Math 425 Computation Linear Algebra

HW6b

Brent A. Thorne

brentathorne@gmail.com

Basis, Othogonality, Projection, Least-squares, Factorization, and SVG.

In [1]:
```
# environment setup, try to make it clear which library I'm using for what
import numpy as np  # nice arrays and other stuff
import scipy as sci # like numpy but nicer
import sympy as sym # symbollic maths
from sympy.matrices import Matrix # pretty matrices
from sympy import Eq # pretty equations
from sympy.physics.quantum.dagger import Dagger # we'll want this later...
from math import e, pi, sqrt # Mathy math math
from mpl_toolkits.mplot3d import Axes3D # we like 3d quivers for tutorials
import matplotlib.pyplot as plt # old standby for plotting like a villian
from IPython.display import display, Math, Latex # used to display formatted re
sults in the console
sym.init_printing()  # initialize pretty printing
```

1. Find the singular values of the matrix $\begin{bmatrix} -5 & 0 \\ 0 & 0 \end{bmatrix}$.

In [2]:
```
A = Matrix([[-5,0],[0,0]])
A, A.rank(), A.T*A, (A.T*A).eigenvects()
#help(Matrix.eigenvects)
```

Out[2]:

$$\left(\begin{bmatrix} -5 & 0 \\ 0 & 0 \end{bmatrix}, 1, \begin{bmatrix} 25 & 0 \\ 0 & 0 \end{bmatrix}, \left[\left(0, 1, \left[\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right]\right), \left(25, 1, \left[\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right]\right)\right]\right)$$

In [3]:
```
print('Show semi-manual process to find SVD:')
V = Matrix([[1,0],[0,1]]) # order our eigenvects

m,n = A.shape
sigma = sym.zeros(m,n) # our matrix for sigma is the same shape as A
sigma_1 = sym.sqrt(25) # made our sigmi
sigma_2 = 0
sigma[0] = sigma_1

u1 = 1/sigma_1*A*V.col(0) # Av_k
u2 = sym.zeros(m,1) # simga_2 is 0 so just cook up a zero vector
U = Matrix([u1.T,u2.T]).T # U is our non-zero Av_k vectors

A, U*sigma*V.T  # validate our result
display(Latex(f'$A=U\\Sigma V^T= \
{sym.latex(U)}{sym.latex(sigma)}{sym.latex(V.T)}=\
{sym.latex(U*sigma*V.T)}$ * Where the non-zero $\Sigma$ are the singular value
s.'))
display(Latex(f'$A={sym.latex(A)}$, *checks out!'))  # think about this, the si
gma are our singular values
```

```
Show semi-manual process to find SVD:
```

$A = U\Sigma V^T = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -5 & 0 \\ 0 & 0 \end{bmatrix}$ * Where the non-zero Σ are the singular values.

$A = \begin{bmatrix} -5 & 0 \\ 0 & 0 \end{bmatrix}$, *checks out!

2. Suppose the factorization below is an SVD of a matrix A, with the entries in U and V rounded to two decimal places.

$$\mathbf{A} = \begin{bmatrix} -0.86 & -0.11 & -0.50 \\ 0.31 & 0.68 & -0.67 \\ 0.41 & -0.73 & -0.55 \end{bmatrix}\begin{bmatrix} 12.48 & 0 & 0 & 0 \\ 0 & 6.34 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0.66 & -0.03 & -0.35 & 0.66 \\ -0.13 & -0.90 & -0.39 & -0.13 \\ 0.65 & 0.08 & -0.16 & -0.73 \\ -0.34 & 0.42 & -0.84 & -0.08 \end{bmatrix}$$

(a) What is the rank of A?

(b) Use this decomposition of A, with no calculations, to write a basis for Col A and a basis for Nul A.

(a) The rank is 2 based on the Diagonal matrix in Σ.

(b) The basis of A is the first two columns of U, $\left\{ \begin{bmatrix} -0.86 & -0.11 \\ 0.31 & 0.68 \\ 0.41 & -0.73 \end{bmatrix} \right\}$.

The basis for the Nul A is the last two rows of the V^T, $\begin{bmatrix} 0.65 & 0.08 & -0.16 & -0.73 \\ -0.34 & 0.42 & -0.84 & -0.08 \end{bmatrix}$ or rather more clearly stated, the last two columns of V,

$$\left\{ \begin{bmatrix} 0.65 \\ 0.08 \\ -0.16 \\ -0.73 \end{bmatrix} \begin{bmatrix} -0.34 \\ 0.42 \\ -0.84 \\ -0.08 \end{bmatrix} \right\}.$$

In [4]:
```
print('Show the calculation to valid our results: (we are expecting floating po
int error)')
U = Matrix([[-0.86, -0.11, -0.50],[0.31,0.68,-0.67],[0.41,-0.73,-0.55]])
sigma = Matrix([[12.48,0,0,0],[0,6.34,0,0],[0,0,0,0]])
V = Matrix([[0.66,-0.03,-0.35,0.66],[-0.13,-0.90,-0.39,-0.13],[0.65,0.08,-0.1
6,-0.73],[-0.34,0.42,-0.84,-0.08]]).T
A = U*sigma*V.T

display(Latex(f'sympy.Matrix.rank() correctly displays the rank as {sym.latex
(A.rank())}.'))
display(Latex(f'$A^TA.eigenval()$ shows two very small eigenvalues, \
${sym.latex((A.T*A).eigenvals())}$, which provides further evidence that the ra
nk is indeed $(4-2)=2$.'))

v1 = A.col(0)
v2 = A.col(1) - v1 * A.col(1).dot(v1)/v1.dot(v1)
v1 = v1/v1.norm()
v2 = v2/v2.norm()
v1,v2

display(Latex('Show $UU^T, VV^T$: *Here we are demostrating U and V are orthono
rmal basises by rounding.'))
(U*U.T).applyfunc(lambda x: round(x,1)), (V*V.T).applyfunc(lambda x: round(x,
1))# A.col(0)/A.col(0).norm(), A.col(1)/A.col().norm()
```

```
Show the calculation to valid our results: (we are expecting floating point err
or)
```

sympy.Matrix.rank() correctly displays the rank as 2.

$A^TA.\ eigenval()$ shows two very small eigenvalues,

$\left\{ -2.13039473560594 \cdot 10^{-15} : 1,\ -2.00925079369198 \cdot 10^{-63} : 1,\ 40.3241537771784 : 1,\ 155.493056575486 : 1 \right\}$
, which provides further evidence that the rank is indeed $(4-2) = 2$.

Show UU^T, VV^T: *Here we are demostrating U and V are orthonormal basises by rounding.

Out[4]:

$$\left(\begin{bmatrix} 1.0 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}, \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix} \right)$$

3. Suppose A is square and invertible. Find the singular value decomposition of A^{-1}.

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

also recall: $A^{-1} = V\Sigma^{-1}U^T$, where $\Sigma^{-1} = diag(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \ldots, \frac{1}{\sigma_n})$

Working backwards see can see this is correct:

$A^{-1}A = (V\Sigma^{-1}U^T)(U\Sigma V^T)$

$= V\Sigma^{-1}(U^TU)\Sigma V^T)$

$= V(\Sigma^{-1}\Sigma)V^T)$

$= VV^T$

$= I$

To the Mathematician it ought to be obvious how this might be made into a proof, being a pedantic lot maybe we should just do this. Here we go...

Proof:

$A^{-1} = (U\Sigma V^T)^{-1}$

$= (V^T)^{-1}\Sigma^{-1}U^{-1}$, ($U$ and V are orthonormal, thus $(V^T)^{-1} = V$ and $U^{-1} = U^T$)

$= V\Sigma^{-1}U^T$, where $\Sigma^{-1} = diag(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \ldots, \frac{1}{\sigma_n})$.

∎

'Ah! Never to escape from Being and Number!'

-Charles Baudelaire, The Void

- see also: Moore-Penrose inverse (pseudoinverse) where, $A^\dagger = V_r\Sigma^{-1}U_r^T$
- see also Lay, Ex 7.4.7

4. Show that if A is square, then $|detA|$ is the product of the singular values of A.

Recall: $A = U\Sigma V^T$

Also recall for a square matrix:

$\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$, where $\sigma_i = \sqrt{\lambda_i}$.
These λ_i are our eigenvalues given by our characteristic polynomial of A.

We also know that U and V^{-1} are orthonormal thus are rotations that will not scale Σ. Further we can note the determinate of an orthonormal basis is 1. This property means U and V are unitary.

Thus by construction using definition of det and the fact that Σ is a diagonal matrix,

$|detA| = \prod(\sigma_1{}^2, \sigma_2{}^2, \ldots, \sigma_n{}^2)$,

$= \prod(\lambda_1, \lambda_2, \ldots, \lambda_n)$

∎

5. Find the minimal length least-squares solution of the equation $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 8 \\ 2 \end{bmatrix}.$$

In [5]:
```
display(Latex('Recall: $A^TAx=A^Tb$'))
display(Latex("Thus, $\\hat{x}=(A^TA)^{-1}A^Tb$, We've been setup! A.T*A is not invertible."))
A = Matrix([[1,1,1,1],[1,1,0,0],[0,0,1,1]]).T # Transposed to make it easiler to type
b = Matrix([1,3,8,2])
A.rank()
```

Recall: $A^TAx = A^Tb$

Thus, $\hat{x} = (A^TA)^{-1}A^Tb$, We've been setup! A.T*A is not invertible.

Out[5]: 2

In [6]:
```
print("Let's do it the long way, we've got some time to kill...")
(A.T*A).eigenvects()
```

```
Let's do it the long way, we've got some time to kill...
```

Out[6]:

$$\left[\left(0, \ 1, \ \left[\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \right] \right), \ \left(2, \ 1, \ \left[\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \right] \right), \ \left(6, \ 1, \ \left[\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \right] \right) \right]$$

In [7]:
```
s3, s2, s1 = [sym.sqrt(eig[0]) for eig in (A.T*A).eigenvects()]
v3, v2, v1 = [Matrix(eig[2][0]) for eig in (A.T*A).eigenvects()]
s1, s2, s3, v1, v2, v3 # we don't need the third one of these but we're here an
yway
```

Out[7]:

$$\left(\sqrt{6},\ \sqrt{2},\ 0,\ \begin{bmatrix}2\\1\\1\end{bmatrix},\ \begin{bmatrix}0\\-1\\1\end{bmatrix},\ \begin{bmatrix}-1\\1\\1\end{bmatrix}\right)$$

In [8]:
```
v1 = v1/v1.norm()
v2 = v2/v2.norm()

u1 = 1/s1 * A * v1
u2 = 1/s2 * A * v2

Vr = Matrix([v1.T,v2.T]).T
Ur = Matrix([u1.T,u2.T]).T
D = sym.diag(s1,s2)
Ur, D, Vr.T
```

Out[8]:

$$\left(\begin{bmatrix}\frac{1}{2} & -\frac{1}{2}\\ \frac{1}{2} & -\frac{1}{2}\\ \frac{1}{2} & \frac{1}{2}\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix},\ \begin{bmatrix}\sqrt{6} & 0\\ 0 & \sqrt{2}\end{bmatrix},\ \begin{bmatrix}\frac{\sqrt{6}}{3} & \frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{6}\\ 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{bmatrix}\right)$$

In [9]:
```
# think about Frobenius norm
f = lambda x: x**2
Af = A.applyfunc(f)
fnorm = sqrt(sum(Af))
Af, fnorm, (Ur.col(0)* sqrt(sum(Af))*Vr.col(0).T).n(1), (Vr.col(0)*1/fnorm*Ur.c
ol(0).T*b).n(2)
```

Out[9]:

$$\left(\begin{bmatrix}1 & 1 & 0\\ 1 & 1 & 0\\ 1 & 0 & 1\\ 1 & 0 & 1\end{bmatrix},\ 2.82842712474619,\ \begin{bmatrix}1.0 & 0.6 & 0.6\\ 1.0 & 0.6 & 0.6\\ 1.0 & 0.6 & 0.6\\ 1.0 & 0.6 & 0.6\end{bmatrix},\ \begin{bmatrix}2.0\\1.0\\1.0\end{bmatrix}\right)$$

In [10]:
```
Ur*D*Vr.T, A  # validate our decomposing zombie brains
```

Out[10]:

$$\left(\begin{bmatrix}1 & 1 & 0\\ 1 & 1 & 0\\ 1 & 0 & 1\\ 1 & 0 & 1\end{bmatrix},\ \begin{bmatrix}1 & 1 & 0\\ 1 & 1 & 0\\ 1 & 0 & 1\\ 1 & 0 & 1\end{bmatrix}\right)$$

In [11]:
```
# finally
x_hat_ls = Vr*D.inv()*Ur.T*b
x_hat_ls
```

Out[11]:

$$\begin{bmatrix} \frac{7}{3} \\ -\frac{1}{3} \\ \frac{8}{3} \end{bmatrix}$$

In [12]:
```
# show orthogonal projection of b_hat of b onto colA
Ur*Ur.T*b  # A*x_hat_ls
```

Out[12]:

$$\begin{bmatrix} 2 \\ 2 \\ 5 \\ 5 \end{bmatrix}$$

In [13]:
```
A*x_hat_ls # same as Ur*Ur.T*b above
```

Out[13]:

$$\begin{bmatrix} 2 \\ 2 \\ 5 \\ 5 \end{bmatrix}$$

In [14]:
```
print('Show our least squares solution: (again)')
x_hat_ls
```

```
Show our least squares solution: (again)
```

Out[14]:

$$\begin{bmatrix} \frac{7}{3} \\ -\frac{1}{3} \\ \frac{8}{3} \end{bmatrix}$$

Appendix 1. BIG PICTURE of linear algebra

4 subspaces

```
- rowspace
- nullspace
- columnspace
- leftnullspace
```

In [15]:
```
A = Matrix([[1,2,3],[4,5,6]])
rowA = A.T.col(0), A.T.col(1)
rowA # dim = r
```

Out[15]:

$$\left(\begin{bmatrix}1\\2\\3\end{bmatrix}, \begin{bmatrix}4\\5\\6\end{bmatrix}\right)$$

In [16]:
```
A.rref()# x1 - x3 = 0,  x2 + 2*x3 = 0, setting x3 == 1 yeilds, [1,-2,1]
nulA = Matrix([1,-2,1]) # n dims
A.rref(), nulA, A.nullspace(), A*nulA  # dim = n- r
```

Out[16]:

$$\left(\left(\begin{bmatrix}1 & 0 & -1\\0 & 1 & 2\end{bmatrix}, (0,\ 1)\right), \begin{bmatrix}1\\-2\\1\end{bmatrix}, \left[\begin{bmatrix}1\\-2\\1\end{bmatrix}\right], \begin{bmatrix}0\\0\end{bmatrix}\right)$$

In [17]:
```
colA = [A.col(0),A.col(1)]
A.rank(), colA, A.columnspace() # dim = r
```

Out[17]:

$$\left(2, \left[\begin{bmatrix}1\\4\end{bmatrix}, \begin{bmatrix}2\\5\end{bmatrix}\right], \left[\begin{bmatrix}1\\4\end{bmatrix}, \begin{bmatrix}2\\5\end{bmatrix}\right]\right)$$

In [18]:
```
A.T.rref() # spans R^2, thus nulAT is [0,0], or simply []
nulAT = Matrix([0,0]) # m dims
nulAT, A.T.nullspace() # dim = m-r
```

Out[18]:

$$\left(\begin{bmatrix}0\\0\end{bmatrix}, []\right)$$

Appendix 2. Practice Problems

In [19]:
```
A = Matrix([[1,2,3],[4,5,6],[7,8,9]])
In = sym.eye(3)
In[2,2]=0
In[1,1]=0

A,In,A*In
```

Out[19]:

$$\left(\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\4 & 0 & 0\\7 & 0 & 0\end{bmatrix}\right)$$

Lay Ex6.6.2

Suppose we wish to approximate the data by an equation of the form $y_1 = \beta_0 + \beta_1 x_1 + \beta_2 x^2$.

Describe the linear model that produces a "least-squares fit" of the data by the above equation.

The coordinate of data points (x_k, y_k) must statisfy the equations of the form $y_k = \beta_0 + \beta_1 x_k + \beta_2 x_k^2 + \epsilon_k$.

In [20]:
```
y = Matrix(sym.symbols('y1 y2 y_n'))
beta = Matrix(sym.symbols('beta:4'))
epsilon = Matrix(sym.symbols('epsilon1 epsilon2 epsilon_n'))
x0 = sym.ones(1,3)
x1 = sym.symbols('x1, x2, x_n')
x2 = [e**2 for e in x1]
x3 = [e**3 for e in x1]
X = Matrix([x0,x1,x2,x3]).T
y, X, beta, epsilon
display(Latex('$y=X\\beta+\\epsilon$'))  # note use of '\\beta' to escape '\' c
haractor
display(Latex("Where $y$ is 'observation vector', $X$ is the 'design matrix', \
$\\beta$ is the 'parameter vector' and $\\epsilon$ is the 'residual vector'."))
display(Latex(f'${sym.latex(y)}={sym.latex(X)}{sym.latex(beta)}+{sym.latex(epsi
lon)}$'))
```

$y = X\beta + \epsilon$

Where y is 'observation vector', X is the 'design matrix', β is the 'parameter vector' and ϵ is the 'residual vector'.

$$\begin{bmatrix}y_1\\y_2\\y_n\end{bmatrix} = \begin{bmatrix}1 & x_1 & x_1^2 & x_1^3\\1 & x_2 & x_2^2 & x_2^3\\1 & x_n & x_n^2 & x_n^3\end{bmatrix}\begin{bmatrix}\beta_0\\\beta_1\\\beta_2\\\beta_3\end{bmatrix} + \begin{bmatrix}\epsilon_1\\\epsilon_2\\\epsilon_n\end{bmatrix}$$