Solution of Gradiance Homework – 1

– By Anirban Haldar

High Dim Data Analysis

1. **Consider the diagonal matrix M =**

1	0	0
0	2	0
0	0	0

Compute its Moore-Penrose pseudoinverse, and then identify, in the list below, the true statement about the elements of the pseudoinverse.

a) **There are seven elements with value 0.**
b) **There is one element with value 0.**
c) **There is one element with value infinity.**
d) **There are seven elements with value infinity.**

=> We know, all matrices don't have invertible, like when the matrix is rectangular or singular or it has linearly dependent rows or columns. But, if a matrix is not normally invertible, than also we can calculate its inverse called pseudoinverse. The most popular pseudoinverse form is Moore-Penrose Pseudoinverse, To know this, first we learn some terminologies as

Full Row Rank : Rank of a matrix (m x n) = number of rows in that matrix.
Full Column Rank : Rank of a matrix (m x n) = number of columns in that matrix.
Rank Defect : Rank of a matrix (m x n) < number of rows and < number of columns.

The Moore-Penrose Pseudoinverse is denoted as A^+
Now, the solution is,

For Full Row Rank matrix A : $A^+ = A^T \bullet (A \bullet A^T)^{-1}$
For Full Column Rank matrix A : $A^+ = (A^T \bullet A)^{-1} \bullet A^T$
For Rank Defect Matrix A : $(U \Sigma V^T)^{-1} = (V^T)^{-1} \Sigma^{-1} U^{-1} = V \Sigma^{-1} U^T$
(Where Σ^{-1} contains the inverses of non-zero diagonal elements of Σ)

Here, our matrix A =

1	0	0
0	2	0
0	0	0

Has row rank = column rank = 2 < matrix row/column,

So, our matrix is Rank Defect matrix. So, SDV Decomposition will be,

```
m = np.matrix([[1,0,0],
               [0,2,0],
               [0,0,0]])

u, s, v = list(map(np.matrix, np.linalg.svd(m)))
s = np.diagflat(s)

# u = [[0. 1. 0.]    s = [[2. 0. 0.]    v = [[0. 1. 0.]
#      [1. 0. 0.]         [0. 1. 0.]         [1. 0. 0.]
#      [0. 0. 1.]]        [0. 0. 0.]]        [0. 0. 1.]]
```

Now to calculate Σ^{-1} (Here s^{-1}), we will invert the elements s[0, 0] = (1/2) and s[1, 1] = (1/1)
And we will calculate $V \Sigma^{-1} U^T$

```
: for i in range(min(len(s), len(s[0]))): # Inverting s
      if s[i, i]: s[i, i] = 1 / s[i, i]

  A_inv = v.dot(s).dot(u.T)
  A_inv
```

```
: matrix([[1. , 0. , 0. ],
          [0. , 0.5, 0. ],
          [0. , 0. , 0. ]])
```

We can also achieve the same result using the built-in numpy function

```
numpy.linalg.pinv()
```

```
: m = np.matrix([[1,0,0],
                 [0,2,0],
                 [0,0,0]])
  np.linalg.pinv(m)
```

```
: matrix([[1. , 0. , 0. ],
          [0. , 0.5, 0. ],
          [0. , 0. , 0. ]])
```

So, from both of the results, we can conclude that, the correct option is, **There are seven elements with value 0.**

2. **Consider the following three vectors u, v, w in a 6-dimensional space:**
   u = [1, 0.25, 0, 0, 0.5, 0]
   v = [0.75, 0, 0, 0.2, 0.4, 0]
   w = [0, 0.1, 0.75, 0, 0, 1]
   Suppose cos(x,y) denotes the similarity of vectors x and y under the cosine similarity measure. Compute all three pairwise similarities among u,v, w. Which of the following statements is true (to 4 decimal places)?

   a) cos(u,v) = 0
   b) cos(u,v) = 0.9496
   c) cos(u,w) = 0.9496
   d) cos(u,v) = 0.0174

⇨ We know, Cosine Similarity is the cosine of the angel between the vectors, which can be derived as,

$$\text{similarity(A, B)} = \frac{A \bullet B}{||A|| * ||B||}$$

So for each pair of vectors, the cosine similarity will be,

```
: u = np.array([1, 0.25, 0, 0, 0.5, 0])
  v = np.array([0.75, 0, 0, 0.2, 0.4, 0])
  w = np.array([0, 0.1, 0.75, 0, 0, 1])

  print('cos(u*v) =', sum(u*v) / (np.linalg.norm(u, 2) * np.linalg.norm(v, 2)))
  print('cos(u*w) =', sum(u*w) / (np.linalg.norm(u, 2) * np.linalg.norm(w, 2)))
  print('cos(v*w) =', sum(v*w) / (np.linalg.norm(v, 2) * np.linalg.norm(w, 2)))
```

```
cos(u*v) = 0.9496291235462798
cos(u*w) = 0.01740183416297251
cos(v*w) = 0.0
```

So, the correct answer is => cos(u,v) = 0.9496

3. **Suppose we have three points in a two dimensional space: (1,1), (2,2), and (3,4). We want to perform PCA on these points, so we construct a 2-by-2 matrix whose eigenvectors are the directions that best represent these three points. Construct this matrix and identify, in the list below, one of its elements.**

   a) 18
   b) 11
   c) 21
   d) 19

⇨ Using these points, we constructed a matrix A as A =

1	1
2	2
3	4

Now, to get the desired matrix, we have to perform $A^T \cdot A$

So, $A^T \cdot A$ =

1	2	3
1	2	4

1	1
2	2
3	4

```
A = np.matrix([[1,1],
               [2,2],
               [3,4]])
A.T.dot(A)
```

```
matrix([[14, 17],
        [17, 21]])
```

=>

14	17
17	21

So, the only number in the option, which belongs to this matrix is 21.

4. **The edit distance is the minimum number of character insertions and character deletions required to turn one string into another. Compute the edit distance between each pair of the strings he, she, his, and hers. Then, identify which of the following is a true statement about the number of pairs at a certain edit distance.**

   a) There are 2 pairs at distance 2.
   b) There are 4 pairs at distance 3.
   c) There is 1 pair at distance 5.
   d) There is 1 pair at distance 2.

=> We can calculate the common part using the recursive python implementation of Longest Common Subsequence algorithm as

```
def LCS(s1, s2, x, y):
    if not x or not y: return 0
    elif s1[x-1] == s2[y-1]: return 1 + LCS(s1, s2, x-1, y-1)
    else: return max(LCS(s1, s2, x, y-1), LCS(s1, s2, x-1, y))

strings = ['he','she','his','hers']
for i in range(len(strings)):
    for j in range(i+1, len(strings)):
        x = LCS(strings[i], strings[j], len(strings[i]), len(strings[j]))
        print(strings[i], strings[j], x)
```

```
he she 2
he his 1
he hers 2
she his 1
she hers 2
his hers 2
```

Now we can calculate the edit distance as

```
edit(s1, s2) = len(s1) + len(s2) - LCS(s1, s2) * 2

he    she => LCS : he   edit distance = 2 + 3 - 2 * 2 = 1
he    his => LCS : h    edit distance = 2 + 3 - 2 * 1 = 3
he   hers => LCS : he   edit distance = 2 + 4 - 2 * 2 = 2
she   his => LCS : h    edit distance = 3 + 3 - 2 * 1 = 4
she  hers => LCS : he   edit distance = 3 + 4 - 2 * 2 = 3
his  hers => LCS : hs   edit distance = 3 + 4 - 2 * 2 = 3
```

So, edit distance 1 -> 1 pairs
edit distance 2 -> 1 pairs
edit distance 3 -> 3 pairs
edit distance 4 -> 1 pairs

5. **Note: In this question, all columns will be written in their transposed form, as rows, to make the typography simpler. Matrix M has three rows and three columns, and the columns form an orthonormal basis. One of the columns is [2/7,3/7,6/7], and another is [6/7, 2/7, -3/7]. Let the third column be [x,y,z]. Since the length of the vector [x,y,z] must be 1, there is a constraint that x2+y2+z2 = 1. However, there are other constraints, and these other constraints can be used to deduce facts about the ratios among x, y, and z. Compute these ratios, and then identify one of them in the list below.**

   a) y = -3z
   b) z = 3y
   c) z = -3y
   d) 2z = 3x

=> We know, the properties of orthonormal vectors are,

- They are mutually perpendicular to each other i.e A • B = 0
- They have unit length i.e. || A || = || B || = 1

We have the matrix as

	C1	C2	C3
R1	$\frac{2}{7}$	$\frac{6}{7}$	x
R2	$\frac{3}{7}$	$\frac{2}{7}$	y
R3	$\frac{6}{7}$	$-\frac{3}{7}$	z

So, here, using 1st property, we can write, for column orthonormal basis,

- ✓ C1 • C3 = $\frac{2}{7}x + \frac{3}{7}y + \frac{6}{7}z = 0$ i
- ✓ C2 • C3 = $\frac{6}{7}x + \frac{2}{7}y - \frac{3}{7}z = 0$ ii

By solving (i) * 3 – (ii) we will get

⇨ $\frac{7}{7}y + \frac{21}{7}z = 0$

⇨ y + 3z = 0

⇨ y = - 3z ……. Solution 1

⇨ y = - 2x ……. Solution 2

⇨ 2x = 3z …….. Solution 3