Maths/LA/Tut6
Orthogonality

18 October 2020

CES

Last updated: 27 Oct2021
Last updated: 18 Mar 2024

Tutorial 6 Help links

Youtube link: playlist

https://www.youtube.com/playlist?list=PLki3aFwg-9exsbmLQXdb7jvhlTOtyu9f_

PDF

Q1-6: https://www.dropbox.com/s/mj0hrsw4vqqix55/Tut6_Q1_6_ces.pdf?dl=0

Q7-11: https://www.dropbox.com/s/6lpmr2z8afckir5/Tut6_Q7_11_ces.pdf?dl=0

Q1,5) Orthogonal vs Orthonormal set of vectors

Ref:

1) Read 6.3 of UCL's writeup
https://www.ucl.ac.uk/~ucahmdl/LessonPlans/Lesson10.pdf

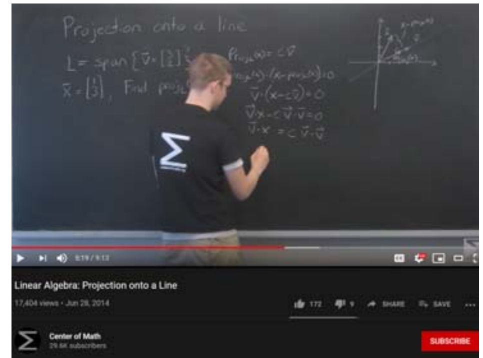
2) Youtube (Prof Dave Explains): "Orthogonality vs Orthonormality"
https://www.youtube.com/watch?v=6nqMegdbxik

3) Center of Math, "Orthonormal basis":
https://www.youtube.com/watch?v=ZJu26chXEiw

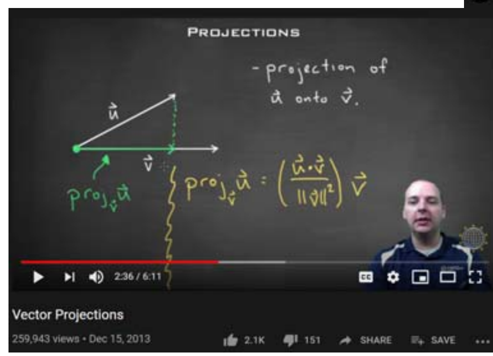
Q3,4) Examples: Videos of Projection onto a line

Ref

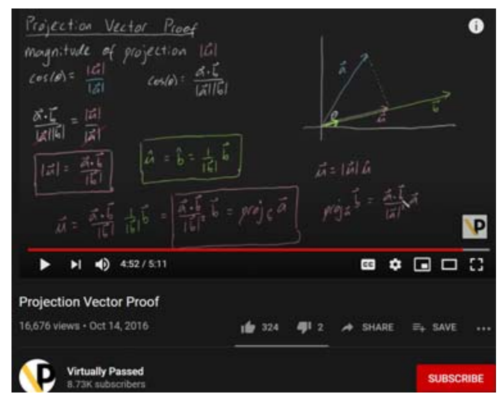
1) Projection of a vector onto a line
https://www.youtube.com/watch?v=GnvYEbaSBoY



2) Firefly (Vector Projection)
https://www.youtube.com/watch?v=fqPiDICPkj8



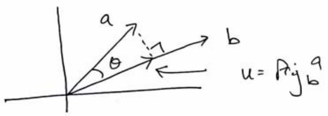
3) Virtually Passed (Projection Vector proof)
https://www.youtube.com/watch?v=aTBtgW7U-Y8



Detail Proof: Proj vector onto line

Proof : $\text{Proj}_b\, a = \dfrac{a \cdot b}{|b||b|}\, b$

Method 1 : projection vector proof by Virtually Passed - YouTube



$|u|$ = magnitude of projection a onto b.

eq (1) $\cos(\theta) = \dfrac{|u|}{|a|}$; cosine rule for ⊥ △

and by dot product

$a \cdot b = |a||b| \cos\theta$

$\Rightarrow \cos(\theta) = \dfrac{a \cdot b}{|a||b|}$. — eq (2)

∴ equating eq (1) & eq (2)

$\dfrac{|u|}{|a|} = \dfrac{a \cdot b}{|a||b|}$. ; we need $|u|$

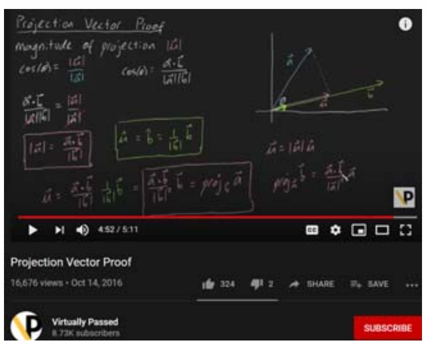
$\Rightarrow |u| = \dfrac{a \cdot b}{|b|}$.

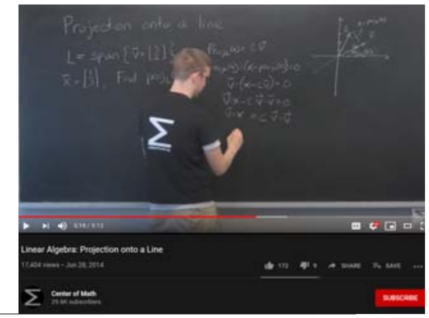
now $\text{Proj}_b\, a$ is a vector in direction of u, of b.

$\underbrace{\hat{u} = \hat{b}}_{\text{normal length}} = \dfrac{b}{|b|}$

$\therefore u = |u|\,\hat{b} = \dfrac{a \cdot b}{|b|} \cdot \dfrac{b}{|b|}$

$u = \text{Proj}_b\, a = \underbrace{\left(\dfrac{a \cdot b}{|b||b|}\right)}_{\alpha \text{ scalar}} b$ ☒



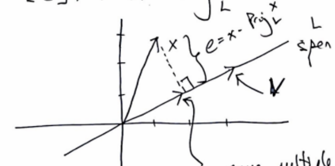


Proof : $\text{Proj}_L\, x$

Method 2: projection onto a line by "Center of Math" YouTube.

$L = \text{span}\left\{ \bar{v} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \right\}$.

$x = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$. Find $\text{Proj}_L\, x$



some multiple of v

$= \text{Proj}_L\, x = \text{Proj}_v\, x$

$= c\,v$

$e = x - \text{Proj}_L\, x$; the error.

$x = \text{Proj}_L\, x + e$

Also $e \perp v$, ∴

$\Rightarrow \underbrace{(x - \text{Proj}_L\, x)}_{e} \cdot v = 0$, remember $\text{Proj}_L\, x = c\,v$

$= v \cdot x - v \cdot (c\,v) = 0.$

$\Rightarrow v \cdot x = c(v \cdot v) = 0.$

$\Rightarrow c = \dfrac{v \cdot x}{v \cdot v} = \dfrac{v \cdot x}{|v|^2}$ ☒

$= \dfrac{v \cdot x}{\|v\|^2}$ ☒

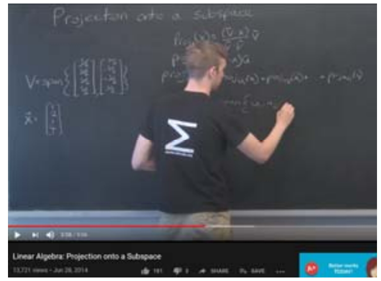
$\therefore \text{Proj}_v\, x = c\,v = \dfrac{v \cdot x}{\|v\|^2}\, v$ ☒

$\text{Proj}_v\, x = \begin{bmatrix} 27/13 \\ 18/13 \end{bmatrix}$ ☒

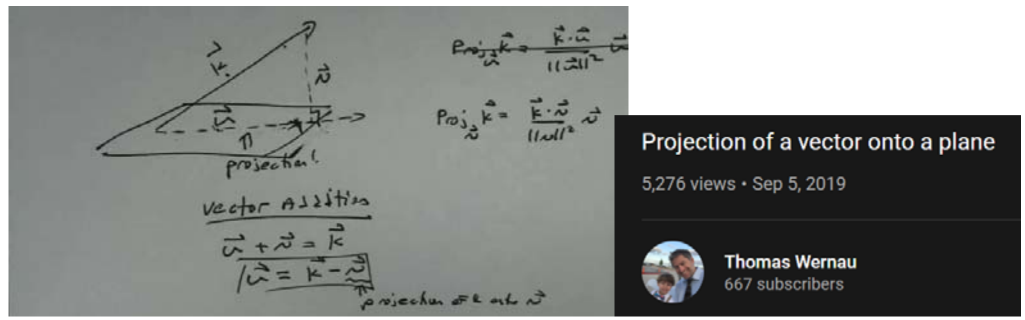
Q3,4,8) Examples: Projecting a vector onto a subspace

Ref:

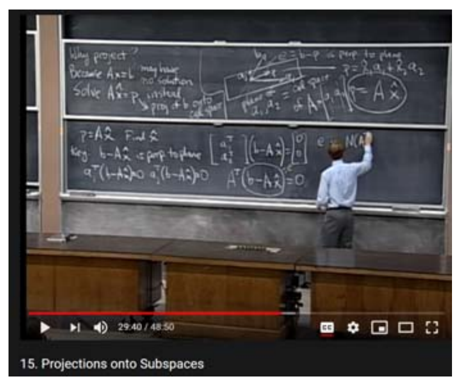
1. Center of Math: Projection onto a subspace (a direct way): https://www.youtube.com/watch?v=zZW6JV4yA54



2. Thomas Wernau: Projection of a vector onto a plane (a roundabout way – using the normal vector to the plane) : https://www.youtube.com/watch?v=qz3Q3v84k9Y



3. MIT Strang explains in L15
(This will lead to chapter 7 – least squares)
https://www.youtube.com/watch?v=Y_Ac6KiQ1t0



FAQ) Confusion with the name "Orthogonal Matrix"

a) Orthogonal matrix are square and its columns form an orthonormal set, hence its inverse is simply its transpose: https://en.wikipedia.org/wiki/Orthogonal_matrix
We like orthogonal matrix because its inverse is its transpose, and it simplify orthogonal decomposition.

b) When we have Ax = b, where A is orthogonal,

- then the number of element (column length) of b MUST be the same as column length of A
- Then A^{-1} exist and is A^T and pre-multiply Ax = b by A^{-1} to solve for x , we have:
  $A^{-1}Ax \quad = \quad A^T Ax$

=> $x = A^T b$

c) A better name for orthogonal matrix is SQUARE Orthonormal matrix
Because it is a square matrix with a set of orthonormal columns.
This is the confusion of the word orthogonal matrix:

- See: pg 5.3 and 5.6 of
- https://www.seas.ucla.edu/~vandenbe/133A/lectures/orthogonal.pdf

Q11) Help in QR –
confusion in the word orthogonal for Q

QR decomposition decomposes a rectangle (or square) matrix into two matrixes Q,R, where

- Q has orthonormal columns,

  Note: if matrix Q is square , then Q is an orthogonal matrix bcos its columns are also orthonormal.

  If matrix Q is not square, then Q'*Q = Identity Matrix

  BUT Q*Q' is not equal to Identity Matrix (its col are only orthonormal) – it is a projection matrix on colSpace(A)
- R is upper triangle

QR may be confusing because

- Size of Q matrix :
  - Depending on size of matrix A and variant in implementing QR decomposition (complete vs reduced), the matrix Q has different sizes
- Invertibility of R matrix :
  - if columns of A are linearly or not linearly independent, it will affect invertibility of R,
  - also depends on variant of implementing QR (complete vs reduced)

See code : test_QR.m

Q11)
[Q,R] = qr(A) vs [Q,R] = qr(A,0) in Matlab

https://www.mathworks.com/help/matlab/ref/qr.html

Given A = mxn matrix where m> n,
Matlab's QR factorization will decompose
- Q into full square matrix for Q of dimension (mxm), and
- R into rectangle matrix mxn, the if the so-called 'full' or 'complete' (numpy-notation) qr decomposition.

Hence, some books will immediately say that Q is an orthogonal matrix, since its columns form an orthonormal set for full decoposition.

However: Economy qr –

[Q,R] = qr(A,0) ; the 0 indicates economy selection.
The Q will be dimesion mxn, and R will be square (nxn)

My Matlab code
https://www.dropbox.com/s/1xbzb7rpjty2y4u/test_QR.m?dl=0

```
>> A = [1 2; 3 4; 5 6]

A =

     1     2
     3     4
     5     6
```

```
>> [Q,R] = qr(A)

Q =

   -0.1690    0.8971    0.4082
   -0.5071    0.2760   -0.8165
   -0.8452   -0.3450    0.4082


R =

   -5.9161   -7.4374
         0    0.8281
         0         0
```

```
>> Q'*Q

ans =

    1.0000   -0.0000   -0.0000
   -0.0000    1.0000    0.0000
   -0.0000    0.0000    1.0000

>> Q*Q'

ans =

    1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000
    0.0000    0.0000    1.0000
```

```
>> [Qecon,Recon] = qr(A,0)

Qecon =

   -0.1690    0.8971
   -0.5071    0.2760
   -0.8452   -0.3450


Recon =

   -5.9161   -7.4374
         0    0.8281
```

```
>> Qecon'*Qecon

ans =

    1.0000   -0.0000
   -0.0000    1.0000

>> Qecon*Qecon'

ans =

    0.8333    0.3333   -0.1667
    0.3333    0.3333    0.3333
   -0.1667    0.3333    0.8333
```

Q11)
[Q,R] = numpy.linalg.qr(A,'reduced' vs 'complete')

https://numpy.org/doc/stable/reference/generated/numpy.linalg.qr.html

In numpy, the choice of full vs economy uses the parameter as 'complete' vs 'reduced'

numpy.linalg.qr¶

numpy.linalg.qr(*a, mode='reduced'*)

Compute the qr factorization of a matrix.

Factor the matrix *a* as *qr*, where *q* is orthonormal and *r* is upper-triangular.

Parameters:

a : *array_like, shape (M, N)*

Matrix to be factored.

mode : *{'reduced', 'complete', 'r', 'raw'}, optional*

If K = min(M, N), then

- 'reduced' : returns q, r with dimensions (M, K), (K, N) (default)
- 'complete' : returns q, r with dimensions (M, M), (M, N)

myPythonCode =
https://www.dropbox.com/s/apmy59m8kn5hoau/test_QR_python.ipynb?dl=0

In [1]:
```
import numpy as np
A = np.array([[1,2], [3,4],[5,6]])
print(A)
```

```
[[1 2]
 [3 4]
 [5 6]]
```

In [2]:
```
[Qfull,Rfull] = np.linalg.qr(A,'complete')
print(Qfull)
print(Rfull)
```

```
[[-0.16903085  0.89708523  0.40824829]
 [-0.50709255  0.27602622 -0.81649658]
 [-0.84515425 -0.34503278  0.40824829]]
[[-5.91607978 -7.43735744]
 [ 0.          0.82807867]
 [ 0.          0.        ]]
```

In [3]:
```
[Qreduced,Rreduced] = np.linalg.qr(A,'reduced')
print(Qreduced)
print(Rreduced)
```

```
[[-0.16903085  0.89708523]
 [-0.50709255  0.27602622]
 [-0.84515425 -0.34503278]]
[[-5.91607978 -7.43735744]
 [ 0.          0.82807867]]
```

Q11) implementing your own QR decomposition (full)

If you wish to implement your own QR decomposition on A a tall and thin matrix of mxn dimension,

- note that using GS will stop after n columns.

- hence to stop GS from terminating, augment your A matrix with identity and perform GS. See discussion (right slide)

'Full' QR factorization

with $A = Q_1 R_1$ the QR factorization as above, write

$$A = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

where $[Q_1 \ Q_2]$ is orthogonal, *i.e.*, columns of $Q_2 \in \mathbf{R}^{n \times (n-r)}$ are orthonormal, orthogonal to Q_1

to find Q_2:

- find any matrix $\tilde{A}$ s.t. $[A \ \tilde{A}]$ is full rank ($e.g.$, $\tilde{A} = I$)
- apply general Gram-Schmidt to $[A \ \tilde{A}]$
- Q_1 are orthonormal vectors obtained from columns of A
- Q_2 are orthonormal vectors obtained from extra columns ($\tilde{A}$)

in pg 4-20
https://see.stanford.edu/materials/lsoeldsee263/04-qr.pdf

Some questions relating to rank and nullspace

1) Why is $A^T A$ invertible when A has full column rank?

https://www.youtube.com/watch?v=ESSMQH6Y5OA

2) Null space of AA^T is the same as $N(A)$

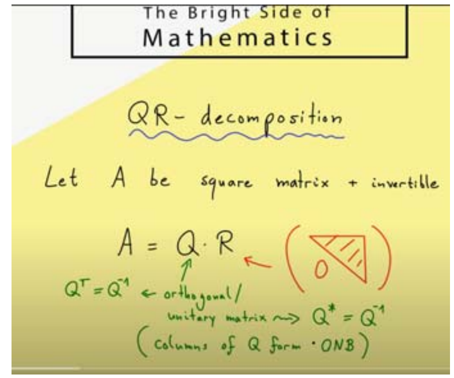
https://math.stackexchange.com/questions/66560/null-space-for-aat-is-the-same-as-null-space-for-at

3) Sum of 2 rank 1 matrix with certain properties in == rank 2

https://math.stackexchange.com/questions/2623005/sum-of-two-rank-1-matrices-with-some-property-gives-rank-2-matrix

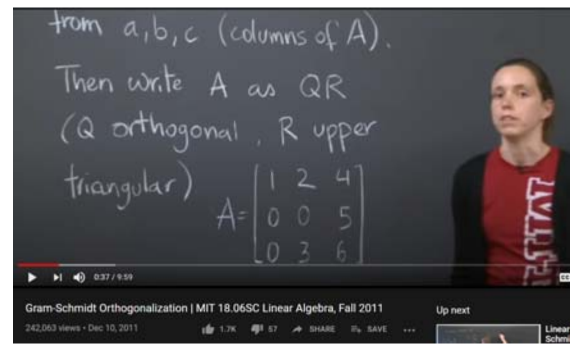
Some videos on QR

1) The bright sight of mathematics “QR on a Square matrix (step by step)”
https://www.youtube.com/watch?v=FAnNBw7d0vg



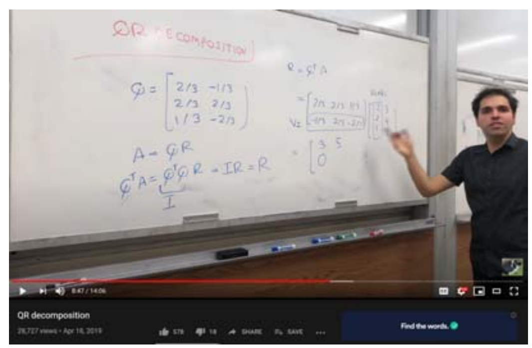
2) MIT TA doing a 3x3 matrix QR

https://www.youtube.com/watch?v=TRktLuAktBQ



3) Dr Peyam doing a 3x2 QR decomposition

https://www.youtube.com/watch?v=J41Ypt6Mftc



Q11) Orthogonal vs Unitary

In some literature, Q is sometimes called an unitary matrix instead of orthogonal matrix. This is because if A is complex, then the resultant Q will be complex. And The real analogue of a unitary matrix is an orthogonal matrix.

See:

https://www.quora.com/What-is-the-difference-between-a-unitary-and-orthogonal-matrix