Tarea 2

Pregunta 1. (20 %)

(a) Sea $\mathcal{X} = \overline{\mathcal{B}_{\|\cdot\|_1}}(0,1)$, la bola unitaria centrada en el origen con respecto a la norma ℓ_1. De un algoritmo para calcular la proyección de $x \in \mathbb{R}^d$ sobre $\mathcal{X}$. Su algoritmo debe correr a lo más en tiempo $O(d \log d)$.
Indicación. Primero reduzca el problema al caso $x \geq 0$. Luego escriba las condiciones de KKT. Busque un método rápido para determinar los multiplicadores de Lagrange del problema.

(b) Calcule el conjunto de puntos extremos de $\mathcal{W} := \overline{\mathcal{B}_{\|\cdot\|_{nuc}}}(0,1)$ en el espacio $\mathcal{S}^d$ de matrices simétricas.
Indicación. Use la desigualdad de Fan y sus consecuencias.

(c) De un algoritmo para calcular la proyección de una matriz simétrica $X \in \mathcal{S}^d$ sobre el conjunto $\mathcal{W}$ definido arriba, con respecto a la norma de Frobenius. Evalúe la eficiencia de su algoritmo.
Indicación. Use la desigualdad de Fan y sus consecuencias. En caso de utilizar subrutinas de álgebra lineal, puede indicar su eficiencia incluyendo una referencia de la subrutina junto con su complejidad aritmética.

Pregunta 2. (20 %) Consideremos el espacio vectorial de matrices $\mathbb{R}^{n\times d}$ (por simplicidad, $n \geq d$) dotado del producto interno de Frobenius $\langle A, B\rangle_F := \mathrm{tr}(A^\top B)$. Consideramos la norma nuclear sobre $\mathbb{R}^{n\times d}$, definida como

$$\|A\|_{nuc} = \sum_{j=1}^{d} \sigma_j(A),$$

donde $\sigma(A) \in \mathbb{R}^d$ es el vector de valores singulares de A en orden decreciente. Sea $\mathcal{Z} = \overline{\mathcal{B}_{\|\cdot\|_{nuc}}}(0,1)$ la bola unitaria centrada en el origen.

(a) Determine el conjunto de puntos extremos de $\mathcal{Z}$.

(b) Dada $A \in \mathbb{R}^{n\times d}$, proponga un algoritmo que calcule la proyección de A sobre $\mathcal{Z}$ con respecto a la norma de Frobenius.

Pregunta 3. (20 %) Sea $\mathcal{X} \subseteq \mathbf{E}$ convexo y compacto. Dada $f : \mathbf{E} \mapsto \mathbb{R}$ convexa y diferenciable sobre $\mathcal{X}$, se define la constante de curvatura de f como

$$C_f := \sup_{\substack{x,z\in\mathcal{X} \\ \gamma\in[0,1] \\ y=(1-\gamma)x+\gamma z}} \frac{2}{\gamma^2}[f(y) - f(x) - \langle \nabla f(x), y - x\rangle].$$

(a) Si $\mathbf{E} = \mathbb{R}^d$, $A \in \mathbb{R}^{m\times d}$ y $b \in \mathbb{R}^d$, calcule C_f para $f(x) = \frac{1}{2}[\langle A^\top Ax, x\rangle - \langle b, x\rangle]$ (aquí, $\mathcal{X} \subseteq \mathbb{R}^d$ es todavía un convexo y compacto cualquiera).

(b) Pruebe que para cualquier norma $\|\cdot\|$ sobre $\mathbf{E}$, si $L_1 \geq 0$ es tal que $f \in \mathcal{F}^1_{\mathbf{E},\|\cdot\|}(L_1)$, entonces

$$C_f \leq L_1 \cdot \operatorname{diam}_{\|\cdot\|}(\mathcal{X})^2.$$

Se presenta el método de gradiente condicional.

Algoritmo 1 Método de Gradiente Condicional

Input: $x_0 \in \mathcal{X}$
for $k = 0, \ldots, K-1$ **do**

$$\begin{aligned} v_k &\in \arg\min_{v \in \mathcal{X}} \langle \nabla f(x_k), v \rangle \quad (P_t) \\ \eta_k &= \frac{2}{k+2} \\ x_{k+1} &= (1-\eta_k)x_k + \eta_k v_k. \end{aligned}$$

end for
return x_K

(c) Pruebe que la iteración del método de gradiente condicional satisface

$$f(x_{k+1}) \leq f(x_k) + \eta_k \langle \nabla f(x_k), v_k - x_k \rangle + \frac{\eta_k^2 C_f}{2}$$

(d) Concluya que el gap de optimalidad $\gamma_k = f(x_k) - f(x^*)$ satisface

$$\gamma_{k+1} \leq \left(1 - \frac{2}{k+2}\right)\gamma_k + \frac{2C_f}{(k+2)^2}$$

(e) Pruebe que el método de gradiente condicional satisface

$$f(x_K) - \min_{x \in \mathcal{X}} f(x) \leq \frac{2C_f}{K+2}.$$

Parte Computacional (40 %)

El objetivo de la parte computacional es considerar un problema de completación de matrices. Consideramos el espacio $(\mathbb{R}^{n\times d}, \|\cdot\|_{nuc})$ como se definió en la Pregunta 2, y usamos el producto interno de Frobenius. Sobre este espacio, consideramos el problema

$$\min\Big\{F(X) := \sum_{(i,j)\in\Omega} (x_{ij}-x^*_{ij})^2 : \|X\|_{nuc}\leq R\Big\} \qquad \text{(Nuc-Norm-Min)}$$

donde $R>0$ es un parámetro ajustable, y $\Omega\subseteq[n]\times[d]$ es un conjunto de coordenadas donde se observa parcialmente una matriz $X^*\in\mathcal{X}$. Nuestro objetivo es estimar esta matriz desde las observaciones parciales.

1. Implemente el método del gradiente proyectado con paso fijo
$$x_{k+1}=\Pi_{\mathcal{X}}[x_k-\eta\nabla f(x_k)]$$
para resolver (P). Para esto, provea una cota sobre la constante de suavidad de F (en la norma de Frobenius).
Indicación. Si bien no hemos visto en clases todavía, las garantías del método del gradiente proyectado son análogas a las tasas de convergencia para el método del gradiente irrestricto.

2. Implemente el método del gradiente proyectado con backtracking.

3. Implemente el método de gradiente condicional para resolver (P). No es necesario que lo demuestre, pero usando la Pregunta 2, parte (a), se deduce que el óptimo de (P_t) se obtiene usando el par de vectores singulares principales de $-\nabla f(X_k)$; es decir, si $-\nabla f(X_k)=\sum_{j=1}^d \sigma_j u_j v_j^\top$ (con $\sigma_1\geq\ldots\geq\sigma_d\geq 0$), entonces $V_k=u_1v_1^\top$. Este par singular puede obtenerse rapidamente con el método de Lanczos (una aceleración sobre el método de la potencia), que puede ejecutar en python a través de la función `svds` (`https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.svds.html`).

4. Pruebe su algoritmo en una matriz aleatoria de bajo rango.
```
np.random.seed(47)
rank = 10
users, movies = 1000, 500
# Generate random low-rank matrices U and V
U = np.random.rand(users, rank)
V = np.random.rand(rank, movies)
# Compute the product to get the low-rank matrix X
Full_Data = np.dot(U, V)
ratings_fraction = 0.1
Mask_Train = np.random.binomial(1,ratings_fraction, size = (users,movies))
Masked_Train_Data = np.multiply(Full_Data,Mask_Train)
```

```
Mask_Validation = np.random.binomial(1,ratings_fraction, size =
(users,movies))
Masked_Validation_Data = np.multiply(Full_Data,Mask_Validation)
```

Llamamos *datos de entrenamiento* al conjunto de datos con el que resolvemos (Nuc-Norm-Min) (es decir, los coeficientes la matriz enmascarada) y *datos de validación* a un conjunto de datos que usamos para comparar el poder predictivo de distintos modelos obtenidos. En este caso, ambos conjuntos de datos corresponden a (los coeficientes no nulos) de las matrices `Masked_Train_Data` y `Masked_Validation_Data`, respectivamente.

Ajuste el parámetro R resolviendo (Nuc-Norm-Min) en los datos de entrenamiento con valores $R = 2^j$ con $j = 3, \ldots, 7$, y escoja el modelo con el mejor error en términos del error sobre el conjunto de validación.

5. Pruebe ahora su método de gradiente condicional en las siguientes instancias del conjunto de datos *MovieLens* (`https://grouplens.org/datasets/movielens/`). Cada instancia contiene una matriz parcialmente observada de evaluaciones de películas por usuarios (con notas 1-5), y el objetivo es inferir el resto de las preferencias no observadas en la matriz con el fin de producir recomendaciones a los usuarios.

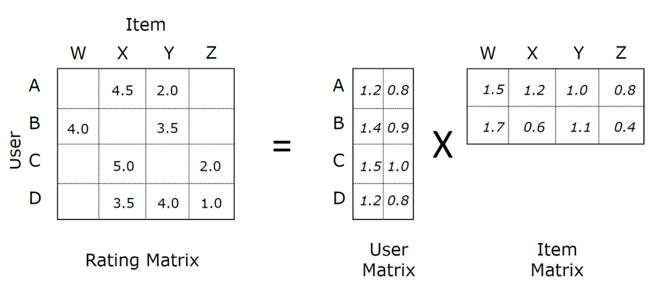

Figura 1: Si bien existen infinitas matrices que interpolan estos datos, el modelo (Nuc-Norm-Min) promueve a aquellas matrices que tienen bajo rango, lo cual corresponde con una representación simple de las preferencias .

Dataset	# Users	# Movies	# Ratings
MovieLens 100k	943	1,682	100,000
MovieLens 1M	6,040	3,900	1,000,209
MovieLens 10M	82,248	10,681	10,000,054
MovieLens 20M	138,493	27,278	20,000,263

Figura 2: Tamaños de los conjuntos de datos de las instancias de MovieLens.

Corra sus algoritmos con un límite de tiempo de 10 minutos, y reporte como falla cuando el algoritmo alcance este tiempo. Para el ajuste del parámetro, considere $R = 2^j$ con $j = 5, \ldots, 10$, y resuelva (Nuc-Norm-Min) sobre un conjunto de datos de entrenamiento, y escoja el mejor modelo en términos del error cuadrático medio sobre el conjunto de validación. Para obtener estos conjuntos, haga lo siguiente:

- Para la base de datos de 100,000 evaluaciones, los datos de entrenamiento están en el archivo `u1.base`. Los datos de validación están en el archivo `u1.test`.
- Para los conjuntos de datos de 1,000,000, 10,000,000 y 20,000,000 de evaluaciones use el script `split_ratings.sh` para obtener los conjuntos de entrenamiento (Nuc-Norm-Min)) y validación.

6. Haga una tabla con una comparación del progreso de los 3 algoritmos propuestos en función del tiempo, para la instancia más grande de *MovieLens* que pueda resolver. Para esto se recomienda usar las funciones del módulo %*time* en Python, `https://docs.python.org/3/library/time.html`.