Revision Notes on Linear Algebra for Undergraduate Engineers

Pete Bunch

Lent Term 2012

1 Introduction

A matrix is more than just a grid full of numbers. Whatever sort of engineering you specialise in, a basic grounding in linear algebra is likely to be useful. Lets look at some examples of how matrix equations arise.

Add examples from structures (truss), electrical (resistor grid), mechanics (kinematics), fluids (some sort of pressure, height, velocity-square problem)

All of these problems lead us to the same equation,

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

We can think of x as a position vectors in an "input space" or "solution space" and b as the corresponding position vector in the "output space" or "constraint space" respectively. The coordinates in these spaces are just the values of the individual components of x and b. A is the linear transformation that takes us from one space to the other.

2 Fundamental Matrix Sub-Spaces

Consider an $m \times n$ matrix A, which maps a point x in the "input space" to y in the "output space". i.e.

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{2}$$

What can we say about the input and output spaces? Firstly, x must be $n \times 1$ so the input space is n-dimensional (nD). y must be $m \times 1$ so the output space is m-dimensional (mD).

2.1 Output Side

Let's consider the output space. We can write A as a set of column vectors.

$$\mathbf{A} = \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \dots & \mathbf{c}_n \end{bmatrix} \tag{3}$$

Now when we multiply A by x, we can write it as,

$$\mathbf{y} = \mathbf{A}\mathbf{x} = x_1\mathbf{c}_1 + x_2\mathbf{c}_2 + \cdots + x_n\mathbf{c}_n. \tag{4}$$

So the output, y, is just a weighted sum of the columns of A. This set of column vectors defines a new vector subspace, the "column space", which contains all the valid y's for the equation $\mathbf{y} = \mathbf{A}\mathbf{x}$.

The output space is always mD, but the column space won't always "fill" the whole of it. If there are only r independent columns of A, then the column space is an rD subspace. For example, if the matrix has 3 rows, but the three column vectors are co-planar, then the column space is a plane in a 3D space. r is called the *rank* of the matrix.

Note that r cannot be larger than n (i.e. if all the columns are independent), so if n is smaller than m, then the column space will definitely not fill the entire output space. In the parlance of simultaneous equations, this is because you have more equations than variables.

If the column space doesn't fill the whole output space, what happens in the rest of it? We can split our output point up into a component in the column space and a perpendicular component that isn't,

$$\mathbf{y} = \mathbf{y}_C + \mathbf{y}_{LN} \tag{5}$$

The perpendicular component lies in the orthogonal subspace to the column space. This is called the "left-nullspace". (Its got to be orthogonal or we could write it (or part of it) in terms of column space vectors, which would mean that it (or part of it) was actually in the column space.) If we're in an mD output space, and the column space is rD, then there are $m-r$ directions perpendicular to the column space, so the left-nullspace is $(m-r)$D.

So, for our equation $\mathbf{y} = \mathbf{A}\mathbf{x}$ to have a solution, we must have $\mathbf{y}_{LN} = 0$. What does this mean? Well, for our structures example, a component in the left-nullspace represents a violation of the geometry constraints, i.e. one of the bars must have changed length. In the electric example, it means that Kirchoff's laws have been broken, and in the mechanics example it means Newton's laws have been broken. Left-nullspace outputs are impossible.

2.2 Input Side

For the input space, we're going to expand A in a different way, in terms of its rows.

$$\mathbf{A} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \vdots \\ \mathbf{r}_m^T \end{bmatrix} \tag{6}$$

Now we can write the multiplication like this,

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} \mathbf{r}_1 \cdot \mathbf{x} \\ \mathbf{r}_2 \cdot \mathbf{x} \\ \vdots \\ \mathbf{r}_m \cdot \mathbf{x} \end{bmatrix}. \tag{7}$$

Taking the dot product of the input point with each of the row vectors gives us the "contribution" of that point in each of the row directions. This defines another interesting subspace, the "row space", which contains all the useful

inputs which have an effect on our system. As before, we can write each input point in terms of a component in the row space and a perpendicular component,

$$\mathbf{x} = \mathbf{x}_R + \mathbf{x}_N \tag{8}$$

Now, there's nothing wrong with having an $\mathbf{x}_N$ component. We're talking about the input to the system here, and we can choose that to be whatever we damn-well please. However, if $\mathbf{x}_N$ is perpendicular to the row space component, then it must also be perpendicular to all the individual rows of A. Perpendicular means that the dot product is zero, which leads us to the fact that,

$$\mathbf{A}\mathbf{x}_N = \mathbf{0}. \tag{9}$$

We call this orthogonal subspace the "nullspace". As we can see, an input (or a part of it) in the nullspace has no effect on the output.

The input space is always nD, but the row space won't always fill the whole of it. The number of row space dimensions is given by the number of independent row vectors. Curiously, this is always r (i.e. the rank, the same as the column space). We'll see why when we do LU decompositions. The nullspace then has $(n - r)$ dimensions.

2.3 Transposing

If we transpose the matrix, then everything flips around. The row space of A is the column space of AT, and vice-versa. The nullspace of A is the left-nullspace of AT and vice-versa. This should be pretty obvious, because the transpose operation turns the rows into columns and the columns into rows. It also gives us a nice way to define the left-nullspace, as the nullspace of AT. This means that $\mathbf{A}^T\mathbf{y}_{LN} = \mathbf{0}$.

2.4 Summary

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{10}$$

- x is in the input space, which is divided into the *row space* and the *nullspace*. The nullspace maps onto $\mathbf{0}$ in the output space. The row space maps onto the column space.
- y is in the output space, which is divided into the *column space* and the *left-nullspace*. If y is in the column space then there is a solution for x (in the row space). If y has a component in the left-nullspace, there is no solution for x.

3 Solving Matrix Equations

We're now going to consider solving equations of the form,

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{11}$$

where A and y are known and fixed and we want to find x. Back in the simple land of scalars, there was always 1 solution to this equation (unless A

was 0). Here in multi-dimensional space there could be one, zero, or an infinite number of solution. We've already considered when no solution exists — its when $\mathbf{y}_{LN} \neq 0$.

It turns out that there will be one solution if $r = n$, and many (infinite) if $r < n$. To see why, lets think about the nullspace. If $r < n$, then the matrix has a nullspace (because it has $n - r > 0$ dimensions), and it's defined by $\mathbf{A}\mathbf{x}_N = \mathbf{0}$. If $\mathbf{x}_0$ is a solution to the equation, i.e. $\mathbf{y} = \mathbf{A}\mathbf{x}_0$, then we could just add some of $\mathbf{x}_N$ onto the original solution and we'd still have a valid solution.

$$\begin{aligned} \mathbf{A}(\mathbf{x}_0 + \lambda\mathbf{x}_N) &= \mathbf{A}\mathbf{x}_0 + \lambda\mathbf{A}\mathbf{x}_N & (12) \\ &= \mathbf{y} + \mathbf{0} & (13) \end{aligned}$$

So there will be an infinite number of solutions when the nullspace exists. Otherwise, there's just one.

We'll get onto how we actually find $\mathbf{x}_0$ and $\mathbf{x}_N$ when we look at decompositions.

3.1 Least Squares Solutions

We said that we can't solve a matrix equation when $\mathbf{y}_{LN} \neq 0$. However, all is not lost. We can still find a best-fit/minimum-error solution. Remember that the left-nullspace is characterised by $\mathbf{A}^T\mathbf{y}_{LN} = \mathbf{0}$. So if we pre-multiply the output (i.e. the constraints) of the equation by A^T, then we can make that awkward left-nullspace bit disappear.

$$\begin{aligned} \mathbf{A}^T(\mathbf{y}_C + \mathbf{y}_{LN}) &= \mathbf{A}^T\mathbf{y}_C + \mathbf{A}^T\mathbf{y}_{LN} & (14) \\ &= \mathbf{A}^T\mathbf{y}_C + \mathbf{0} & (15) \end{aligned}$$

So if we take the original solution-less equation and pre-multiply by A^T, we get a new equation which does have a solution.

$$\mathbf{A}^T\mathbf{y} = \mathbf{A}^T\mathbf{A}\mathbf{x} \tag{16}$$

This solution results in an output of $\mathbf{y}_C$, which is as close as we can get to y.

An elegant connection can be made here. Consider that we're trying to find the value of x for which Ax is as close to y as possible. Hence, we want to minimise,

$$\begin{aligned} ||\mathbf{y} - \mathbf{A}\mathbf{x}||^2 &= (\mathbf{y} - \mathbf{A}\mathbf{x})^T(\mathbf{y} - \mathbf{A}\mathbf{x}) \\ &= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}. & (17) \end{aligned}$$

The minimum is found by differentiating this w.r.t. x and setting the result to 0. This give us

$$\begin{aligned} -2\mathbf{y}^T\mathbf{A} + 2\mathbf{x}^T\mathbf{A}^T\mathbf{A} &= 0 \\ \mathbf{x}^T\mathbf{A}^T\mathbf{A} &= \mathbf{y}^T\mathbf{A} \\ \mathbf{A}^T\mathbf{A}\mathbf{x} &= \mathbf{A}^T\mathbf{y} & (18) \end{aligned}$$

So the least squared error solution is the same as the column space-only solution, which is as expected. Good.

4 LU Decompositions

An LU decomposition (specifically, a Doolittle LU decomposition) is a factorisation of the form,

$$\underbrace{\begin{bmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix}}_{\mathbf{A}(m\times n)} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ . & 1 & 0 \\ . & . & 1 \end{bmatrix}}_{\mathbf{L}(m\times m)} \underbrace{\begin{bmatrix} . & . & . & . \\ 0 & . & . & . \\ 0 & 0 & . & . \end{bmatrix}}_{\mathbf{U}(m\times n)} \tag{19}$$

4.1 Row Space Interpretation

Lets reason our way to the LU decomposition by considering the row space. Our row space vectors of A are not very nice — they're all in arbitrary directions, which makes calculations difficult. We can try to modify them by tying them to the axes a bit. First we make the first row of A "responsible" for the first dimension (the x-axis). Otherwise, it remains unchanged. We then scale and add this x-row to the other rows to get rid of their x components (i.e. forcing the 0s in the first column of U). Next we make the (modified) second row responsible for the y-axis. We scale and add this y-row to the others to get get rid of their y components. Continue until we run out of rows or dimensions. The resulting modified rows make up our U matrix. L contains the scale factors required to "rebuild" the row of A from the rows of U.

$$\begin{bmatrix} \mathbf{u}_1 \\ l_{2,1}\mathbf{u}_1 + \mathbf{u}_2 \\ l_{3,1}\mathbf{u}_1 + l_{3,2}\mathbf{u}_2 + \mathbf{u}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{2,1} & 1 & 0 \\ l_{3,1} & l_{3,2} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix} \tag{20}$$

All we did to the rows of A to get the rows of U was scaling and adding, so we have not changed the vector space that they span. Hence, the row space of A is the row space of U.

What happens if the rows of U are not independent? We find that having attended to r of the rows, then when we do the scaling and adding to get rid of the first r components of the $(r+1)$th row, the entire row ends up being filled with zeros. This follows from the fact that the $(r+1)$th row is just a linear combination of the first r rows. Thus, the rank of the matrix is the number of non-zero rows of U. Of course, if m is larger than n, then there will always be at least one row of zeros in U.

$$\begin{bmatrix} \mathbf{u}_1 \\ l_{2,1}\mathbf{u}_1 + \mathbf{u}_2 \\ l_{3,1}\mathbf{u}_1 + l_{3,2}\mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{2,1} & 1 & 0 \\ l_{3,1} & l_{3,2} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{0} \end{bmatrix} \tag{21}$$

L is square and has determinant 1, so its always invertible. This makes it really easy to find the nullspace of A,

$$\begin{aligned} \mathbf{A}\mathbf{x}_N &= \mathbf{0} \\ \mathbf{L}\mathbf{U}\mathbf{x}_N &= \mathbf{0} \\ \mathbf{U}\mathbf{x}_N &= \mathbf{L}^{-1}\mathbf{0} = \mathbf{0}. \end{aligned} \tag{22}$$

4.2 Column Space interpretation

We can now turn the whole thing on its head and consider the column spaces. The columns of A are built up from the columns of L, using the factors of U as scale factors.

$$\begin{bmatrix} u_{1,1}\mathbf{l}_1 & u_{1,2}\mathbf{l}_1 + u_{2,2}\mathbf{l}_2 & u_{1,3}\mathbf{l}_1 + u_{2,3}\mathbf{l}_2 + u_{3,3}\mathbf{l}_3 & u_{1,4}\mathbf{l}_1 + u_{2,4}\mathbf{l}_2 + u_{3,4}\mathbf{l}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{l}_1 & \mathbf{l}_2 & \mathbf{l}_3 \end{bmatrix} \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ 0 & u_{2,2} & u_{2,3} & u_{2,4} \\ 0 & 0 & u_{3,3} & u_{3,4} \end{bmatrix} \quad (23)$$

The columns of A are just linear combinations of those of L, so we might expect the column space of A to be the same as the column space of L. However, if there is a row in U populated entirely with zeros, then we don't actually use the corresponding column of L when rebuilding A. So in fact the column space of A is spanned by the columns of L *corresponding to non-zero rows of U* (or in other words, the first r columns of L). This, incidently, is how we prove that the row and column spaces have the same number of dimensions.

The left-nullspace can be found from its definition (perpendicular to the column space) by finding vectors orthogonal to the first r columns of L.

4.3 Partial Pivoting

Remember when we "made the first row responsible for the first dimension". What if it doesn't have a component in that direction, or it does but its tiny? In these circumstances, we need to swap the rows around. This can be achieved by premultiplying A by a permutation matrix, so we now have,

$$\mathbf{PA} = \mathbf{LU}. \quad (24)$$

$\mathbf{P}$ has one 1 in each row and each column. All other components are 0. It is orthogonal, i.e. $\mathbf{P}^{-1} = \mathbf{P}^T$.

4.4 Solving Equations

Recall from earlier that we need methods for finding $\mathbf{x}_0$ and $\mathbf{x}_N$ in order to find the general solution to $\mathbf{y} = \mathbf{Ax}$. Once we have an LU decomposition, these are easy. For $\mathbf{x}_0$, solve

$$\mathbf{y} = \mathbf{Lz} \quad (25)$$

$$\mathbf{z} = \mathbf{Ux} \quad (26)$$

These are both easy because of the triangular structure of L and U. The second equation is likely to be underspecified (fewer equations than unknowns). We only need one solution, so some of the variables can simply be fixed (to 0 is easiest).

For $\mathbf{x}_N$, just use,

$$\mathbf{Ux}_N = \mathbf{0}, \quad (27)$$

from the definition. Again, there will probably be multiple solutions, so some variables must be set to arbitrary (non-zero) values.

5 QR Decompositions

QR decompositions can be a bit confusing because there are two forms, referred to here as full and reduced.

$$\begin{aligned} \mathbf{A} &= \mathbf{QR} \\ &= \begin{bmatrix} \mathbf{Q}_C & \mathbf{Q}_{LN} \end{bmatrix} \begin{bmatrix} \mathbf{R}_R \\ \mathbf{0}_{(m-r\times n)} \end{bmatrix} \qquad (28) \\ &= \mathbf{Q}_C\mathbf{R}_R \qquad (29) \end{aligned}$$

$\mathbf{Q}_C$ is $m \times r$, and $\mathbf{Q}_{LN}$ is $m \times m - r$, making $\mathbf{Q}$ a square, $m \times m$ matrix. $\mathbf{R}_R$ is $r \times n$. For some problems and proofs we will need the full form, but most of the time we can get away with using the reduced variety, which requires less calculation. The reduced form looks like this,

$$\underbrace{\begin{bmatrix} . & . & . \\ . & . & . \\ . & . & . \\ . & . & . \end{bmatrix}}_{\mathbf{A}(m\times n)} = \underbrace{\begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 \end{bmatrix}}_{\mathbf{Q}_C(m\times r)} \underbrace{\begin{bmatrix} . & . & . \\ 0 & . & . \\ 0 & 0 & . \end{bmatrix}}_{\mathbf{R}_R(r\times n)}, \qquad (30)$$

where we make the q_is an orthonormal basis (i.e. orthogonal unit vectors) for the column space of A. This will be really useful for solving equations.

5.1 Column Space Interpretation

With the LU decomposition, we manipulated the row space vectors of A to make them a bit nicer, i.e. to force all but one vector onto the $x = 0$ plane, all but two vectors onto the $y = 0$ plane, etc. For a QR decomposition, we are going to manipulate the column space vectors to make them orthonormal. We can do this using a procedure called Gram-Schmidt orthogonalisation (sounds painful). This is not complex, but it is tedious.

1: **for** $i = 1$ **to** n **do**
2: **for** $j = 1$ **to** $i - 1$ **do**
3: Find $\tilde{r}_{j,i} = \mathbf{a}_i \cdot \mathbf{q}_j$, the projection of the ith column of A in the direction of the jth column of Q_C.
4: **end for**
5: Subtract the projections from $\mathbf{a}_i$, $\tilde{\mathbf{q}}_i = \mathbf{a}_i - \sum_j \tilde{r}_{j,i}\mathbf{q}_j$. What remains is perpendicular to all previous columns.
6: Find the magnitude, $s_i = |\tilde{\mathbf{q}}_i|$.
7: Normalise the new column, $\mathbf{q}_i = \tilde{\mathbf{q}}_i / s_i$.
8: **end for**

The values in the R_R matrix are just those required to "rebuild" the columns of A from the columns of Q_C. Hence,

$$\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3\end{bmatrix} = \begin{bmatrix}\mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3\end{bmatrix} \begin{bmatrix} s_1 & \tilde{r}_{1,2}s_2 & \tilde{r}_{1,3}s_3 \\ 0 & s_2 & \tilde{r}_{2,3}s_3 \\ 0 & 0 & s_3 \end{bmatrix}, \tag{31}$$

If the columns of A are not all independent, then we will find that we get $\tilde{\mathbf{q}}_{r+1} = \mathbf{0}$ in our orthogonalisation procedure. In this case, we can just throw away that column and the corresponding row in R_R — we do not need it for the orthonormal basis. This does not affect the column space spanned because the discarded column was a linear combination of the others. The Q_C matrix becomes $m \times r$ and the R_R matrix $r \times n$.

Because the orthogonalisation is just a set of linear operations on the columns of A, the column space of A is the same as that of Q_C. By switching to a row space interpretation (like that of the LU decomposition), we can reason that the row space and nullspace of A are the same as those of R_R.

The columns of matrix Q_C are orthogonal and unit length. This means that,

$$\begin{aligned} \mathbf{Q}_C^T\mathbf{Q}_C &= \begin{bmatrix}\mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \mathbf{q}_3^T\end{bmatrix} \begin{bmatrix}\mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{q}_1\cdot\mathbf{q}_1 & \mathbf{q}_1\cdot\mathbf{q}_2 & \mathbf{q}_1\cdot\mathbf{q}_3 \\ \mathbf{q}_2\cdot\mathbf{q}_1 & \mathbf{q}_2\cdot\mathbf{q}_2 & \mathbf{q}_2\cdot\mathbf{q}_3 \\ \mathbf{q}_3\cdot\mathbf{q}_1 & \mathbf{q}_3\cdot\mathbf{q}_2 & \mathbf{q}_3\cdot\mathbf{q}_3\end{bmatrix} \\ &= \mathbf{I}. \end{aligned} \tag{32}$$

However, the reverse is only true if the matrix is square! (i.e. in general $\mathbf{Q}_C\mathbf{Q}_C^T \neq \mathbf{I}$, but see next section)

5.2 Full Form

To compute the full form of the QR decomposition, we find an additional $m-r$ vectors which form an orthonormal basis for the left-nullspace. We concatenate these to Q_C in order to make Q a square, full rank, orthogonal matrix. Note that this means the column space of Q is no longer the same as the column space of A. We add zeros in the rows of R corresponding to these new columns. Q is now a proper orthogonal matrix ($\mathbf{Q}^T = \mathbf{Q}^{-1}$), which is useful, but this requires a whole lot more calculation of orthogonal vectors, so don't do it by hand. MATLAB uses this form.

5.3 Solving Equations

The orthogonal property of the Q matrix is super-useful for solving equations. If an exact solution exists, then we can find it using,

$$\mathbf{y} = \mathbf{Q}\mathbf{z} \tag{33}$$

$$\mathbf{z} = \mathbf{R}\mathbf{x} \tag{34}$$

This is easy if we use the full form decomposition, because we can invert Q simply by taking a transpose. The second equation is simple because R is triangular. With reduced form equations, it is non-trivial.

Least squares problems are even easier using QR decompositions. Using reduced form, the column space (and thus the left-nullspace) of Q_C is the same as that for A. This means we can use the left-nullspace trick with Q_C instead of A, i.e. $\mathbf{Q}_C^T(\mathbf{y}_C + \mathbf{y}_{LN}) = \mathbf{Q}_C^T\mathbf{y}_C$. Hence we can find a least squares solution by solving,

$$\begin{aligned} \mathbf{y} &= \mathbf{A}\mathbf{x} \\ \mathbf{Q}_C^T\mathbf{y} &= \mathbf{Q}_C^T\mathbf{Q}_C\mathbf{R}_R\mathbf{x} \\ \mathbf{Q}_C^T\mathbf{y} &= \mathbf{R}_R\mathbf{x}. \end{aligned} \tag{35}$$

Again, this will be easy because R_R is triangular.

From here on, there are quite a lot of unproved statements.

6 Eigenvalues

Eigenvalues and vectors are properties of square matrices only, specified jointly by the equation,

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}. \tag{36}$$

If A is a $n \times n$ square matrix then this has n solutions (not necessarily unique).

We need to consider how eigenvectors fit into the vector subspace picture of a matrix. Because the matrix is square, the input and output spaces are the same.

If one of the eigenvalues is zero then we have,

$$\mathbf{A}\mathbf{u}_i = \mathbf{0}. \tag{37}$$

This means that $\mathbf{u}_i$ lies in the nullspace. The rank of the matrix must therefore be n minus the number of eigenvalues equal to zero.

If an eigenvalue is non-zero then we have,

$$\mathbf{A}\mathbf{u}_i = \lambda_i\mathbf{u}_i. \tag{38}$$

This means that $\mathbf{u}_i$ must lie in the column space.

7 Singular Values

Singular values and vectors are jointly defined by the equations,

$$\mathbf{A}\mathbf{v} = \sigma\mathbf{u} \tag{39}$$

$$\mathbf{A}^\mathbf{T}\mathbf{u} = \sigma\mathbf{v}. \tag{40}$$

These work with matrices of any size.

To find the singular values and vectors, substitute each equation into the other,

$$\mathbf{A}\mathbf{A}^T\mathbf{u} = \sigma^2\mathbf{u} \tag{41}$$

$$\mathbf{A}^T\mathbf{A}\mathbf{v} = \sigma^2\mathbf{v}. \tag{42}$$

Hence, $\mathbf{u}$ and σ^2 are the eigenvectors and eigenvalues of $\mathbf{A}\mathbf{A}^T$ (so there will be m of them), and $\mathbf{v}$ and σ^2 are the eigenvectors and eigenvalues of $\mathbf{A}^T\mathbf{A}$ (n of them). Because $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ are clearly symmetric, the singular vectors are all orthogonal. Furthermore, both product matrices ony have r non-zero eigenvalues, so A has only r singular values.

If one of the singular values is zero then the corresponding vectors lie in the nullspace and left-nullspace,

$$\mathbf{A}\mathbf{v} = \mathbf{0} \tag{43}$$
$$\mathbf{A}^\mathbf{T}\mathbf{u} = \mathbf{0}. \tag{44}$$

8 Eigenvalue and Singular Value Decompositions

The n eigenvalue/eigenvector equations can be grouped into a matrix equation,

$$\mathbf{A}\underbrace{\begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix}}_{\mathbf{U}} = \underbrace{\begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix}}_{\mathbf{U}}\underbrace{\begin{bmatrix}\lambda_1 & 0 & 0\\ 0 & \lambda_2 & 0\\ 0 & 0 & \lambda_3\end{bmatrix}}_{\mathbf{\Lambda}}. \tag{45}$$

Now rearranging,

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} \tag{46}$$

In the convenient case that A is symmetric, then $\mathbf{U}$ is orthogonal, so $\mathbf{U}^{-1} = \mathbf{U}^T$.

We can do the same grouping thing with singular values. In this example, $m = 3$, $n = 4$, $r = 2$.

$$\mathbf{A}\underbrace{\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4\end{bmatrix}}_{\mathbf{V}} = \underbrace{\begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix}}_{\mathbf{U}}\underbrace{\begin{bmatrix}\sigma_1 & 0 & 0 & 0\\ 0 & \sigma_2 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}}_{\mathbf{\Sigma}} \tag{47}$$

$$\mathbf{A}^T\underbrace{\begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix}}_{\mathbf{U}} = \underbrace{\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4\end{bmatrix}}_{\mathbf{V}}\underbrace{\begin{bmatrix}\sigma_1 & 0 & 0\\ 0 & \sigma_2 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}}_{\mathbf{\Sigma}^T} \tag{48}$$

$$. \tag{49}$$

$\mathbf{U}$ and $\mathbf{V}$ are square and orthogonal (see previous section), and $\mathbf{\Sigma}$ is $m \times n$. Now rearranging,

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{50}$$

We can interpret both the eigenvalue and singular value decompositions geometrically. The input vector is first rotated in the input space, then scaled along the axes. If $\mathbf{\Sigma}$ is not square, then it dimensions are also added or removed. The resulting vector is then rotated in the output space to give the output point.