) = \begin{pmatrix} Cos(s) & Sin(s) \\ Cos(s+\frac{\pi}{2}) & Sin(s+\frac{\pi}{2}) \end{pmatrix} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}.$$

In summary, multiplying the parametric form of a curve hung on a perpendicular frame by an aligner matrix defined by this frame results in the curve being aligned with the $x - y$ axes (Figure 1).

Provided we are dealing with the same perpendicular frame, note that the aligner matrix (which is the transpose of the hanger matrix) *undoes* what the hanger matrix does. That is, one is the inverse of the other:

$$\underbrace{\begin{pmatrix} Cos(s) & Cos(s+\frac{\pi}{2}) \\ Sin(s) & Sin(s+\frac{\pi}{2}) \end{pmatrix}}_{hanger(s)} \cdot \underbrace{\begin{pmatrix} Cos(s) & Sin(s) \\ Cos(s+\frac{\pi}{2}) & Sin(s+\frac{\pi}{2}) \end{pmatrix}}_{aligner(s)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

1.3 Diagonal matrices—the $x - y$ stretcher

Another name for the $2D$ diagonal matrix is xy-stretcher. The reason for this other name is that when we multiply a curve by a diagonal matrix all measurements along the x and y axes are stretched out. As an example consider the diagonal matrix

$$d = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix},$$

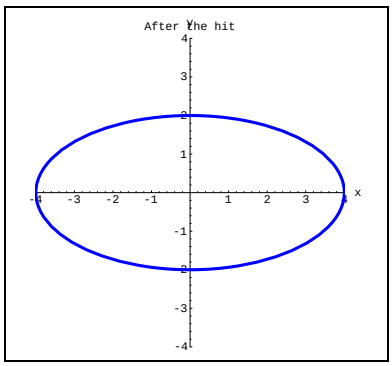

Figure 3: The unit circle *after* it has been multiplied by the diagonal matrix d above.

which multiplies every point of the unit circle given in parametric form. The result is shown in Figure 3, where it can be seen that all measurements along the x-axis have been stretched by a factor of 4 (the $x_{stretch}$ factor) and all measurements along the y-axis by a factor of 2 (the $y_{stretch}$ factor).

To invert the stretcher matrix d, we have to undo what multiplication by d did; that is, we have to shrink all measurements along the x-axis by a factor of 4 and all measurements along the y-axis by a factor of 2. So, the inverse of d is the easily computable matrix

$$d^{-1} = \begin{pmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}.$$

1.4 SVD analysis of $2D$ matrices

SVD analysis tells us that every matrix can be duplicated with the staple ingredients: hanger, stretcher and aligner. To show how this is done, consider a $2D$ matrix

$$m = \begin{pmatrix} -0.155468 & 0.427949 \\ 1.61437 & 1.03518 \end{pmatrix}.$$

Denote by $\{alignerframe[1], alignerframe[2]\}$ the aligner frame, by $x_{stretch}$, and $y_{stretch}$ the stretch factors and by $\{hangerframe[1], hangerframe[2]\}^T$ the hanger frame. Note that by simple matrix operations it is easily verified that if

$$m = hanger(s_1) \cdot stretcher \cdot aligner(s_2)$$

then

$$m \cdot alignerframe[1] = x_{stretch} hangerframe[1]$$

and

$$m \cdot alignerframe[2] = y_{stretch} hangerframe[2].$$

The above formulae subsume the eigenvectors/eigenvalues formulae, where for example we have

$$m \cdot alignerframe[1] = x_{stretch} alignerframe[1].$$

To duplicate this matrix m means coming up with an aligner frame, stretch factors and a hanger frame so that

$$m = hanger(s_1) \cdot stretcher \cdot aligner(s_2).$$

First set

$$\{alignerframe[1], alignerframe[2]\} = \{\{Cos(s), Sin(s)\}, \{Cos(s + \frac{\pi}{2}), Sin(s + \frac{\pi}{2})\}\}$$

for the two vectors of the aligner frame and then compute an s that makes

$$(m \cdot alignerframe[1]) \cdot (m \cdot alignerframe[2]) = 0.$$

The latter simply states that the aligner frame is a perpendicular frame. In our case $s = 0.582922$ radians, and the aligner frame becomes

$$\{alignerframe[1], alignerframe[2]\} = \{\{0.834858, 0.550466\}, \{-0.550466, 0.834858\}\}.$$

In matrix form we have

$$aligner(s_2) = \begin{pmatrix} 0.834858 & 0.550466 \\ -0.550466 & 0.834858 \end{pmatrix}.$$

Using the fact that the vectors of the aligner frame are of length 1 and perpendicular, we next compute

$$x_{stretch} = \|m \cdot alignerframe[1]\| = 1.92052$$

and

$$y_{stretch} = \|m \cdot alignerframe[2]\| = 0.44353,$$

from which we see that the diagonal matrix is

$$stretcher = \begin{pmatrix} 1.92052 & 0 \\ 0 & 0.44353 \end{pmatrix}.$$

Finally, the vectors of the hanger frame are computed from the equations

$$hangerframe[1] = \frac{1}{x_{stretch}} m \cdot alignerframe[1] = \{0.0550775, 0.998482\}$$

and

$$hangerframe[2] = \frac{1}{y_{stretch}} m \cdot alignerframe[2] = \{0.998482, -0.0550775\}.$$

In matrix form the hanger matrix is

$$hanger(s_1) = \begin{pmatrix} 0.0550775 & 0.998482 \\ 0.998482 & -0.0550775 \end{pmatrix},$$

and we need to note that there is a new angle corresponding to the hanger frame, namely $s_1 = 1.51569$ radians!

It is easily verified that

$$m = hanger(s_1) \cdot stretcher \cdot aligner(s_2)$$

as was expected. We also mention in passing that $s_1 = s_2$ only for *symmetrical* matrices, in which case

$$aligner frame[1] = hanger frame[1] = eigenvector[1],$$

$$aligner frame[2] = hanger frame[2] = eigenvector[2]$$

and

$$x_{stretch} = eigenvalue[1],$$
$$y_{stretch} = eigenvalue[2].$$

2 SVD Applications in Education

As we mentioned in the abstract, the teaching of linear algebra concepts is greatly enhanced and facilitated with the use of SVD analysis and Computer Algebra systems. Below we see how to explain the action of a matrix on a curve, when the inverse of a matrix does not exist—and what this means geometrically, and the relation between the inverse and the transform of a matrix.

2.1 Action of a matrix on a curve

Consider the same matrix m analyzed in section 1.4, that is

$$m = \begin{pmatrix} -0.155468 & 0.427949 \\ 1.61437 & 1.03518 \end{pmatrix},$$

and let us see the result of multiplying a curve times this matrix. For a curve, consider the unit circle shown in Figure 4:

$$\{x(t), y(t)\} = \{Cos(t), Sin(t)\}, \quad (0 \leq t \leq 2\pi).$$

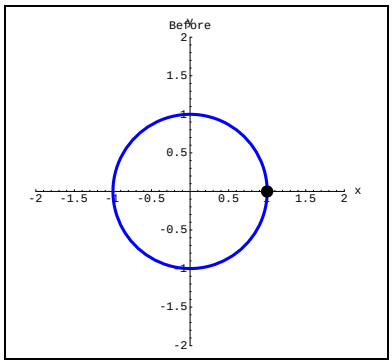

Figure 4: The unit circle *before* it has been multiplied times the matrix m above.

The curve we obtain *after* we multiply the unit circle in Figure 4 times the matrix m is an ellipse(!) and is shown in Figure 5.

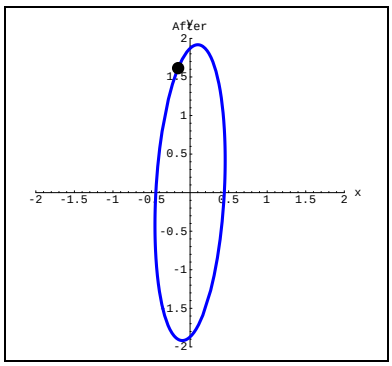

Figure 5: The unit circle became an ellipse *after* it was multiplied times the matrix m above.

An explanation for this phenomenon is given by the SVD analysis of m, which has been done in section 1.4. In that section we expressed

$$m = hanger(s_1) \cdot stretcher \cdot aligner(s_2).$$

We conclude that when we form the product $m \cdot \{x(t), y(t)\}$, it is the product $aligner(s_2) \cdot \{x(t), y(t)\}$ that is being performed first. The result of this multiplication is shown in Figure 6.

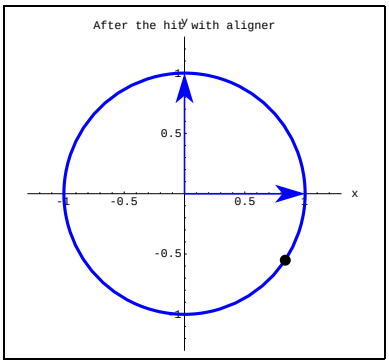

Figure 6: The unit circle *after* it was multiplied times the aligner matrix of m above.

Note how the aligner frame has been aligned with the $x - y$ axes and the dot—that was at the point $\{0, 0\}$—moved clockwise.

Next, it is the product $stretcher \cdot (aligner(s_2) \cdot \{x(t), y(t)\})$ that is being formed. The result is in Figure 7.

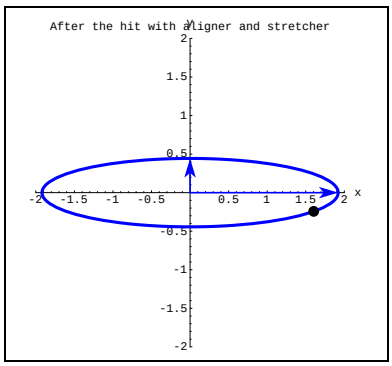

Figure 7: The unit circle *after* it was multiplied times the aligner and the stretcher matrices of m above.

Note how the measurements along the x-axis have grown by a factor of 1.92052 whereas the measurements along the y axis have shrunk, multiplied by 0.44353.

Finally, the product $hanger \cdot (stretcher \cdot (aligner(s_2) \cdot \{x(t), y(t)\}))$ is being formed. The result is shown in Figure 8.

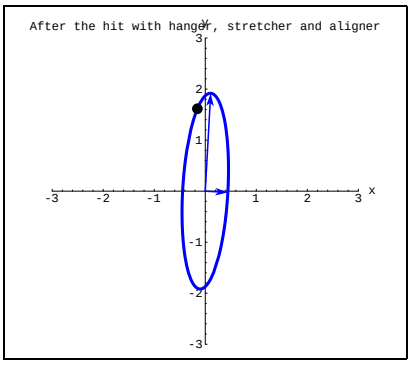

Figure 8: The unit circle *after* it was multiplied times the aligner, the stretcher and the hanger matrices of m above.

It is the end result, and the ellipse is hung on the hanger frame of matrix m. The final flip of the point is due to the hanger frame being a left hand perpendicular frame.

2.2 Matrices that cannot be inverted

In the matrix m we used before, let us change the stretcher matrix somewhat; namely, let us form the matrix

$$m_{new} = hanger(s_1) \cdot \begin{pmatrix} 1.92052 & 0 \\ 0 & 0 \end{pmatrix} \cdot aligner(s_2),$$

where the y-stretch factor is 0. As can be easily checked, the matrix m_{new} has no inverse, which means its determinant is 0. This leads to the alternative definition of the determinant of a matrix, as the product of the stretch factors. Multiplying now the unit circle times m_{new}, we obtain the straight line shown in Figure 9.

What happened here is that two points of the ellipse have been deposited on the same point on the line. Since we cannot tell which part of the ellipse they came from, it is impossible to undo the multiplication times the matrix m_{new}.

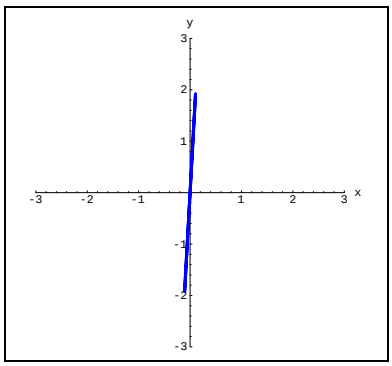

Figure 9: The unit circle *after* it was multiplied times the non invertible matrix m_{new} above.

It should be added that the inverse of the hanger and aligner matrices always exists since it is the transpose of the corresponding matrix. (If these matrices are orthogonal, then their pseudoinverse is the transpose of the corresponding matrix. Recall that for any matrix A we have $PseudoInverse(A) = (A^T A)^{-1} A^T$ [4].) It is only when the inverse of the stretcher matrix does not exist that the inverse of a matrix m does not exist as well.

3 SVD for solving linear least-squares problems

The least-squares problem arises when we try to fit the polynomial

$$f(x) = c_1 + c_2 x + c_3 x^2 + \cdots + c_n x^{n-1}$$

to some data points $\{(x_i, y_i)\},\ i = 1, \ldots, m$, where $n < m$. Compare this with interpolation where we choose the degree $n-1$ of the polynomial high enough to pass through all the points; to wit, $n = m$.

A further generalization of the linear least-squares problem is to take a linear combination of *basis functions* $\{f_1(x), f_2(x), \ldots, f_n(x)\}$:

$$f(x) = c_1 f_1(x) + c_2 f_2(x) + \cdots + c_n f_n(x),$$

but we will think of the basis functions $\{f_1(x), f_2(x), \ldots, f_n(x)\}$ as being the powers of x: $1, x, \ldots, x^{n-1}$.

Note that in our case ($n < m$) we have an *overdetermined* system of equations, i.e., there are more equations than there are unknowns. By forcing n of the equations to be exactly satisfied we may cause the others to exhibit large errors. Therefore, we would like to choose values that allow all of the equations to be approximately satisfied instead of forcing n of them to be exactly satisfied.

The best approximation f_{approx} to the given data by a linear combination of basis functions is the one for which the residuals $y_i - f_{approx}(x_i)$ are "smallest". Among the various possibilities for measuring the size of the residuals we take the quantity

$$\sqrt{\sum_{i=1}^{m} |y_i - f(x_i)|^2}.$$

The problem of fitting a polynomial of degree $n-1$ can be written as

$$\begin{array}{ccccccccc} c_1 & + & c_2x_1 & + & \cdots & + & c_nx_1^{n-1} & \approx & y_1, \\ c_1 & + & c_2x_2 & + & \cdots & + & c_nx_2^{n-1} & \approx & y_2, \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ c_1 & + & c_2x_m & + & \cdots & + & c_nx_m^{n-1} & \approx & y_m, \end{array}$$

which denotes an overdetermined system of linear equations because $m > n$. Also recall that the x_i and y_i are given and that the c_j are the unknowns.

In matrix form we have $A\mathbf{c} \approx \mathbf{y}$, where A is a nonsquare "tall" matrix, the unknown $\mathbf{c}$ is a "short" vector, and $\mathbf{y}$ is a "tall" vector:

$$A = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^{n-1} \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_m \end{pmatrix}.$$

The residual vector is

$$\mathbf{r} = \mathbf{y} - A\mathbf{c}.$$

Using the Euclidean norm,

$$\| \mathbf{u} \|_2 = \sqrt{\sum_{i=1}^{m} u_i^2},$$

the least-squares problem becomes

$$\min_{\mathbf{c}} \| \mathbf{y} - A\mathbf{c} \|_2 .$$

A known solution to the least-squares problem is the solution $\mathbf{c}$ to the linear system

$$A^TA\mathbf{c} = A^T\mathbf{y},$$

which are known as the *normal equations*. A less known solution to the least-squares problem is obtained using SVD analysis of A as shown below.

We have $A = U^T\Sigma V$ and

$$\| \mathbf{y} - A\mathbf{c} \|_2 = \| \mathbf{y} - U^T\Sigma V\mathbf{c} \|_2 = \| U(\mathbf{y} - U^T\Sigma V\mathbf{c}) \|_2 = \| U\mathbf{y} - \Sigma V\mathbf{c} \|_2,$$

since $UU^T = I_{rank(A)}$.

Denoting $U\mathbf{y}$ by $\mathbf{d}$, $V\mathbf{c}$ by $\mathbf{z}$, and $rank(A)$ by r, we have

$$\| \mathbf{y} - A\mathbf{c} \|_2^2 = \| \mathbf{d} - \Sigma\mathbf{z} \|_2^2 = \left\| \begin{pmatrix} d_1 - \sigma_1 z_1 \\ d_2 - \sigma_2 z_2 \\ \vdots \\ d_r - \sigma_r z_r \end{pmatrix} \right\|_2^2 = (d_1 - \sigma_1 z_1)^2 + \cdots + (d_r - \sigma_r z_r)^2.$$

We can now uniquely select the $z_i = \frac{d_i}{\sigma_i}$, for $i = 1, \ldots, rank(A)$ to reduce this expression to its minimum value. The solution to the least-squares problem is then obtained from $V\mathbf{c} = \mathbf{z}$:

$$\mathbf{c} = PseudoInverse(V)\mathbf{z} = V^T\mathbf{z}.$$

Example: The following data depicts the growth of the population (in millions) in the United States for the years 1900 through 1970

$$\begin{pmatrix} 1900 & 1910 & 1920 & 1930 & 1940 & 1950 & 19960 & 1970 \\ 75.99 & 91.97 & 105.71 & 122.75 & 131.67 & 150.69 & 179.32 & 203.21 \end{pmatrix}.$$

We want to fit a polynomial of degree 2 to these points. To wit, the polynomial $f(x) = c_1 + c_2x + c_3x^2$ should be such that $f(x_i)$ should be as close as possible to y_i, where the points $\{x_i, y_i\}$, $i = 1, 2, \ldots, 8$ represent, respectively, the year (x_i) and the population (y_i).

Our system of equations $A\mathbf{c} = \mathbf{y}$ is

$$\begin{pmatrix} 1 & 1900 & 3610000 \\ 1 & 1910 & 3648100 \\ 1 & 1920 & 3686400 \\ 1 & 1930 & 3724900 \\ 1 & 1940 & 3763600 \\ 1 & 1950 & 3802500 \\ 1 & 1960 & 3841600 \\ 1 & 1970 & 3880900 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 75.99 \\ 91.97 \\ 105.71 \\ 122.75 \\ 131.67 \\ 150.69 \\ 179.32 \\ 203.21 \end{pmatrix}.$$

The SVD analysis of the matrix $A = U^T\Sigma V$ is obtained with the help of *Mathematica* and is shown below:

$$U = \begin{pmatrix} -0.341 & -0.344 & -0.348 & -0.352 & -0.355 & -0.359 & -0.363 & -0.366 \\ -0.543 & -0.393 & -0.24 & -0.089 & 0.065 & 0.221 & 0.378 & 0.537 \\ 0.546 & 0.081 & -0.229 & -0.385 & -0.386 & -0.234 & 0.073 & 0.534 \end{pmatrix},$$

$$\Sigma = \begin{pmatrix} 1.05947 \cdot 10^7 & 0 & 0 \\ 0 & 64.7746 & 0 \\ 0 & 0 & 0.000346202 \end{pmatrix},$$

$$V = \begin{pmatrix} -2.66890 \cdot 10^{-7} & -0.000516579 & -1. \\ -0.00103368 & -0.999999 & 0.000516579 \\ 0.999999 & -0.00103368 & 2.67087 \cdot 10^{-7} \end{pmatrix}.$$

We set $U\mathbf{y} = \mathbf{d} = \{-377.813, 104.728, 13.098\}^T$ and compute

$$\mathbf{z} = \{-0.0000356605, 1.61681, 37833.1\}^T$$

by dividing $\frac{d_i}{\sigma_i}$, $i = 1, 2, 3$. (We will not concern ourselves with analyzing the condition number of this matrix—and there something to be said about it here.)

The solution to the least-squares problem is then obtained from $V\mathbf{c} = \mathbf{z}$:

$$\mathbf{c} = V^T\mathbf{z} = \{37833.1, -40.7242, 0.0109756\}^T,$$

and is the same obtained using the function **Fit** in *Mathematica*. (See [5] for a discussion on the problems of this function and on robust regression.)

4 SVD and data compression

In this section, we will use the second form of the singular-value decomposition, namely:

$$A = U^T \Sigma V = \sum_{i=1}^{rank(A)} \sigma_i u_i^T v_i,$$

where the matrix $u_i^T v_i$ is the *outer product* of the i-th row of U with the corresponding row of V. Since each of these matrices can be stored using only $m + n$ locations rather than mn locations, this form is suitable for applications in data compression.

Data compression will be used on the following problem from [3], eloquently solved in [1]. "Suppose that a satellite in space is taking photographs of Jupiter to be sent back to earth. The satellite digitizes the picture by subdividing it into tiny squares called *pixels* or picture elements. Each pixel is represented by a single number that records the average light intensity in that square. If each photograph were divided into 500×500 pixels, it would have to send $250,000$ numbers to earth for each picture. (This amounts to a $500D$ matrix.) This would take a great deal of time and would limit the number of photographs that could be transmitted. It is (sometimes) possible to approximate this matrix with a 'simpler' matrix which requires less storage."

Let us start by looking at a "picture" of only 4 pixels in Figure 10. The gray level specification for each square is determined by the corresponding entry in the matrix A

$$\begin{pmatrix} 0.748611 & 0.407564 \\ 0.263696 & 0.469433 \end{pmatrix}.$$

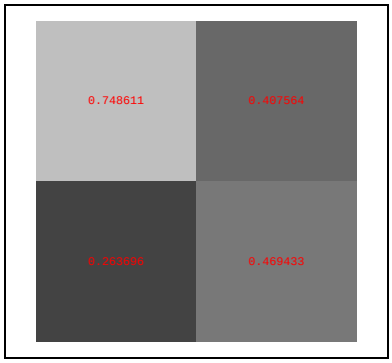

Figure 10: Picture based on a $2D$ matrix whose entries are numbers between 0.0 and 1.0.

Figure 11 depicts the graphic generated by a $16D$ matrix A whose entries are numbers between 0.0 and 1.0 (A itself is too large to be displayed!)

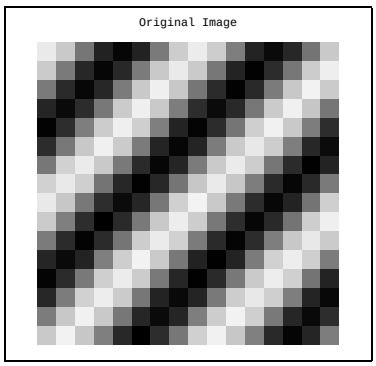

Figure 11: Picture based on a $16D$ matrix whose entries are numbers between 0.0 and 1.0.

To reproduce this picture exactly, we need to keep track of all of the 256 ($= 16 \times 16$) entries of A. However, using SVD analysis of A, we can approximately reproduce this image keeping track of much less data. To compress the data, we set the smaller singular values to zero. The singular values of the $16D$ matrix A are

$$\begin{array}{rrrr} 7.63116 & 3.64554 & 3.62529 & 0.089207 \\ 0.081537 & 0.0660488 & 0.0553387 & 0.0487359 \\ 0.0466094 & 0.0340483 & 0.0314476 & 0.0261613 \\ 0.0187258 & 0.0133512 & 0.00755902 & 0.00556133 \end{array}.$$

Using only the first 3 singular values of A (the ones > 1), we have

$$A = \sum_{i=1}^{16} \sigma_i u_i^T v_i \approx \sum_{i=1}^{3} \sigma_i u_i^T v_i,$$

and the compressed image is shown in Figure 12.

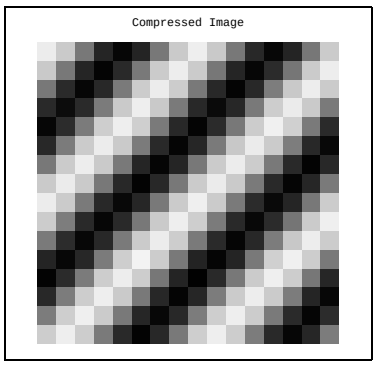

Figure 12: Picture generated using only 3 singular values (and their corresponding rows of U and V) of the $16D$ matrix above.

The satellite transmits the following data for the compressed image: the first 3 rows of U, the first 3 singular values and the first 3 rows of V. And then the recipient on earth forms the $16D$ matrix

$$\sum_{i=1}^{3} \sigma_i u_i^T v_i.$$

To wit, the satellite transmits $3 \times 16 + 3 + 3 \times 16 = 99$ elements. This is a lot less data than the 256 elements that would have to be transmitted otherwise in our case.

5 Conclusions

The singular value decomposition is over a hundred years old. For the case of square matrices, it was discovered independently by Beltrami in 1873 and Jordan in 1874. The technique was extended to rectangular matrices by Eckart and Young in the 1930's and its use as a computational tool dates back to the 1960's. Golub and van Loan [2] demonstrated its usefulness and feasibility in a wide variety of applications.

Despite its history, singular value decomposition of matrices is still not widely used in education. The book by Davis and Uhl [1] is the only one known to the authors that—together with the Computer Algebra system *Mathematica*—bases an entire linear algebra course on this concept. Hopefully, our presentation has contributed something to the effort to acquaint university teachers with the beauty of this subject.

6 Acknowledgements

We would like to sincerely thank one referee for the detailed comments which greatly improved the presentation of this paper.

References

[1] Bill Davis and Jerry Uhl, *Matrices, Geometry & Mathematica*, Math Everywhere, Inc., 1999 (Part of the "Calculus & *Mathematica*" series of books).

[2] Gene H. Golub and Charles F. van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, MD, 1983.

[3] David Kahaner, Cleve Moler and Stephen Nash, *Numerical Methods and Software*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

[4] Robert D. Skeel and Jerry B. Keiper, *Elementary Numerical Computing with Mathematica*, McGraw-Hill, New York, N.Y., 1993.

[5] William T. Shaw and Jason Tigg, *Applied Mathematica: Getting Started, Getting it Done*, Addison-Wesley, Reading MA., 1994