The orthogonalization algorithm

Algebra

Consider the following linear space

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ -1 & 2 & 1 & 0 \end{pmatrix}$$

Iteration 1				
a_1	v_1	v_2	v_3	v_4
1	1	0	0	0
1	0	1	0	0
0	0	0	1	0
1	0	0	0	1
t	1	1	0	1

Transformed matrix				
	w_1	v_2	v_3	v_4
	1	-1	0	-1
	0	1	0	0
	0	0	1	0
	0	0	0	1

Iteration 2				
a_2	w_1	v_2	v_3	v_4
0	1	-1	0	-1
1	0	1	0	0
0	0	0	1	0
1	0	0	0	1
t	0	1	0	1

Transformed matrix				
	w_1	w_2	v_3	v_4
	1	-1	0	0
	0	1	0	-1
	0	0	1	0
	0	0	0	1

Iteration 3				
a_3	w_1	w_2	v_3	v_4
0	1	-1	0	0
0	0	1	0	-1
-1	0	0	1	0
1	0	0	0	1
t	0	0	-1	1

Transformed matrix				
	w_1	w_2	w_3	v_4
	1	-1	0	0
	0	1	0	-1
	0	0	-1	1
	0	0	0	1

Iteration 4				
a_4	w_1	w_2	w_3	v_4
-1	1	-1	0	0
2	0	1	0	-1
1	0	0	-1	1
0	0	0	0	1
t	-1	3	-1	-1

Transformed matrix				
	w_1	w_2	w_3	w_4
	1	-1	0	0
	1	-2	1	1
	-1	3	-2	-1
	-1	3	-1	-1

CASE OF A NON INITIAL IDENTITY MATRIX

	Iteration 1			
a_1	v_1	v_2	$\mathbf{v}_3^\star$	v_4
1	-1	-1	**0**	0
1	1	1	**0**	1
0	1	0	**0**	0
1	0	0	**1**	1
t	0	0	1	2

	Iteration 2			
a_2	$\mathbf{v}_1^\star$	v_2	w_1	v_4
0	**-1**	-1	0	0
1	**1**	1	0	1
0	**1**	0	0	0
1	**0**	0	1	-1
t	1	1	1	0

	Iteration 3			
a_3	w_2	$\mathbf{v}_2^\star$	w_1	v_4
0	-1	**0**	1	0
0	1	**0**	-1	1
-1	1	**-1**	-1	0
1	0	**0**	1	-1
t	-1	1	2	-1

	Iteration 4			
a_4	w_2	w_3	w_1	$\mathbf{v}_4^\star$
-1	-1	0	1	**0**
2	1	0	-1	**1**
1	0	-1	1	**-1**
0	0	0	1	**-1**
t	3	-1	-2	1

	Final Table			
	w_2	w_3	w_1	w_4
	-1	0	1	0
	-2	1	1	1
	3	-2	-1	-1
	3	-1	-1	-1

We get the same result as before

Iteration 1					
a_1	v_1^1	v_2^1	v_3^1	v_4^1	v_5^1
1	1	0	0	0	0
1	0	1	0	0	0
0	0	0	1	0	0
1	0	0	0	1	0
0	0	0	0	0	1
t^1	1	1	0	1	0

Iteration 2					
	w_1	v_1	v_2	v_3	v_4
	1	-1	0	-1	0
	0	1	0	0	0
	0	0	1	0	0
	0	0	0	1	0
	0	0	0	0	1

$$w_1 = v_1^1 / 1$$

$$v_1 = v_2^1 - w_1$$

$$v_2 = v_3^1$$

$$v_3 = v_4^1 - w_1$$

$$v_4 = v_5^1$$

The pivot column becomes itself (yellow) divided by its pivot (yellow)

The other columns are transformed into themselves (green) minus their scalar product (green) by the transformed pivot (blue)

A JUSTIFICATION OF THE ORTHOGONALIZATION ALGORITHM

If the orthogonalization algorithm is used with the vector a1, we have:

The scalar product of vector a1 by the transformed pivot column vector is unity.

In fact, if $t_p = a_1 \cdot V_p$ is the scalar product of a1 by the pivot vector Vp and Vp* = Vp / tp is the transformed pivot vector, we have:

$$a_1 \cdot V_p^* = a_1 \cdot \frac{V_p}{t_p} = \frac{a_1 \cdot V_p}{t_p} = \frac{t_p}{t_p} = 1,$$

that is, the scalar product of vector a1 by the transformed pivot vector V_p^* is always the unit.

A JUSTIFICATION OF THE ORTHOGONALIZATION ALGORITHM

If the orthogonalization algorithm is used with the vector a1, we have:

The scalar product of vector a1 by any of the non-pivot vectors of the transformed matrix is zero.

In fact, if $t_h = a_1 \cdot V_h$ are the scalar products of a_1 by the non-pivot vectors V_h and $V_h^* = V_h - t_h V_p^*$; $h \neq p$ are the transformed vectors of V_h, respectively, we have:

$$a_1 \cdot V_h^* = a_1 \cdot (V_h - t_h V_p^*) = a_i \cdot V_h - t_h a_1 \cdot V_p^* = t_h - t_h = 0; h \neq p,$$

that is, the scalar products of vector a1 by any of the transformed vectors of non-pivot columns, V_h^*; $h \neq p$, are always null.

The consequence is that the orthogonalization algorithm when incorporating a vector a1 = (1, 1, 0, 1) considers the subspace generated by a1 and obtains its orthogonal subspace.

$$\mathcal{L}(a_1)^{\perp} = \begin{pmatrix} -1 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}_{\rho},$$

that is, the subspace generated by the columns not used as a pivot, and its complement or complementary subspace, by the pivot ones:

$$\overline{\mathcal{L}(a_1)}^{\perp} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}_{\rho},$$

that is, the subspace generated by the columns used as a pivot, where $\mathcal{L}(a_1, a_2, \ldots, a_n)$ represents the subspace generated by the vectors a_1, $a_2, \ldots, a_n$ and the notation A_{ρ} refers to the subspace generated by the columns of the matrix A.

ANOTHER WAY OF INTERPRETING THE ALGORITHM

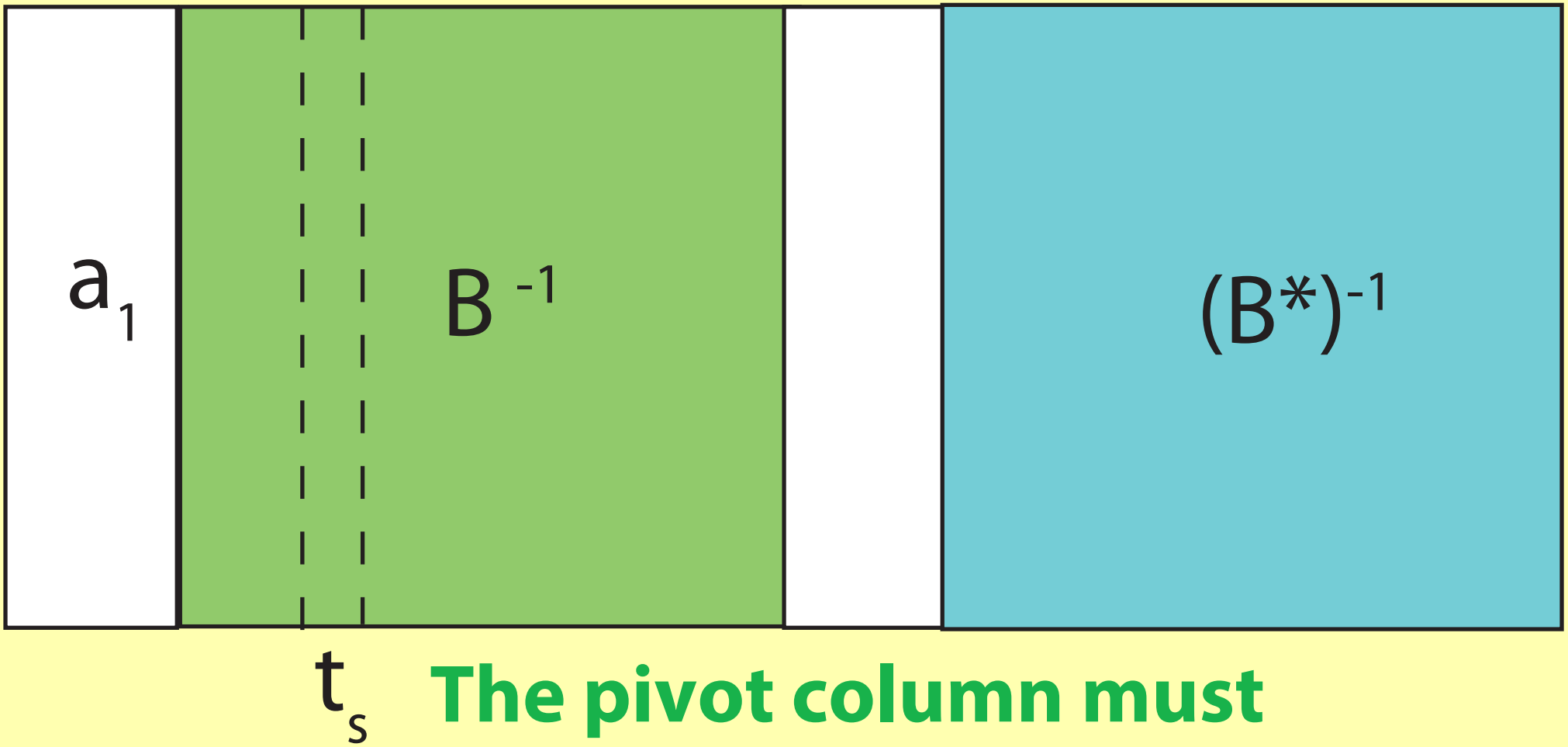
There is another way to interpret the algorithm, which is to use it to update the inverse of a matrix when changing one row.

Suppose we have a matrix A and its inverse A^{-1} and that you want to get the inverse of the matrix A^* resulting after changing in A one row a_s by another a'_s. Then, to obtain the new inverse A^{*-1} it suffices to start the algorithm with matrix A^{-1}, introduce the new row a'_s and pivot with column s. After pivoting, we get the inverse of the transformed matrix.

Update the inverse when changing a row

B is the initial matrix

B* is the initial matrix after changing row s by a_1



The pivot column must be the s column