Reconstrucción de Matrices

José Gustavo Lucas Cerón Rodríguez
Paulo César Manrique Mirón

6 de agosto de 2019

Índice

1. Introducción **1**

2. Objetivos **1**

3. Hipótesis **1**

4. Metodología **2**
4.1. Proceso de esparcificación de una matriz. 2
4.2. Recuperación de una Matriz 2

5. Resultados **4**
5.1. Esparsificación 4
5.2. Reconstrucción de Matriz 5

6. Apéndice **7**
6.1. Código 7

1. Introducción

Nos centramos en trabajar diferentes problemas relacionados con las esparsificación, compresión y reconstrucción de matrices. El trabajo consiste evaluar la metodología entregada en [1] bajo ciertas hipótesis mencionadas en él, para luego utilizar la metodología en la recuperación de matrices de bajo rango, que puedan presentar datos corruptos. Esto será aplicado en la recuperación de datos biológicos, en particular datos recopilados de un protocolo médico que estudia una enfermedad conocida como "disautonomía".

2. Objetivos

Evaluar y utilizar metodología propuesta en [1] para recuperar datos biológicos de cierta enfermedad.

3. Hipótesis

Asumimos que las hipótesis en [1] se cumplen y por lo tanto aplicamos el procedimiento mencionado en él.

4. Metodología

4.1. Proceso de esparcificación de una matriz.

Definir una matriz A de tamaño $N \times n$ que tenga rango $r << min\{N, n\}$. Asumimos que $N \geq n$ y denotamos los valores singulares diferentes de cero como $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_r$. La norma espectral de A es $||A|| = \theta_1$. Además definen los siguientes parámetros $||A||_2 = \sqrt{\sum_{i,j} a_{ij}^2}$, $||A||_1 = \sum_{i,j} |a_{ij}|$, $||A||_{max} = max_{ij}|a_{ij}|$.

Queremos contestar la siguiente pregunta: ¿en qué medida el procedimiento de dispersión modifica los parámetros espectrales claves de la matriz?.

Para eso utilizamos la esparsificacion aleatoria. Para ello obtenemos la siguiente probabilidad $p_{ij} = \frac{1}{2}(\frac{a_{i,j}^2}{||A||_2^2} + \frac{|a_{i,j}|}{||A||_1})$ con $1 \leq i \leq N, 1 \leq j \leq n$. Se crea una matriz B, tal que todas sus entradas son ceros menos la entrada i, j con probabilidad $p_{i,j}$. Repetimos este procedimiento m veces. Terminaremos con m matrices $B_1, \ldots, B_m$. Obtenemos $S(A) = \frac{1}{m}\sum_i^m B_i$.
Con el siguiente teorema respondemos a la pregunta anterior. Kundu and Drineas (ver [2] y [3])

Teorema 1 *Para cada $\epsilon > 0$, entonces existe $C > 0$ tal que si A es una matriz $N \times n$ y $m \geq Nlog(2N)$, entonces*

$$||A - S(A)|| \leq C||A||_2\sqrt{\frac{Nlog(2N)}{m}} \tag{1}$$

El nuevo resultado entregador por [1] tiene una pequeña modificación del algoritmo. Cambiamos cada entrada i, j de A, independientemente con probabilidad $\tilde{p}_{i,j} = mp_{i,j}$ y definimos $S(A)$ con entradas $\tilde{a}_{i,j} = a_{i,j}\tilde{p}_{i,j}^{-1}\chi_{i,j}$ donde $\chi_{i,j}$ son variables Bernoulli independientes con probabilidad $p_{i,j}$. La ventaje de este proceso es que se seleccionan variadas entradas al mismo tiempo, haciendo más rápido el procedimiento de esparcificación. Para garantizar que $\tilde{p}_{i,j} \leq 1$ (para tener una verdadera probabilidad) asumimos que

$$m \leq min\{\frac{||A||_2^2}{||A||_{max}^2}, \frac{||A||_1}{||A||_{max}}\} \tag{2}$$

4.2. Recuperación de una Matriz

Sea $A = \{a_{i,j}\}$ una matriz determinista de tamaño $N \times n$ de rango r, $r << min\{N, n\}$. Suponemos que se conocen un pequeño conjunto (aleatorio) de entradas de A (sub-gaussinas). Además, las entradas observadas están corrompidas por el ruido aleatorio. Por lo tanto la matriz observada $B = (b_{i,j})$ es una matriz aleatoria de tamaño $N \times n$ con entradas:

$$b_{i,j} = (a_{i,j} + z_{i,j})\chi_{i,j}\frac{1}{p} \tag{3}$$

donde $Z = (z_{i,j})$ es la matriz de ruido con entradas independientes y $\chi_{i,j}$ son variables aleatorias independientes e idénticamente distribuidas con expectativa $p \in (0, 1]$.
El objetivo es recuperar A, con la mayor precisión posible con datos parcialmente dañados.

Para esto utilizamos el siguiente teorema.

Teorema 2 ***(Descomposición de Valor Singular)*** *Cada matriz $A(N \times n)$ con rango r puede ser descompuesta como:*

$$A_{nxp} = U_{nxn} S_{nxp} V^t_{pxp} \tag{4}$$

donde U y V son columnas ortonormales, es decir $U^tU = V^tV = I_r$ y $S = diag(\theta_1^{1/2}, \ldots, \theta_r^{1/2})$, $\theta_j > 0$. Los valores $\theta_1, \ldots, \theta_r$ son valores propios diferentes de cero de las matrices AA^t y A^tA. U y V^t son correspondientemente los r vectores propios de esas matrices (ver [4],[pág. 61]).

Una vez obtenidos los vectores y valores propios de la matriz A utilizando Teorema 2, Se forma la matriz de proyección ortogonal sobre el subespacio $U = \{u_1 \ldots u_r\}$, P_u.
Hay dos maneras de encontrar la dimensión r:
La primera y más rápida es que los primeros r valores singulares sean significantemente más grande que los últimos.

$$\sum_{i=1}^{n} \theta_i^2 >> \sum_{j=n}^{r} \theta_j^2 \tag{5}$$

La segunda es iterar n veces, donde n son las columnas de nuestra matriz A. r será la iteración donde se encuentran la menor cantidad de valores nulos.

Ya con la matriz Pu y B se debe llegar a una buena aproximación de la matriz A.

$$A \approx P_u B \tag{6}$$

5. Resultados

5.1. Esparsificación

	0	1	2	3	4	5	6	7	8	9
0	42,45	45,09	42,96	41,61	44,9	40,11	33,13	28,91	27	69,86
1	41,32	43,68	41,89	44,07	45,99	45,26	47,38	46,04	46,33	63,49
2	52,9	43,17	45,23	41,41	43,6	38,01	47,94	41,06	45,81	71,33
3	36,33	33,93	37,05	36,69	40,16	34,57	38,21	35,19	37,66	58,61
4	55,52	47,71	57,93	46,08	56,73	43,82	53,64	46,91	54,2	85,35
5	45,28	39,85	48,25	41,63	42,79	35,75	47,02	39,2	51,73	77,74
6	55,1	55,16	57,48	50,57	55,51	51,88	55,66	50,21	55,9	110,36
7	64,26	59,54	66,54	58,01	64,87	58,38	66,82	57,1	59,38	113,21
8	50,73	46,02	50,3	46,88	48,64	49,02	47,34	46,39	45,35	73,48
9	57,83	61,67	63,31	59,64	60	57,36	63,22	59,54	62,62	90,17
10	42,81	37,04	39,48	33,15	43,5	34,83	49,7	39,43	50,56	74,47
11	77,05	44,82	72,52	41,75	67,2	42,26	67,2	45,12	69,61	117,39
12	51,36	50,72	52,67	48,95	51,5	48,53	52,78	46,81	54,04	78,08
13	69,81	66,54	69,83	62,06	66,94	62,03	66,96	62,39	55,11	96,25
14	59,25	57,92	60,8	54,15	61,05	54,69	57,79	54,98	57,68	88,11
15	53,63	54,18	51,7	46,31	49,12	48,73	51,42	49,65	53,31	77,9
16	77,51	63,94	73,5	64,35	75,65	45,56	67,98	53,48	60,89	114,5
17	46,88	42,96	43,31	43,55	43,14	43,64	43,09	43,9	41,63	73,59
18	31,28	31,23	27,19	28,62	25,54	26,97	28,79	24,54	29,67	47,14
19	38,85	39,23	41,25	39,05	39,39	34,32	39,39	34,32	39,39	71,39
20	36,42	32,17	37,19	36,53	35,21	31,09	35,2	35,25	35,47	57,93
21	22,06	22,03	29,32	26,21	36,29	25,2	37,81	22,67	36,15	56,43
22	46,87	49,37	41,95	46,21	38,56	47,69	38,8	38,63	45,93	68,78
23	41,58	34,98	38,88	33,45	34,38	34,59	39,44	35,12	44,94	63,44
24	62,08	55,96	63,39	53,25	64,36	50,8	57,39	53,97	61,41	97,88
25	57,64	52,42	56,25	56,28	57,31	53,12	54,67	56,46	52,96	88,8
26	69,01	66,04	65,08	54,58	64,13	58,75	63,28	59,33	61,71	103,41
27	72,9	64,49	70,47	63,73	68,7	59,81	69,29	62,98	63,92	109,97
28	68,49	57,01	63,8	56,74	61,34	58,34	54,14	46,43	41,98	100,57
29	64,92	60,87	64,3	62,04	59,94	57,33	58,16	63,74	59,08	139,15
30	80,3	79,88	80,13	77,69	78,67	73,99	83,01	78,26	78,62	115,58
31	62,01	60,16	64,99	60,6	62,65	59,02	64,62	67,74	65,55	90,43
32	59,43	51,45	54,44	45,28	54,68	45,08	55,26	51,11	56,11	91,97
33	23,53	23,09	24,66	22,36	24,09	22,34	23,93	22,81	23,5	37,6
34	82,27	74,74	73,21	68,86	70,11	69,86	70,61	82,93	71,15	124,53
35	53,83	53,88	51,44	53,72	52,88	52,94	53,65	58,47	53,64	73,15
36	78,82	71,77	80,21	70,35	77,82	75,82	73,57	73,27	75,87	112,95
37	65,39	65,98	65,14	59,17	63,2	59,05	56,63	59,32	60,52	162,46
38	38,85	37,4	37,38	34,54	34,37	34,08	35,01	34,37	33,89	66,07
39	54,36	51,69	51,47	50,23	51,52	48,4	49,93	52,33	52,97	77,25
40	52,91	57,44	53,76	60,03	54,32	59,59	61,61	64,19	62,21	130,5
41	39,84	41,3	42,04	42,17	42,65	40,9	43,36	42,72	41,31	57,33
42	49,7	40,2	46,9	34,6	43,08	38,55	47,96	38,94	43,66	71,85
43	35,71	34,08	38,65	35,59	34,4	35,02	40,54	51,91	35,64	70,53

Figura 1: Matriz A de tamaño 43×9.

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	439,44	0	0	0	0	0
1	0	505,27	0	0	0	0	0	494,11	0	0
2	464,29	0	0	0	0	0	0	0	0	0
3	0	0	539,51	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	439,26	303,37
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	474,05	0
11	0	0	0	0	412,42	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	445,56	354,5
15	0	0	0	0	0	0	0	0	0	0
16	381,67	0	0	0	0	0	0	0	0	301,13
17	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0
20	543	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0
22	490,3	0	0	0	0	0	530,03	0	0	0
23	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	421,78	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	352,87
26	0	0	0	457,53	422,55	0	0	0	0	0
27	0	421,34	402,15	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	446,88	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
31	0	436,42	0	0	0	0	0	0	0	349,07
32	0	0	0	0	457,14	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	401,72	0	0	284,84
35	0	0	0	0	0	0	0	0	0	0
36	0	0	0	402,51	0	0	0	0	0	303,82
37	0	0	0	0	425,72	0	0	0	0	236,44
38	0	0	0	0	0	0	0	0	0	0
39	0	0	470,21	0	0	0	0	0	464,01	0
40	0	0	0	0	0	0	431,25	0	0	0
41	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	397,96
43	0	556,4	0	0	0	0	0	0	0	0

Figura 2: Esparsificación de la matriz A. Contiene 37 valores no cero. Su $|\frac{||S\tilde{(}A)||}{||A||} - 1| = 0{,}0199$.

Se ha visto que cuando una matriz es escasa, los cálculos se pueden hacer mucho mas rápido, pudiendo almacenar mayor cantidad de datos y a la vez transportar de forma más eficiente. Por supuesto uno necesita crear una versión dispersa de la matriz donde sus parámetros esenciales no se desvíen demasiado de los del original como en el ejemplo de la figura 2.

5.2. Reconstrucción de Matriz

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	102	0	0	0	0	28	583	56,59	82,77	35,8	0,82	0,56	2,31
1	102	0	0	0	0	0	582	56,21	83,16	35,42	0,83	0,57	2,33
2	103	0	0	0	0	0	574	55,82	83,54	34,65	0,86	0,58	2,4
3	104	0	0	0	0	0	576	55,82	83,93	34,65	0,86	0,58	2,4
4	104	0	0	0	0	0	583	56,59	85,47	34,65	0,88	0,59	2,46
5	102	0	0	0	0	0	587	56,98	85,47	35,03	0,87	0,58	2,42
6	102	103	62	75	0	28	587	56,59	84,31	35,42	0,85	0,57	2,36
7	102	99	61	73	0	28	597	56,98	85,85	36,57	0,85	0,57	2,34
8	100	98	61	73	0	0	593	57,36	86,62	35,8	0,87	0,58	2,41
9	101	97	61	73	0	0	593	57,75	85,85	35,42	0,86	0,58	2,4
10	101	99	61	73	0	0	595	58,9	87,39	37,34	0,84	0,56	2,33
11	100	102	61	74	0	28	591	59,67	87,39	38,5	0,8	0,55	2,27
12	101	106	61	76	0	29	589	60,44	87,78	41,19	0,76	0,53	2,13
13	101	111	61	77	0	0	587	61,21	88,55	40,81	0,77	0,53	2,15
14	102	114	61	78	0	0	589	61,6	88,55	41,19	0,76	0,53	2,14
15	101	114	61	78	0	0	593	60,83	88,16	38,11	0,8	0,56	2,31
16	101	113	62	79	0	0	595	60,83	88,16	38,11	0,81	0,56	2,31
17	100	113	62	79	0	0	597	61,21	88,16	39,27	0,78	0,54	2,24
18	100	112	62	78	0	30	601	62,37	88,16	38,88	0,77	0,55	2,25
19	99	112	62	78	0	29	599	62,75	88,55	41,19	0,75	0,53	2,14
20	100	113	62	79	0	0	599	61,98	87,01	40,04	0,75	0,53	2,17
21	100	113	62	79	0	0	607	60,44	85,47	38,5	0,77	0,54	2,21
22	98	113	63	79	0	0	626	59,67	84,7	36,96	0,79	0,56	2,28
23	95	114	63	80	0	0	644	60,06	84,31	36,96	0,77	0,55	2,26
24	93	113	63	79	0	0	638	60,83	85,47	38,5	0,76	0,54	2,21

Figura 3: Tenemos una matriz A de tamaño 2069×13, rango $r = 13$ y con 4807 datos corruptos.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	102	75,6822	45,2457	55,8828	7,3169	28	583	56,59	82,77	35,8	0,82	0,56	2,31
1	102	231,022	138,114	170,584	22,335	18,9869	582	56,21	83,16	35,42	0,83	0,57	2,33
2	103	0	0	0	0	0	574	55,82	83,54	34,65	0,86	0,58	2,4
3	104	0	0	0	0	0	576	55,82	83,93	34,65	0,86	0,58	2,4
4	104	137,799	82,3817	101,749	13,3223	11,3253	583	56,59	85,47	34,65	0,88	0,59	2,46
5	102	63,0464	37,6915	46,5526	6,09528	5,18158	587	56,98	85,47	35,03	0,87	0,58	2,42
6	102	103	62	75	8,40873	28	587	56,59	84,31	35,42	0,85	0,57	2,36
7	102	99	61	73	11,3955	28	597	56,98	85,85	36,57	0,85	0,57	2,34
8	100	98	61	73	3,23428	2,74946	593	57,36	86,62	35,8	0,87	0,58	2,41
9	101	97	61	73	0	0	593	57,75	85,85	35,42	0,86	0,58	2,4
10	101	99	61	73	0	0	595	58,9	87,39	37,34	0,84	0,56	2,33
11	100	102	61	74	0	28	591	59,67	87,39	38,5	0,8	0,55	2,27
12	101	106	61	76	7,75481	29	589	60,44	87,78	41,19	0,76	0,53	2,13
13	101	111	61	77	9,29157	7,89873	587	61,21	88,55	40,81	0,77	0,53	2,15
14	102	114	61	78	9,37005	7,96545	589	61,6	88,55	41,19	0,76	0,53	2,14
15	101	114	61	78	9,13949	7,76946	593	60,83	88,16	38,11	0,8	0,56	2,31
16	101	113	62	79	9,09134	7,72853	595	60,83	88,16	38,11	0,81	0,56	2,31
17	100	113	62	79	9,07352	7,71337	597	61,21	88,16	39,27	0,78	0,54	2,24
18	100	112	62	78	9,96725	30	601	62,37	88,16	38,88	0,77	0,55	2,25
19	99	112	62	78	9,95397	29	599	62,75	88,55	41,19	0,75	0,53	2,14
20	100	113	62	79	9,10634	7,74127	599	61,98	87,01	40,04	0,75	0,53	2,17
21	100	113	62	79	9,03841	7,68353	607	60,44	85,47	38,5	0,77	0,54	2,21
22	98	113	63	79	8,81261	7,49157	626	59,67	84,7	36,96	0,79	0,56	2,28
23	95	114	63	80	8,54208	7,2616	644	60,06	84,31	36,96	0,77	0,55	2,26
24	93	113	63	79	8,48492	7,213	638	60,83	85,47	38,5	0,76	0,54	2,21

Figura 4: Primera iteración con el modelo propuesto en [3][4][5][6], nos entrega 65 valores sin poder recuperar.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	102	75,6822	45,2457	55,8828	7,3169	28	583	56,59	82,77	35,8	0,82	0,56	2,31
1	102	231,022	138,114	170,584	22,335	18,9869	582	56,21	83,16	35,42	0,83	0,57	2,33
2	103	0	0	0	0	0	574	55,82	83,54	34,65	0,86	0,58	2,4
3	104	45,9552	26,3908	33,5574	7,40432	6,00667	576	55,82	83,93	34,65	0,86	0,58	2,4
4	104	137,799	82,3817	101,749	13,3223	11,3253	583	56,59	85,47	34,65	0,88	0,59	2,46
5	102	63,0464	37,6915	46,5526	6,09528	5,18158	587	56,98	85,47	35,03	0,87	0,58	2,42
6	102	103	62	75	8,40873	28	587	56,59	84,31	35,42	0,85	0,57	2,36
7	102	99	61	73	11,3955	28	597	56,98	85,85	36,57	0,85	0,57	2,34
8	100	98	61	73	3,23428	2,74946	593	57,36	86,62	35,8	0,87	0,58	2,41
9	101	97	61	73	14,6712	11,9019	593	57,75	85,85	35,42	0,86	0,58	2,4
10	101	99	61	73	0,57567	0,46701	595	58,9	87,39	37,34	0,84	0,56	2,33
11	100	102	61	74	5,6701	28	591	59,67	87,39	38,5	0,8	0,55	2,27
12	101	106	61	76	7,75481	29	589	60,44	87,78	41,19	0,76	0,53	2,13
13	101	111	61	77	9,29157	7,89873	587	61,21	88,55	40,81	0,77	0,53	2,15
14	102	114	61	78	9,37005	7,96545	589	61,6	88,55	41,19	0,76	0,53	2,14
15	101	114	61	78	9,13949	7,76946	593	60,83	88,16	38,11	0,8	0,56	2,31
16	101	113	62	79	9,09134	7,72853	595	60,83	88,16	38,11	0,81	0,56	2,31
17	100	113	62	79	9,07352	7,71337	597	61,21	88,16	39,27	0,78	0,54	2,24
18	100	112	62	78	9,96725	30	601	62,37	88,16	38,88	0,77	0,55	2,25
19	99	112	62	78	9,95397	29	599	62,75	88,55	41,19	0,75	0,53	2,14
20	100	113	62	79	9,10634	7,74127	599	61,98	87,01	40,04	0,75	0,53	2,17
21	100	113	62	79	9,03841	7,68353	607	60,44	85,47	38,5	0,77	0,54	2,21
22	98	113	63	79	8,81261	7,49157	626	59,67	84,7	36,96	0,79	0,56	2,28
23	95	114	63	80	8,54208	7,2616	644	60,06	84,31	36,96	0,77	0,55	2,26
24	93	113	63	79	8,48492	7,213	638	60,83	85,47	38,5	0,76	0,54	2,21

Figura 5: Cuarta iteración, 5 valores sin recuperar.

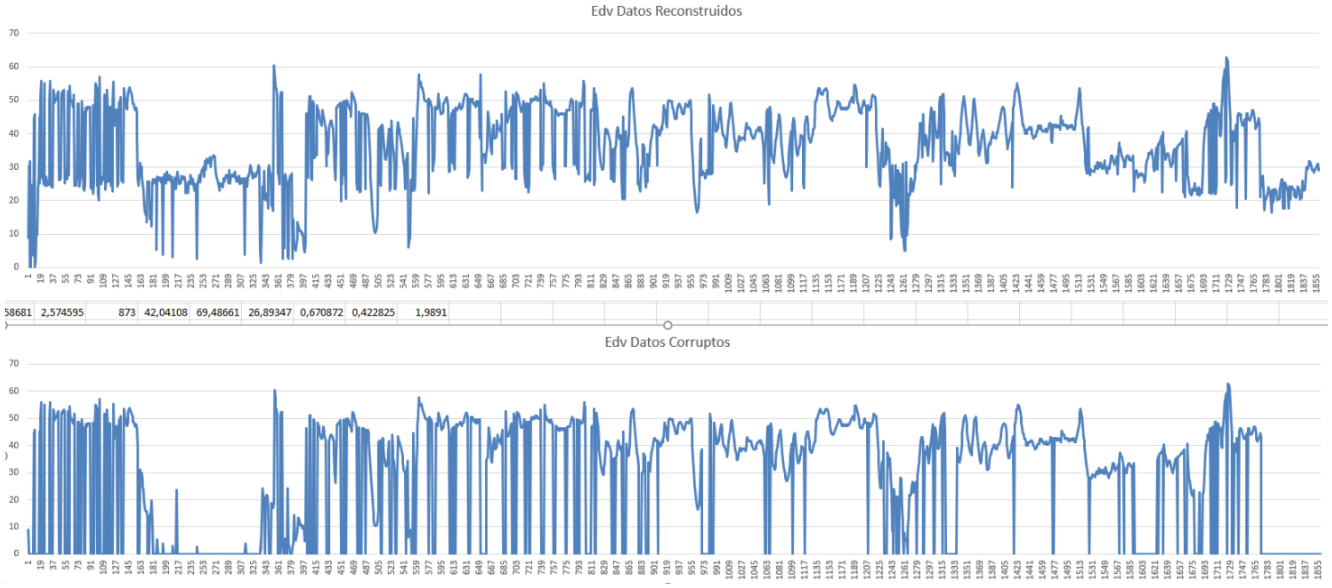

Figura 6: Graficas EDV.

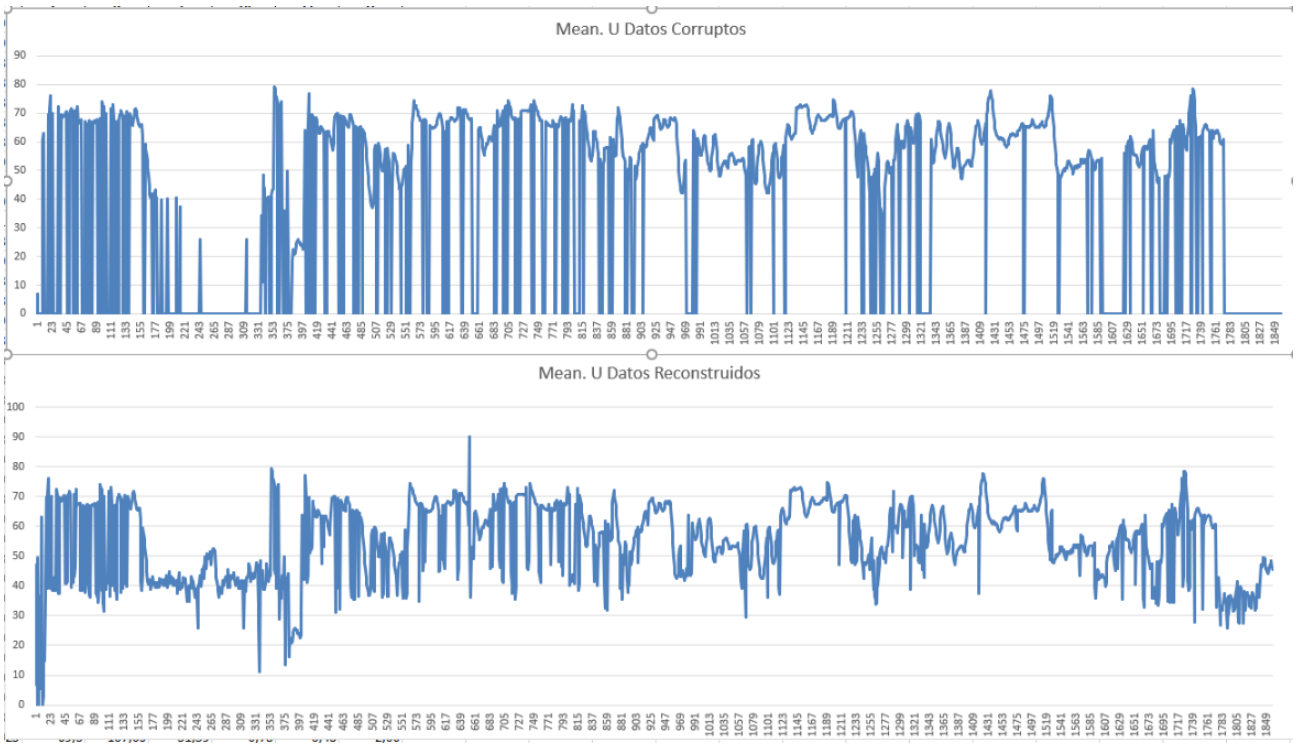

Figura 7: Graficas MeanU.

Dado que los datos utilizados son valores que representan diferentes actividades de una persona en particular, en un protocolo medico, lo natural es que haya cierta tendencia en las graficas reales. En las reconstrucciones planteadas, se muestra que si tienen cierta coherencias con lo que está pasando en la persona.

6. Apéndice

6.1. Código

```
def norma_dos(x):
    sumatoria = 0
    for i in range(len(x)):
        for j in range(len(x[0])):
            sumatoria = sumatoria + (x[i][j]**2)
    raiz = sumatoria ** 0.5
    return raiz


def norma_uno(x):
    sum_abs = 0
    for i in range(len(x)):
        for j in range(len(x[0])):
            sum_abs = sum_abs + abs(x[i][j])
    return sum_abs

def suma_2(x):
    suma= 0
    for i in range(len(x)):
```

```
        for j in range(len(x[0])):
            suma += x[i][j]**2
    return suma

def suma_abs(x):
    suma= 0
    for i in range(len(x)):
        for j in range(len(x[0])):
            suma += x[i][j]
    return suma


#Encontrar el maximo valor de una matriz
#np.amax()

#Encontrar constante Cauchy-Schwartz
def cs(x):
    cs = norma_uno(x)/ ((norma_dos(x)) * (len(x)* len(x[0]))**0.5)
    return cs

#Encontrar el valor de M para las iteraciones.
def valor_M(x):
    valor = (norma_dos(x)**2 / np.amax(x)**2) , (norma_uno(x)/ np.amax(x))
    min = np.amin(valor)
    return min

#Encontrar la matriz de distribucion Pij
def distribucion(x):
    P = np.zeros((len(x),len(x[0])))
    a = norma_dos(x)**2
    b= norma_uno(x)
    for i in range (len(x)):
        for j in range(len(x[0])):
            P[i][j] = 0.5 * (((x[i][j]**2)/ (a)) + (abs(x[i][j])/b))
    return P

#Convierte la matriz en una lista
#.flatten()

def m_base(x):
    m = len(x)* np.log(2*len(x))
    return m

def posicion(x):
    Posicion= np.zeros(len(x.flatten()))
    for i in range(len(x.flatten())):
        Posicion[i] = i
    return Posicion

def Matriz_SA(x, m= None):
```

```
    if m == None:
        m = m_base(x)
    #Crear una lista que contiene varias listas.
    obj = [[] for i in range(int(m))]
    B = np.zeros(len(x.flatten()))
    S_A = np.array(np.zeros((len(x), len(x[0]))))
    distri = distribucion(x).flatten()
    for i in range(int(m)):
        Elemento = np.random.choice(posicion(x), 1 , p = distri)
        obj[i] = np.zeros(len(x.flatten()))
        obj[i][int(Elemento)] = (x.flatten()[int(Elemento)])
        B = B + obj[i]
    S_A = B /valor_M(x)
    S_A = S_A.reshape(len(x), len(x[0]))
    return S_A


def Nueva_P(x):
    N_P = np.zeros((len(x),len(x[0])))
    m = valor_M(x)
    a = norma_dos(x)**2
    b= norma_uno(x)
    for i in range (len(x)):
        for j in range(len(x[0])):
            #Definir la nueva matriz de probabilidades
            N_P[i][j] = m *(0.5 * (((x[i][j]**2)/ (a)) + (abs(x[i][j])/b)))
    return N_P

def P_funciona(x):
    y = Nueva_P(x)
    P = np.zeros((len(x), len(x[0])))
    for i in range(len(y)):
        for j in range(len(y[0])):
            if y[i][j] == 0:
                P[i][j] = 0
            else:
                P[i][j] = y[i][j] **-1
    return P

def Nueva_SA(x):
    X = np.random.binomial(1, p = Nueva_P(x))
    Nueva_SA = np.zeros((len(x),len(x[0])))
    P = P_funciona(x)
    for i in range(len(x)):
        for j in range(len(x[0])):
            Nueva_SA[i][j] =  x[i][j] * P[i][j] * X[i][j]
    return Nueva_SA

def valor_minimo(x,y):
```

```
    minimo = 1
    a = np.linalg.norm(x,2)
    SA_uso = np.zeros((len(x), len(x[0])))
    for i in range(y):
        b = Nueva_SA(x)
        espec = np.linalg.norm(b,2)
        c = espec/a - 1
        if c < minimo:
            SA_uso = b
            minimo = c
    return SA_uso, abs(minimo)

def Matriz_B(x, p = None):
    if p == None:
        p = 0.5
    B= np.zeros((len(x), len(x[0])))
    for i in range(len(x)):
        for j in range(len(x[0])):
            B[i][j] = (x[i][j] + np.random.normal(0,1)) * np.random.binomial(1, p ) * (1
    return B


def funcion_Pu(x, y= None):
    if y == None:
        y = 5
    u , s, vt = np.linalg.svd(x ,full_matrices=True)
    X= np.zeros((len(x),y))
    for i in range(y):
        X[:,i]= u[i]

    Xt = np.zeros((len(X[0]),len(X)))
    for i in range(len(X[0])):
        for j in range(len(X)):
            Xt[i][j] = X[j][i]

    Pu = np.dot(X, Xt)
    PuB = np.dot(Pu,Matriz_B(x))


    return PuB

def negativos(x):
    b=0
    for i in range(len(x)):
        for j in range(len(x[0])):
            if x[i][j] < 0:
                b += 1
    return b

def conversion(x):
```

```
    x = x.fillna(0)
    x = np.array(x)
    x = x[:,1:15]
    x = np.array(x, dtype = 'float')
    return x

def funcion_Pu_ideal(x):
    y = np.linalg.matrix_rank(x)
    a = len(x) * len(x[0])
    i=1
    while i <= y:
        w = funcion_Pu(x, i)
        b = negativos(w)
        if b < a:
            f = i
            a = b
        i +=1
    z = funcion_Pu(x,f)
    return z

def no_negativos(x):
    for i in range(len(x)):
        for j in range(len(x[0])):
            if x[i][j] <0:
                x[i][j] = 0
    return x

def completar(x , y):
    for i in range(len(x)):
        for j in range(len(x[0])):
            if x[i][j] != 0:
                y[i][j] = x[i][j]
    return y

def cant_ceros(x):
    b=0
    for i in range(len(x)):
        for j in range(len(x[0])):
            if x[i][j] == 0:
                b += 1
    return b
```

Referencias

[1] O'Rourke, Sean and Vu, Van and Wang, Ke. Random perturbation and matrix sparsification and completion, 2018

[2] A. Kundu and P.Drineas. A note on randomized element-wise matrix sparsification. arXiv:1404.0320, 2014.

[3] J.A. Tropp. An introduction to matrix concentration inequalities. Found. Trends Mach. Learn., 8(1-2):1-230, May 2015.

[4] © Springer-Verlag Berlin Heidelberg 2015 W.K. Härdle, L. Simar, Applied Multivariate Statistical Analysis, DOI 10.1007/978-3-662-45171-7

Recuperación de Matrices

José Cerón Rodríguez.
Paulo Manrique Mirón.

IV **Verano de la Investigación en Matemáticas 2019**

Cuernavaca, Agosto 2019

Introducción

Nos centramos en trabajar diferentes problemas relacionados con las esparsificación, compresión y reconstrucción de matrices. El trabajo consiste evaluar la metodología entregada en [1] bajo ciertas hipótesis mencionadas en él, para luego utilizar la metodología en la recuperación de matrices de bajo rango, que puedan presentar datos corruptos. Esto será aplicado en la recuperación de datos biológicos, en particular datos recopilados de un protocolo médico que estudia una enfermedad conocida como "disautonomía".

Objetivos

- General: Evaluar y utilizar metodología propuesta en [1] para recuperar datos biológicos de cierta enfermedad.
- Específicos:
  - Analizar metodología propuesta en [1].
  - Programar código en lenguaje Python.
  - Reconstruir matriz con valores de un protocolo médico.

Hipótesis

Asumimos que las hipótesis en [1] se cumplen y por lo tanto aplicamos el procedimiento mencionado en él.

Definir una matriz A de tamaño $N \times n$ que tenga rango $r << min\{N, n\}$. Asumimos que $N \geq n$ y denotamos los valores singulares diferentes de cero como $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_r$. La norma espectral de A es $||A|| = \theta_1$. Además definen los siguientes parámetros $||A||_2 = \sqrt{\sum_{i,j} a_{ij}^2}$, $||A||_1 = \sum_{i,j} |a_{ij}|$, $||A||_{max} = max_{ij}|a_{ij}|$.

¿En qué medida el procedimiento de dispersión modifica los parámetros espectrales claves de la matriz?

Para ello obtenemos la siguiente probabilidad $p_{ij} = \frac{1}{2}(\frac{a_{i,j}^2}{||A||_2^2} + \frac{|a_{i,j}|}{||A||_1})$ con $1 \leq i \leq N$, $1 \leq j \leq n$.

Se crea una matriz B, tal que todas sus entradas son ceros menos la entrada i, j con probabilidad $p_{i,j}$.

Repetimos este procedimiento m veces. Terminaremos con m matrices $B_1, \ldots, B_m$. Obtenemos $S(A) = \frac{1}{m}\sum_i^m B_i$.

Con el siguiente teorema respondemos a la pregunta anterior. Kundu and Drineas (ver [2] y [3])

Teorema

Para cada $\epsilon > 0$, entonces existe $C > 0$ tal que si A es una matriz $N \times n$ y $m \geq Nlog(2N)$, entonces

$$||A - S(A)|| \leq C||A||_2\sqrt{\frac{Nlog(2N)}{m}} \quad (1)$$

El nuevo resultado entregado por [1] tiene una pequeña modificación del algoritmo. Cambiamos cada entrada i, j de A, independientemente con probabilidad $\tilde{p}_{i,j} = mp_{i,j}$ y definimos $S(A)$ con entradas $\tilde{a}_{i,j} = a_{i,j}\tilde{p}_{i,j}^{-1}\chi_{i,j}$ donde $\chi_{i,j}$ son variables Bernoulli independientes con probabilidad $p_{i,j}$. La ventaje de este proceso es que se seleccionan variadas entradas al mismo tiempo, haciendo más rápido el procedimiento de esparcificación. Para garantizar que $\tilde{p}_{i,j} \leq 1$ (para tener una verdadera probabilidad) asumimos que

$$m \leq min\{\frac{||A||_2^2}{||A||_{max}^2}, \frac{||A||_1}{||A||_{max}}\} \tag{2}$$

Sea $A = \{a_{i,j}\}$ una matriz determinista de tamaño $N \times n$ de rango r, $r << min\{N, n\}$.
Suponemos que se conocen un pequeño conjunto (aleatorio) de entradas de A (sub-gaussina). Además, las entradas observadas están corrompidas por el ruido aleatorio. Por lo tanto la matriz observada $B = (b_{i,j})$ es una matriz aleatoria de tamaño $N \times n$ con entradas:

$$b_{i,j} = (a_{i,j} + z_{i,j})\chi_{i,j} \tag{3}$$

donde $Z = (z_{i,j})$ es la matriz de ruido con entradas independientes y $\chi_{i,j}$ son variables aleatorias independientes e idénticamente distribuidas con expectativa $p \in (0, 1]$.

El objetivo es recuperar A, con la mayor precisión posible con datos parcialmente dañados.
Para esto utilizamos el siguiente teorema.

Teorema

*(**Descomposición de Valor Singular**) Cada matriz $A(N \times n)$ con rango r puede ser descompuesta como:*

$$A_{nxp} = U_{nxn} S_{nxp} V^t_{pxp} \tag{4}$$

donde U y V son columnas ortonormales, es decir $U^tU = V^tV = I_r$ y $S = diag(\theta_1^{1/2}, \ldots, \theta_r^{1/2})$, $\theta_j > 0$. Los valores $\theta_1, \ldots, \theta_r$ son valores propios diferentes de cero de las matrices AA^t y A^tA. U y V^t son correspondientemente los r vectores propios de esas matrices (ver [4],[pág. 61]).

Una vez obtenidos los vectores y valores propios de la matriz A utilizando Teorema 2, Se forma la matriz de proyección ortogonal sobre el subespacio $U = \{u_1 \dots u_r\}$, $Pu_{i,j}$.
Hay dos maneras de encontrar la dimensión r: La primera y más rápida es que los primeros r valores singulares sean significantemente más grande que los últimos.

$$\sum_{i=1}^{n} \theta_i^2 >> \sum_{j=n}^{r} \theta_j^2 \tag{5}$$

La segunda es iterar n veces, donde n son las columnas de nuestra matriz A. r será la iteración donde se encuentran la menor cantidad de valores nulos.

Ya con la matriz P_u y B se debe llegar a una buena aproximación de la matriz A.

$$A \approx P_u B \tag{6}$$

Cuadro: A

42,45	45,09	42,96	41,61	44,9	40,11	33,13	28,91	27	69,86
41,32	43,68	41,89	44,07	45,99	45,26	47,38	46,04	46,33	63,49
52,9	43,17	45,23	41,41	43,6	38,01	47,94	41,06	45,81	71,33
36,33	33,93	37,05	36,69	40,16	34,57	38,21	35,19	37,66	58,61
55,52	47,71	57,93	46,08	56,73	43,82	53,64	46,91	54,2	85,35
45,28	39,85	48,25	41,63	42,79	35,75	47,02	39,2	51,73	77,74
55,1	55,16	57,48	50,57	55,51	51,88	55,66	50,21	55,9	110,36
64,26	59,54	66,54	58,01	64,87	58,38	66,82	57,1	59,38	113,21
50,73	46,02	50,3	46,88	48,64	49,02	47,34	46,39	45,35	73,48
57,83	61,67	63,31	59,64	60	57,36	63,22	59,54	62,62	90,17
42,81	37,04	39,48	33,15	43,5	34,83	49,7	39,43	50,56	74,47
77,05	44,82	72,52	41,75	67,2	42,26	67,2	45,12	69,61	117,39
51,36	50,72	52,67	48,95	51,5	48,53	52,78	46,81	54,04	78,08
69,81	66,54	69,83	62,06	66,94	62,03	66,96	62,39	55,11	96,25
59,25	57,92	60,8	54,15	61,05	54,69	57,79	54,98	57,68	88,11
53,63	54,18	51,7	46,31	49,12	48,73	51,42	49,65	53,31	77,9
77,51	63,94	73,5	64,35	75,65	45,56	67,98	53,48	60,89	114,5
46,88	42,96	43,31	43,55	43,14	43,64	43,09	43,9	41,63	73,59
31,28	31,23	27,19	28,62	25,54	26,97	28,79	24,54	29,67	47,14
38,85	39,23	41,25	39,05	39,39	34,32	39,39	34,32	39,39	71,39
36,42	32,17	37,19	36,53	35,21	31,09	35,2	35,25	35,47	57,93
22,06	22,03	29,32	26,21	36,29	25,2	37,81	22,67	36,15	56,43

Cuadro: $\tilde{S}(A)$. Su $|\frac{||S\tilde{(A)}||}{||A||} - 1| = 0{,}0199$.

0	0	0	0	499,4355	0	0	0	0	0
0	505,2678	0	0	0	0	0	494,1061	0	0
464,2926	0	0	0	0	0	0	0	0	0
0	0	539,5058	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	439,2567	303,3666
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	474,0494	0
0	0	0	0	412,4193	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	445,5598	354,5042
0	0	0	0	0	0	0	0	0	0
381,6747	0	0	0	0	0	0	0	0	301,1341
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
543,0022	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Se ha visto que cuando una matriz es escasa, los cálculos se pueden hacer mucho mas rápido, pudiendo almacenar mayor cantidad de datos y a la vez transportar de forma más eficiente la información. Por supuesto uno necesita crear una versión dispersa de la matriz donde sus parámetros esenciales no se desvíen demasiado de los del original como en el ejemplo de la figura 2.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	102	0	0	0	0	28	583	56,59	82,77	35,8	0,82	0,56	2,31
1	102	0	0	0	0	0	582	56,21	83,16	35,42	0,83	0,57	2,33
2	103	0	0	0	0	0	574	55,82	83,54	34,65	0,86	0,58	2,4
3	104	0	0	0	0	0	576	55,82	83,93	34,65	0,86	0,58	2,4
4	104	0	0	0	0	0	583	56,59	85,47	34,65	0,88	0,59	2,46
5	102	0	0	0	0	0	587	56,98	85,47	35,03	0,87	0,58	2,42
6	102	103	62	75	0	28	587	56,59	84,31	35,42	0,85	0,57	2,36
7	102	99	61	73	0	28	597	56,98	85,85	36,57	0,85	0,57	2,34
8	100	98	61	73	0	0	593	57,36	86,62	35,8	0,87	0,58	2,41
9	101	97	61	73	0	0	593	57,75	85,85	35,42	0,86	0,58	2,4
10	101	99	61	73	0	0	595	58,9	87,39	37,34	0,84	0,56	2,33
11	100	102	61	74	0	28	591	59,67	87,39	38,5	0,8	0,55	2,27
12	101	106	61	76	0	29	589	60,44	87,78	41,19	0,76	0,53	2,13
13	101	111	61	77	0	0	587	61,21	88,55	40,81	0,77	0,53	2,15
14	102	114	61	78	0	0	589	61,6	88,55	41,19	0,76	0,53	2,14
15	101	114	61	78	0	0	593	60,83	88,16	38,11	0,8	0,56	2,31
16	101	113	62	79	0	0	595	60,83	88,16	38,11	0,81	0,56	2,31
17	100	113	62	79	0	0	597	61,21	88,16	39,27	0,78	0,54	2,24
18	100	112	62	78	0	30	601	62,37	88,16	38,88	0,77	0,55	2,25
19	99	112	62	78	0	29	599	62,75	88,55	41,19	0,75	0,53	2,14
20	100	113	62	79	0	0	599	61,98	87,01	40,04	0,75	0,53	2,17
21	100	113	62	79	0	0	607	60,44	85,47	38,5	0,77	0,54	2,21
22	98	113	63	79	0	0	626	59,67	84,7	36,96	0,79	0,56	2,28
23	95	114	63	80	0	0	644	60,06	84,31	36,96	0,77	0,55	2,26
24	93	113	63	79	0	0	638	60,83	85,47	38,5	0,76	0,54	2,21

Figura: Tenemos una matriz A de tamaño 2069×13, rango $r = 13$ y con 4807 datos corruptos.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	102	75,6822	45,2457	55,8828	7,3169	28	583	56,59	82,77	35,8	0,82	0,56	2,31
1	102	231,022	138,114	170,584	22,335	18,9869	582	56,21	83,16	35,42	0,83	0,57	2,33
2	103	0	0	0	0	0	574	55,82	83,54	34,65	0,86	0,58	2,4
3	104	0	0	0	0	0	576	55,82	83,93	34,65	0,86	0,58	2,4
4	104	137,799	82,3817	101,749	13,3223	11,3253	583	56,59	85,47	34,65	0,88	0,59	2,46
5	102	63,0464	37,6915	46,5526	6,09528	5,18158	587	56,98	85,47	35,03	0,87	0,58	2,42
6	102	103	62	75	8,40873	28	587	56,59	84,31	35,42	0,85	0,57	2,36
7	102	99	61	73	11,3955	28	597	56,98	85,85	36,57	0,85	0,57	2,34
8	100	98	61	73	3,23428	2,74946	593	57,36	86,62	35,8	0,87	0,58	2,41
9	101	97	61	73	0	0	593	57,75	85,85	35,42	0,86	0,58	2,4
10	101	99	61	73	0	0	595	58,9	87,39	37,34	0,84	0,56	2,33
11	100	102	61	74	0	28	591	59,67	87,39	38,5	0,8	0,55	2,27
12	101	106	61	76	7,75481	29	589	60,44	87,78	41,19	0,76	0,53	2,13
13	101	111	61	77	9,29157	7,89873	587	61,21	88,55	40,81	0,77	0,53	2,15
14	102	114	61	78	9,37005	7,96545	589	61,6	88,55	41,19	0,76	0,53	2,14
15	101	114	61	78	9,13949	7,76946	593	60,83	88,16	38,11	0,8	0,56	2,31
16	101	113	62	79	9,09134	7,72853	595	60,83	88,16	38,11	0,81	0,56	2,31
17	100	113	62	79	9,07352	7,71337	597	61,21	88,16	39,27	0,78	0,54	2,24
18	100	112	62	78	9,96725	30	601	62,37	88,16	38,88	0,77	0,55	2,25
19	99	112	62	78	9,95397	29	599	62,75	88,55	41,19	0,75	0,53	2,14
20	100	113	62	79	9,10634	7,74127	599	61,98	87,01	40,04	0,75	0,53	2,17
21	100	113	62	79	9,03841	7,68353	607	60,44	85,47	38,5	0,77	0,54	2,21
22	98	113	63	79	8,81261	7,49157	626	59,67	84,7	36,96	0,79	0,56	2,28
23	95	114	63	80	8,54208	7,2616	644	60,06	84,31	36,96	0,77	0,55	2,26
24	93	113	63	79	8,48492	7,213	638	60,83	85,47	38,5	0,76	0,54	2,21

Figura: Primera iteración con el modelo propuesto en [3][4][5][6], nos entrega 65 valores sin poder recuperar.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	102	75,6822	45,2457	55,8828	7,3169	28	583	56,59	82,77	35,8	0,82	0,56	2,31
1	102	231,022	138,114	170,584	22,335	18,9869	582	56,21	83,16	35,42	0,83	0,57	2,33
2	103	0	0	0	0	0	574	55,82	83,54	34,65	0,86	0,58	2,4
3	104	45,9552	26,3908	33,5574	7,40432	6,00667	576	55,82	83,93	34,65	0,86	0,58	2,4
4	104	137,799	82,3817	101,749	13,3223	11,3253	583	56,59	85,47	34,65	0,88	0,59	2,46
5	102	63,0464	37,6915	46,5526	6,09528	5,18158	587	56,98	85,47	35,03	0,87	0,58	2,42
6	102	103	62	75	8,40873	28	587	56,59	84,31	35,42	0,85	0,57	2,36
7	102	99	61	73	11,3955	28	597	56,98	85,85	36,57	0,85	0,57	2,34
8	100	98	61	73	3,23428	2,74946	593	57,36	86,62	35,8	0,87	0,58	2,41
9	101	97	61	73	14,6712	11,9019	593	57,75	85,85	35,42	0,86	0,58	2,4
10	101	99	61	73	0,57567	0,46701	595	58,9	87,39	37,34	0,84	0,56	2,33
11	100	102	61	74	5,6701	28	591	59,67	87,39	38,5	0,8	0,55	2,27
12	101	106	61	76	7,75481	29	589	60,44	87,78	41,19	0,76	0,53	2,13
13	101	111	61	77	9,29157	7,89873	587	61,21	88,55	40,81	0,77	0,53	2,15
14	102	114	61	78	9,37005	7,96545	589	61,6	88,55	41,19	0,76	0,53	2,14
15	101	114	61	78	9,13949	7,76946	593	60,83	88,16	38,11	0,8	0,56	2,31
16	101	113	62	79	9,09134	7,72853	595	60,83	88,16	38,11	0,81	0,56	2,31
17	100	113	62	79	9,07352	7,71337	597	61,21	88,16	39,27	0,78	0,54	2,24
18	100	112	62	78	9,96725	30	601	62,37	88,16	38,88	0,77	0,55	2,25
19	99	112	62	78	9,95397	29	599	62,75	88,55	41,19	0,75	0,53	2,14
20	100	113	62	79	9,10634	7,74127	599	61,98	87,01	40,04	0,75	0,53	2,17
21	100	113	62	79	9,03841	7,68353	607	60,44	85,47	38,5	0,77	0,54	2,21
22	98	113	63	79	8,81261	7,49157	626	59,67	84,7	36,96	0,79	0,56	2,28
23	95	114	63	80	8,54208	7,2616	644	60,06	84,31	36,96	0,77	0,55	2,26
24	93	113	63	79	8,48492	7,213	638	60,83	85,47	38,5	0,76	0,54	2,21

Figura: Cuarta iteración, 5 valores sin recuperar.

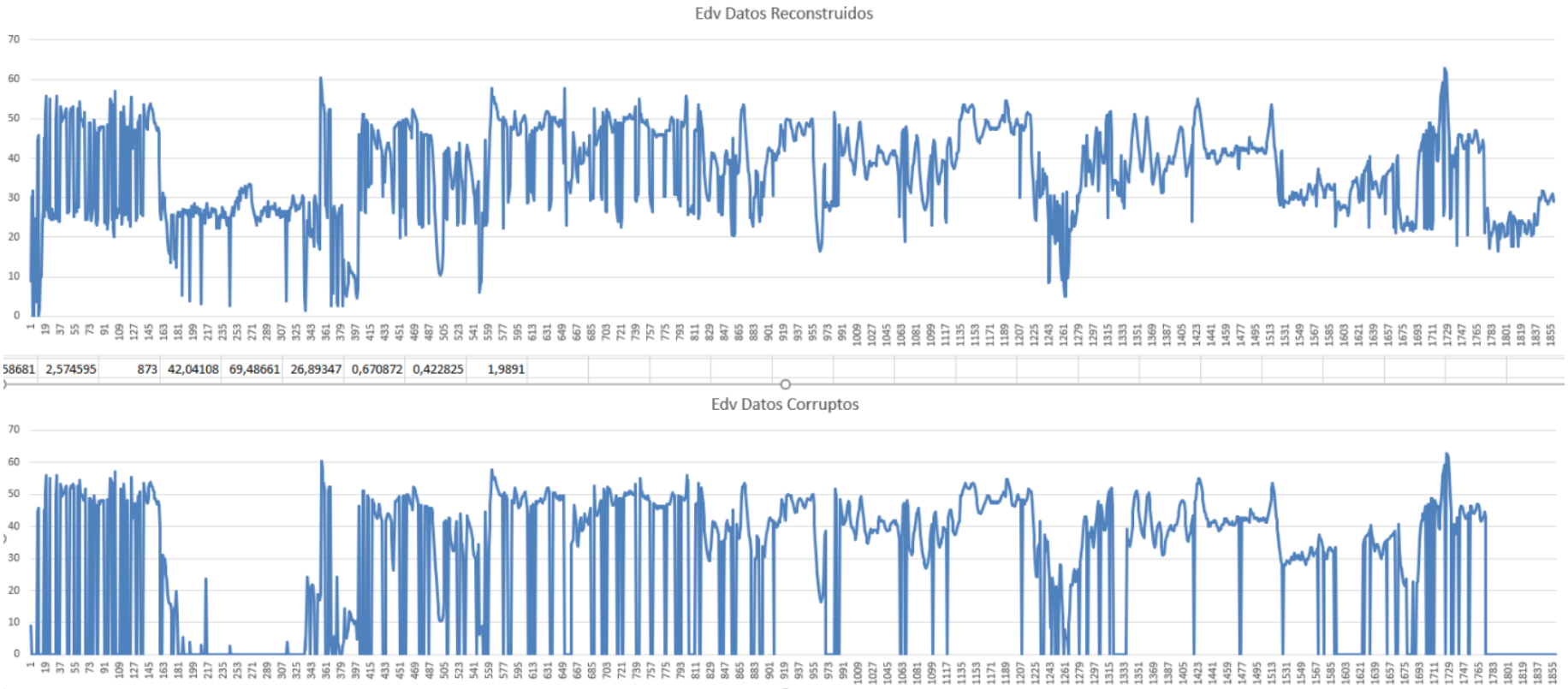

Figura: Graficas EDV.

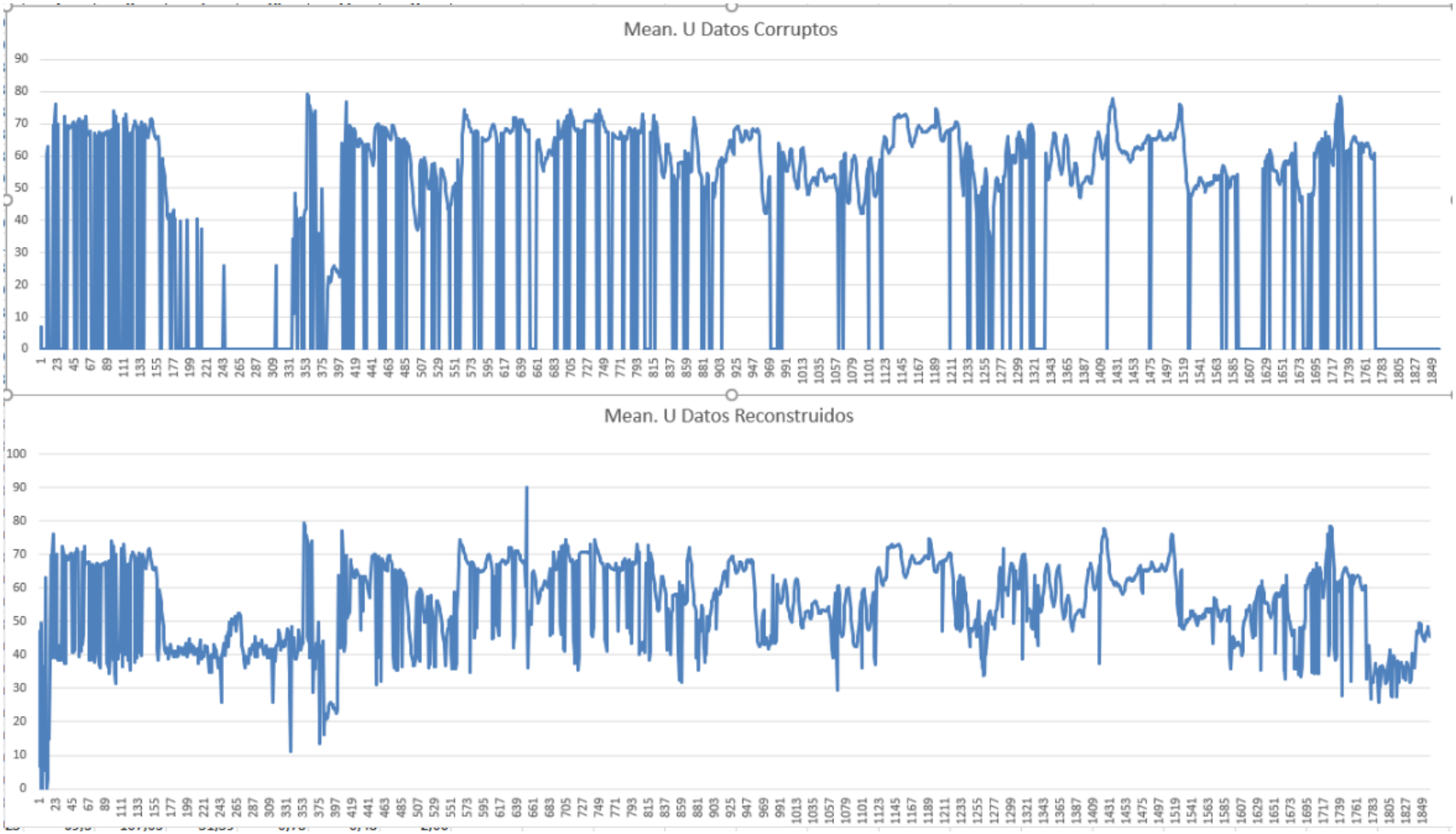

Figura: Graficas MeanU.

Dado que los datos utilizados son valores que representan diferentes actividades de una persona en particular en un protocolo médico, lo natural es que haya cierta tendencia en las gráficas reales. En las reconstrucciones planteadas, se muestra que si tienen ciertas coherencias con lo que está pasando en la persona.