Diagonalization and subspaces

Luis Fabián Huecas López

6 de marzo de 2025

1 The process of getting the subspaces

We have a subspace U described by its cartesian equations:

$$a_{00}x_0 + a_{01}x_1 + ... + a_{0j}x_j + ... + a_{0ncol-1}x_{ncol-1} = 0 \tag{1.1}$$

$$... \tag{1.2}$$

$$a_{i0}x_0 + a_{i1}x_1 + ... + a_{ij}x_j + ... + a_{incol-1}x_{ncol-1} = 0 \tag{1.3}$$

$$... \tag{1.4}$$

$$a_{nrow-1,0}x_0 + a_{nrow-1,1}x_1 + ... + a_{nrow-1,j}x_j + ... + a_{nrow-1,ncol-1}x_{ncol-1} = 0 \tag{1.5}$$

Our goal is to get the base that generates it. The number of variables is $ncol$. This is the total size of the space. So the dimension of the subspace is $n <= ncol$.

It's good to have an idea what we have here in terms of linear spaces. Our U space is a subspace of the linear space V. Let's supposed a base of this space is $B = (e_0, e_1, ..., e_{ncol-1})$. Let's call v a vector of U, if we express it in terms of the base B we have: $v = \sum x_i e_i$. If we consider a base $B_U = (u_0, u_1, ..., u_n)$ of U we can express $v = \sum \lambda_i u_i$. Of course we can express the set $(u_i)_i$ in the base B:

$$u_i = (d_{0i}, d_{1i}, ..., d_{n_col-1,i})_B \quad i = 0, ..., n-1 \tag{1.6}$$

So we have:

$$v = \sum_i x_i e_i = \sum_i \lambda_i \sum_j d_{ji} e_j \tag{1.7}$$

Reordering it we have:

$$v = \sum_i x_i e_i = \sum_j (\sum_i \lambda_i d_{ji}) e_j \tag{1.8}$$

So we can have the so called parametric equations:

$$x_j = \sum_i d_{ji} \lambda_i, i = 0, ..., n-1 \tag{1.9}$$

The set d_{ji} gives us the base of the subspace U. But how can we get the parametric equations from our cartesian equations? Cartesian equations provides us different ways of expressing x_j in terms of the other coefficientes $(x_i)_{inoesj)}$:

$$x_j = \sum_k b_{jk} x_k) k = 0, ..., n_{col-1}, knoesj \tag{1.10}$$

As you can see the expressions are similar. If we make $x_k = \lambda_k$ we would parametrizicing the coefficients x_k but we don't know which coefficients can actually be parameters. Those are the independent coefficents, they do not depend on the others, while the ones that can be express as a linear combinations of the the independent coefficnets are the dependent coefficients. How can one know which ones are the independent ones? The answer is to transform the matrix coeffienct in a reduced matrix.

1.1 Hermite form

jfk

1.1.1 AbsArgMax

Receives a matrix and in a particular column finds the row which contains the maximum absolute value. In general not all the rows are taken into account, only from row_min.

1.1.2 row reduce rref

Perform row reduction to Reduced Row Echelon Form (RREF).

Arguments:

M(double): matrix nrow(integer): number of rows ncol(integer): number of columns ind-row(integer): indexes of the rows which are independent

Returns:

ind(integer): number of independent rows (not zeros)

We work with matrices of $ncol$ columns and $nrow$ rows:

$$\begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0j} & \cdots & a_{0ncol-1} \\ a_{10} & a_{11} & \cdots & a_{1j} & \cdots & a_{1ncol-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i0} & a_{i1} & \cdots & a_{ij} & \cdots & a_{incol-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{nrow-1,0} & a_{nrow-1,1} & \cdots & a_{nrow-1,j} & \cdots & a_{nrow-1,ncol-1} \end{pmatrix}$$

To get the RREF form we will be moving column to column and row to row from the first ones. To do this we use a while-loop and before that we initialize the variables to zero.

First of all we use the variable pivot-row to get the row where the pivot is in that particular column. From the row we are to the end of the matrix (last row).

If the pivot value is zero we just go for the next column. That means that every value in that column is zero because the pivot value is the maximum absolute value. And this row does not represent and independent equation.

If not, we interchange the rows in the matrix. The row where we are in the while-loop and pivot-row. And we take the value of the pivot and storage it in the pivot-value variable.

Now we normalize the row where we are with the pivot value. So we will have a matrix with the following form:

$$\begin{pmatrix} 1 & \frac{b_{01}}{b_{00}} & ... & \frac{b_{0j}}{b_{00}} & ... & \frac{b_{0ncol-1}}{b_{00}} \\ b_{10} & b_{11} & ... & b_{1j} & ... & b_{1ncol-1} \\ ... & ... & ... & ... & ... & ... \\ b_{i0} & b_{i1} & ... & b_{ij} & ... & b_{incol-1} \\ ... & ... & ... & ... & ... & ... \\ b_{nrow-1,0} & b_{nrow-1,1} & ... & b_{nrow-1,j} & ... & b_{nrow-1,ncol-1} \end{pmatrix}$$

Now we apply an operation to every row except the rwo where we are. And that's it, we substract from the row the row where we are in the while-loop multiplied by the element which is in the same row but always in the column where we are.

In this first case we will get the following matrix:

$$\begin{pmatrix} 1 & \frac{b_{01}}{b_{00}} & ... & \frac{b_{0j}}{b_{00}} & ... & \frac{b_{0ncol-1}}{b_{00}} \\ b_{10} - b_{10} & b_{11} - b_{10} * \frac{b_{01}}{b_{00}} & ... & b_{1j} - b_{10} * \frac{b_{0j}}{b_{00}} & ... & b_{1ncol-1} - b_{10} \\ ... & ... & ... & ... & ... & ... \\ b_{i0} - b_{i0} & b_{i1} - b_{i0} * \frac{b_{01}}{b_{00}} & ... & b_{ij} - b_{i0} * \frac{b_{0j}}{b_{00}} & ... & b_{incol-1} - b_{i0} \\ ... & ... & ... & ... & ... & ... \\ b_{nrow-1,0} - b_{nrow-1,0} & b_{nrow-1,1} - b_{nrow-1,0} * \frac{b_{01}}{b_{00}} & ... & b_{nrow-1,j} - b_{nrow-1,0} * \frac{b_{0j}}{b_{00}} & ... & b_{nrow-1,ncol-1} - b_{nro} \end{pmatrix}$$

O de otra forma:

$$\begin{pmatrix} 1 & \frac{b_{01}}{b_{00}} & ... & \frac{b_{0j}}{b_{00}} & ... & \frac{b_{0ncol-1}}{b_{00}} \\ 0 & b_{11} - b_{10} * \frac{b_{01}}{b_{00}} & ... & b_{1j} - b_{10} * \frac{b_{0j}}{b_{00}} & ... & b_{1ncol-1} - b_{10} * \frac{b_{0ncol-1}}{b_{00}} \\ 0 & ... & ... & ... & ... & ... \\ 0 & b_{i1} - b_{i0} * \frac{b_{01}}{b_{00}} & ... & b_{ij} - b_{i0} * \frac{b_{0j}}{b_{00}} & ... & b_{incol-1} - b_{i0} * \frac{b_{0ncol-1}}{b_{00}} \\ 0 & ... & ... & ... & ... & ... \\ 0 & b_{nrow-1,1} - b_{nrow-1,0} * \frac{b_{01}}{b_{00}} & ... & b_{nrow-1,j} - b_{nrow-1,0} * \frac{b_{0j}}{b_{00}} & ... & b_{nrow-1,ncol-1} - b_{nrow-1,0} * \frac{b_{0ncol-1}}{b_{00}} \end{pmatrix}$$

Now we have to move on to the next row and the next column and do everything from there.

In this function we have another important variable. ind-var[] is a vector that contains the rows that are linearly independent. The rows is considered dependent when the pivot is zero.

Until here we have the capacity to transform any matrix into the RFEF form.

1.2 Extract-Pivots

First of all we start with n-pivot = 0. Number of pivots equal to zero. And we will go column by column with a for-loop. And for each loop we initialize row=column and Pivot-found=false. That means we reserve a row variable that coincides with the column and suppose the pivot has not been found yet.

Inside the for-loop we incorporate a while-loop that operates while the pivot is not found and while the row has not reached its limit.

Inside the while-loop we initialize Is-Pivot = true. Now we check the value of the element of the matrix situated in that column and moving from row=column.

If it is zero we move to the next row not doing anything. If it is different from zero we initialize to zero a row-variable that we call k.

And here we initialize another while-loop. This operates while Is-Pivot is true and while k has not reached the limit of rows. Now we examine the elements of same column but different rows of the matrix. If it is not zero the Is-pivot turns false and always k++.

So the while-loop breaks when there is another element in the column that is not zero, that means that our element in row is not a pivot, or when the entire column has been checked.

Now we check Is-pivot, if it is, a column is incorporated in Pivots[]. n-pivot++ and Pivot-Found turns true so we move to the next column and repeat the process. If it is not a pivot then we just go for the next row inside the same column and check.

So after all we will have a vector Pivots[] where there are the indices of the columns who have a pivot.

1.3 Find-Basis

Find the basis of the subspace defined by the given Cartesian equations.

The rows are the different elements of the basis, the number of them is the dimension of the subspace, it depends on the specific M. The columns is just the size of the space, it is determined by the number of vars (num-vars).

We start with diagonal=true. for-loop from d-row -¿0 til the end, and another for-loop from d-col -¿0 til the end LESS ONE... NOT CONSIDERING THE FREE COEFFICIENTS HERE.

If d-row is not d-col (a.k. if it is not a diagonal element). If that element is not zero then diagonal = false and the structure breaks. That means the matrix is not diagonal.

Now, if it is diagonal: for-loop for rows. From 0 til the end. And we check the element of the diagonal. If it is zero, that means we have a vector of the basis that has as a independent variable that variable.

If it is not, we need to transform the matrix into the RFEF form with the former function. We extract the pivots (columns with pivots and the number of pivots). Those pivots are the dependent variables.

First we introduce the free variables in a vector called free-vars[]. And then we follow the same proccess for all the free variables.

We use a for-loop whose associated variable is k, from 0 to n-free-vars. Then we initialize a vector vect[] to zero for all the elements, from 0 to num-vars. These are gonna be the values for each element of the basis (free variables).

FREE VARIABLES = ELEMENTS OF THE BASIS = DIMENSION OF THE SUBSPACE.

Now we change the value of the vect associated with the free variable free-vars[k] = 1.

Now for the free variable we have to put the correct value for the other variables (the pivots) while the other free variables will keep the zero. We take the pivots from pivots[]. Here we associate vect[pivot] to the value in front of the free variable, that is, $-1 * M[pivot][free_var[k]]$.

Then we include the values in basis-vector[k][i]. i from 0 to num-vars. And k is the free variable.