Tarea 1

Instrucciones: Resuelva las siguientes preguntas y envíe el procedimiento realizado y las respuestas en un Jupyter Notebbok al correo andres.garcia@itam.mx a más tardar el miércoles 24 de mayo de 2023 a las11:59pm.

1.- Escribe una función que reciba como entrada un array de numpy y te regrese un array con el índice de cada elemento ordenado del array original. Por ejemplo:

Entrada: [9, 4, 12, 1]

Salida: [2, 1, 3, 0]

2.- Escribe una función que tome como entrada una matriz de numpy y calcule la suma de los elementos que se encuentren en la diagonal de la matriz.

3.- Implementación del algoritmo Gram-Schmidt
El método de Gram-Schmidt nos permite construir una base ortogonal a partir de una base cualquiera.

La proyección de un vector v sobre un vector u está dada por:

$$\operatorname{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u},$$

El algoritmo de Gram-Schmidt consiste en:

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{v}_2 - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{v}_3 - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \operatorname{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
\mathbf{u}_4 &= \mathbf{v}_4 - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \operatorname{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \operatorname{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 &= \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|} \\
&\vdots & &\vdots \\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1} \operatorname{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}.
\end{aligned}$$

Si tenemos dos vectores v1=[3,1] y v2=[2,2], ¿cuáles son los vectores resultantes de aplicar el algoritmo de Gram-Schmidt? Comprueba que los vectores resultantes son ortogonales.

4.- Suponiendo que contamos con los siguiente YTM de bonos (tasa fija, freq pago cupón **semestral**), obtén los factores de descuento y las tasas cero en composición continua que resultan del bootstrapping.

Tasa cupón	6%	5%	7%	5%
YTM	5%	6%	7%	8%
Vencimiento	6M	1Y	1.5Y	2Y

5.- De acuerdo con la curva anterior, ¿cuál sería el precio de un bono de tasa fija, tasa cupón de 10%, frecuencia de pago trimestral, y vencimiento en 15M?

6.- En algunos problemas no es necesario encontrar todos los eigenvalores y eigenvectores de una matriz, si no que vamos a estar interesados únicamente el eigenvalor más grande y su correspondiente eigenvector. En estos casos se puede aplicar el **método de las potencias**.

Supongamos que tenemos una matriz *A* de *nxn* que tiene n eigenvalores $\lambda_1, \lambda_2, \ldots, \lambda_n$ con sus correspondiente eigenvectores $v_1, v_2, \ldots, v_n$ que son linealmente independientes. Podríamos ordenar los valores absolutos de los eigenvalores de tal forma que:

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$$

Dado que los eigenvectores son linealmente independientes significa que cualquier vector en ese espacio vectorial se puede escribir como una combinación lineal de los vectores base. Entonces cualquier vector x_0 lo podemos expresar como una combinación lineal de los eigenvectores:

$$x_0 = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$$

Donde vamos a restringir que c_1 sea diferente de *0*. Al multiplicar ambos lados de la ecuación por *A* obtenemos que:

$$Ax_0 = c_1 A v_1 + c_2 A v_2 + \cdots + c_n A v_n$$

Por la ecuación característica de eigenvalores y eigenvectores sabemos que:

$$Av_i = \lambda_i v_i$$

Y sustituyendo en la ecuación anterior obtenemos que:

$$Ax_0 = c_1 \lambda_1 v_1 + c_2 \lambda_2 v_2 + \cdots + c_n \lambda_n v_n$$

Si factorizamos el término $c_1 \lambda_1$ de la ecuación anterior obtenemos:

$$Ax_0 = c_1 \lambda_1 \left[v_1 + \frac{c_2 \lambda_2}{c_1 \lambda_1} v_2 + \cdots + \frac{c_n \lambda_n}{c_1 \lambda_1} v_n \right] = c_1 \lambda_1 \boldsymbol{x_1}$$

En donde x_1 es un nuevo vector $x_1 = \left[v_1 + \frac{c_2 \lambda_2}{c_1 \lambda_1} v_2 + \cdots + \frac{c_n \lambda_n}{c_1 \lambda_1} v_n \right]$, y con este nuevo vector terminaríamos la primera iteración del método de las potencias. La segunda iteración la empezaríamos al multiplicar *A* por el vector x_1 que obtuvimos y nos da como resultado:

$$Ax_1 = \lambda_1 \left[v_1 + \frac{c_2 \lambda_2^2}{c_1 \lambda_1^2} v_2 + \cdots + \frac{c_n \lambda_n^2}{c_1 \lambda_1^2} v_n \right] = \lambda_1 \boldsymbol{x_2}$$

En donde x_2 es un nuevo vector $x_2 = v_1 + \frac{c_2\lambda_2^2}{c_1\lambda_1^2}\, v_2 + \cdots + \frac{c_n\lambda_n^2}{c_1\lambda_1^2} v_n$, y si continuamos multiplicando la matriz A a cada nuevo vector que obtengamos, en la k-ésima iteración tendríamos que:

$$Ax_{k-1} = \lambda_1 \left[v_1 + \frac{c_2\lambda_2^k}{c_1\lambda_1^k}\, v_2 + \cdots + \frac{c_n\lambda_n^k}{c_1\lambda_1^k} v_n \right] = \lambda_1 \boldsymbol{x_k}$$

Dado que λ_1 es el eigenvalor más grande el cociente de cada eigenvalor $\frac{\lambda_i}{\lambda_1}$ siempre será menor a 1. Con un número de iteraciones *k* lo suficientemente grande el factor $\left(\frac{\lambda_n}{\lambda_1}\right)^k$ será muy cercano a 0 por lo que todos los términos que contienen este factor se pueden descartar conforme vaya incrementando k y nos quedaría que:

$$Ax_{k-1} = \lambda_1 \left[v_1 + \frac{c_2\lambda_2^k}{c_1\lambda_1^k}\, v_2 + \cdots + \frac{c_n\lambda_n^k}{c_1\lambda_1^k} v_n \right]$$

$$Ax_{k-1} \approx \lambda_1 v_1$$

Al implementar este método comúnmente debemos normalizar el vector resultante x_k en cada iteración, y esto lo hacemos dividiendo el vector entre el elemento más grande del vector. Por ejemplo:

$$x_k = \begin{bmatrix} 2 \\ 1 \\ 8 \end{bmatrix} = 8 \begin{bmatrix} 0.25 \\ 0.125 \\ 1 \end{bmatrix}$$

Por lo que el vector con el que empezaríamos la siguiente iteración sería *[0.25, 0.125, 1]*.

Utilizando el método de las potencias encuentra el eigenvalor más grande de la siguiente matriz:

$$A = \begin{bmatrix} 4 & 4 & 6 \\ 5 & 6 & 1 \\ 9 & 7 & 8 \end{bmatrix}$$

7.- Se puede considerar que el algoritmo más sencillo para clasificación y regresión es el “k nearest neighbors”. El input del modelo serán los datos conocidos y el output dependerá del problema a resolver. Para clasificación, el modelo asignará la clase más popular entre los k vecinos. Cuando k=1 entonces simplemente se asignará la clase del vecino más cercano.

Utiliza el modelo de k nearest neighbors y el dataset en https://raw.githubusercontent.com/Apgt512/Modulo_II_LinAlg_Prob/main/credit_rating_data.csv para resolver lo siguiente:

a) Implementa una función con el algoritmo de k nearest neighbors, que encuentre los k vecinos más cercanos y regrese la clase más repetida en esos k vecinos. Es decir deberás calcular la distancia de tu vector a predecir respecto a cada uno de los datos, y seleccionar los k datos más cercanos. En este caso queremos identificar a qué rating pertenecería una empresa dados sus datos financieros por lo que el output del modelo será el rating que se repita más en los k vecinos más cercanos.

b) Utilizando la función ya implementada, ¿qué rating le asignarías a una empresa que tiene las siguientes características? Utiliza k=1, k=5, k=20 y compara los resultados