"El saber de mis hijos
hará mi grandeza"

Universidad De Sonora

División de Ciencias Exactas y Naturales

Licenciatura En Física

Física Computacional I

Actividad 7

"Álgebra Lineal con Python"

Ismael Espinoza Arias

Profesor Carlos Lizárraga Celaya

Hermosillo, Sonora a marzo 07 de 2021

1 Introducción

En esta actividad trabajamos con el tema de álgebra lineal. Repasamos lo visto anteriormente en los cursos de álgebra lineal ahora implementando problemas computacionales con Python en la actividad de número 7. Hicimos uso de nuevas bibliotecas en Python, para poder realizar los ejercicios computacionales de nuestro nuevo tema para poder poner en práctica la teoría vista durante toda la semana. Es por eso que en esta sección trabajamos con los temas de matrices, determinantes, matrices cuadradas, el teorema de Caley-Hamilton, ecuaciones características, polinomios, resolución de sistema de ecuaciones, aplicaciones del Gauss-Jordan y eliminación Gaussiana, soluciones entre otras funciones elementales del álgebra lineal.

2 Comentarios

Para poder desarrollar las actividades en esta sección, debimos ver la teoría semanal impartida por el docente, con ella y con la resolución de unos ejercicios fue como desarrollamos las habilidades requeridas en esta actividad. Primero encontramos el uso de las nuevas bibliotecas que en nuestro caso son: Scipy.linalg, Numpy.linalg y el uso de unas bibliotecas que ya estábamos previamente usando pero que resultan muy efectivas, Numpy y Matplotlib.pyplot.

Después nos encontramos con los ejercicios compuatcionales, como lo son la definición de matrices con Python, la demostración del teorema de Caley-Hamilton, la resolución de un sistema de ecuaciones por el método de eliminación gaussiana, encontrar los eigenvectores y los eigenvalores de una matriz, realizar una interpolación polinómica y por último una regresión lineal de las temperaturas ya antes utilizadas en el curso.

Una de las herramientas matemáticas más utilizadas en machine learning y data mining es el Álgebra lineal, por lo tanto, si queremos incursionar en el fascinante mundo del aprendizaje automático y el análisis de datos es importante reforzar los conceptos que forman parte de sus cimientos.

Como ya sabemos el Álgebra lineal es una rama de las matemáticas que es sumamente utilizada en el estudio de una gran variedad de ciencias, como ser, ingeniería, finanzas, investigación operativa, entre otras. Es una extensión del álgebra que aprendemos en la escuela secundaria, hacia un mayor número de dimensiones; en lugar de trabajar con incógnitas a nivel de escalares comenzamos a trabajar con matrices y vectores.

3 Desarrollo de la actividad

Empezamos importando nuestras bibliotecas:

```
#ESPINOZA ARIAS ISMAEL ACTIVIDAD 7 Álgebra Lineal con Python
# 3 de marzo de 2021
# Cargamos las bibliotecas para trabajar en Algebra Lineal

import numpy as np
import scipy.linalg as la
from numpy.linalg import matrix_power as mpow

import matplotlib.pyplot as plt
%matplotlib inline
```

Procediamos a definir las matrices con las que hibamos a operar:

Ejercicio 1. Defina las siguientes matrices.

Definimos nuestras matrices Para ello definiremos la matriz A como:

$$A = \begin{bmatrix} 1 & 3 \\ -1 & 7 \end{bmatrix}$$

```
#Definimos nuestra amtriz A:
A = np.array([[1,3],[-1,7]])
print("Matriz A = ")
print(A)
```

```
Matriz A =
[[ 1  3]
 [-1  7]]
```

Después solamente, definimos de la misma forma cada matriz y con estas fuimos operando.
En el ejercicio 2, nos encontramos con la demostración del teorema de Caley-Hamilton, donde tenemos lo siguiente:

Ejercicio 2. Los polinomios característicos y el Teorema de Cayley-Hamilton.

El polinomio característico de una matriz M esta dado en general por la ecuación

$$\det(M - \lambda I) = 0$$

Para una matriz cuadrada M (2x2), el polinomio característico se puede escribir como

$$P_2(\lambda) = \det(M) - \mathrm{tr}(M)\,\lambda + \lambda^2$$

y para una matriz cuadrada M (3x3), el polinomio característico resulta ser

$$P_3(\lambda) = \frac{1}{6}[\mathrm{tr}^3(M) + 2\,\mathrm{tr}(M^3) - 3\,\mathrm{tr}(M)\mathrm{tr}(M^2)] - \frac{1}{2}[\mathrm{tr}^2(M) - \mathrm{tr}(M^2)]\lambda + \mathrm{tr}(M)\lambda^2 - \lambda^3$$

El Teorema de Cayley-Hamilton nos dice que una matriz cuadrada M (2x2), satisface el polinomio característico

$$P_2(M) = \det(M) - \mathrm{tr}(M)M + M^2$$

y en general cualquier matriz cuadrada M (nxn), satisface $P_n(M)$.

Demuestre esto para cualquier matriz M (2x2), ($\det(M) \neq 0$).

Para la demostración de este teorema, lo que hicimos fue hacerlo de forma general, y después con un ejemplo en especifico para confirmar que se cumple apara cualquier matriz cuadrada con determinante distinto de 0.

$$M, \det(M - \lambda I) = 0$$

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \ \det(M - \lambda I) = 0$$

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

$$\det \begin{bmatrix} a - \lambda & b \\ c & d - \lambda \end{bmatrix} = (a - \lambda)(d - \lambda) - (c)(b) = 0$$

$$\lambda^2 - \lambda(a + d) + (ad - cb)I = 0$$

$$M^2 = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a^2+bc & ab+bd \\ ac+dc & cb+d^2 \end{bmatrix}$$

$$\begin{bmatrix} a^2+bc & ab+bd \\ ac+dc & cb+d^2 \end{bmatrix} - \begin{bmatrix} a & b \\ c & d \end{bmatrix} (a+d) + (ad-cb) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} a^2+bc & ab+bd \\ ac+dc & cb+d^2 \end{bmatrix} - \begin{bmatrix} a^2+ad & ab+db \\ ac+dc & cd+d^2 \end{bmatrix} + \begin{bmatrix} ad-ad & 0 \\ 0 & ad-cb \end{bmatrix} =$$

$$\begin{bmatrix} -ad+bc & 0 \\ 0 & -ad+bc \end{bmatrix} + \begin{bmatrix} ad-bc & 0 \\ 0 & ad-bc \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Con esta demostración, ya solo nos queda el ejemplo de como aplicarlo en un caso específico, en este caso la matriz M.

$$M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
#Definimos nuestra amtriz M:
M = np.array([[1,2],[3,4]])
print("Matriz A = ")
print(M)
```

```
Matriz A =
[[1 2]
 [3 4]]
```

ıos si nuestra matriz tiene un determinante distinto de 0.

```
# Determinante de la matriz M
print(M)
print(la.det(M))
```

```
[[1 2]
 [3 4]]
-2.0
```

Remplazamos con nuestra matriz:

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} - 5\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} - 2\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = 0$$

```
[ ]  Procedemos a solucionarlo con python:
```

```
[ ]  R = M@M - 5*M - 2*I
     print(R)
```

```
[[0 0]
 [0 0]]
```

Por lo tanto queda demostrado el teorema anterior.

Con la siguiente ejercicio solucionamos un sistema de ecuaciones por medio del método de eliminación Gaussiana.

Ejercicio 3: Resuelva el sistema de ecuaciones.

$$\begin{aligned} x - 3y + z &= 1\\ 3x - 4y + z &= 5\\ 2y - z &= 0 \end{aligned}$$

de dos formas, utilizando el Método de Eliminación Gaussiana y la utilzando la función scipy.linalg.solve()

Definiremos 3 funciones que nos ayudaran a resolver este sistema:

```python
# Implementación del método de eliminación Gaussiana
# Definimos 3 funciones

# Intercambio de posición de renglones.
def switch_rows(A,i,j):
    "Intercambiar renglones i y j en la matriz A."
    n = A.shape[0]
    E = np.eye(n)
    E[i,i] = 0
    E[j,j] = 0
    E[i,j] = 1
    E[j,i] = 1
    return E @ A

# Multiplicar un renglón por una constante no nula.
def scale_row(A,k,i):
    "Multiplicar el renglón i por k en la matriz A."
    n = A.shape[0]
    E = np.eye(n)
    E[i,i] = k
    return E @ A

# Sumar un múltiplo de un reglon a otro renglón.
def add_row(A,k,i,j):
    "Sumar k veces el renglón j al renglón i en la matriz A."
    n = A.shape[0]
    E = np.eye(n)
    if i == j:
        E[i,i] = k + 1
    else:
        E[i,j] = k
    return E @ A

#Definimos nuestra matriz
```

```
#Definimos nuestra amtriz G:
G = np.array([[1,-3,1,1],[3,-4,1,5],[0,2,-1,0]])
print("Matriz G = ")
print(G)
```

```
Matriz G =
[[ 1 -3  1  1]
 [ 3 -4  1  5]
 [ 0  2 -1  0]]
```

Emepezamos la eliminasion Gaussiana:

```
[ ] G1 = (scale_row(G,3,0))
    print(scale_row(G,3,0))
```

```
[[ 3. -9.  3.  3.]
 [ 3. -4.  1.  5.]
 [ 0.  2. -1.  0.]]
```

```
[ ] G2 = add_row(G1,-1,1,0)
    print(G2)
```

```
[[ 3. -9.  3.  3.]
 [ 0.  5. -2.  2.]
 [ 0.  2. -1.  0.]]
```

```
[ ] G3 = scale_row(G2,2,1)
    G4 = scale_row(G3,5,2)
    print(G4)
```

```
[[ 3. -9.  3.  3.]
 [ 0. 10. -4.  4.]
 [ 0. 10. -5.  0.]]
```

Lllegamos al final, al siguiente resulatdo:

$$X = 3, Y = 2, Z = 4$$

Pasamos al sigueinte ejercicio.

Ejercicio 4: Dadas las siguientes matrices B_1, B_2 y B_3

(Ejemplos del artículo de Wikipedia sobre Eigenvalores y Eigenvectores)

$$B_1 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 4 \\ 0 & 4 & 9 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$B_3 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

Aquí, haremos cosas similares como en los ejercicios anteriores, donde vamos a definir nuestras amtrices y procedemos a usar las fórmulas de nuestars bibliotecas para consegiuir por pedido.

```
#Definimos nuestra amtriz B1:
B1 = np.array([[2,0,0],[0,3,4],[0,4,9]])
print("Matriz B1 = ")
print(B1)
```

```
Matriz B1 =
[[2 0 0]
 [0 3 4]
 [0 4 9]]
```

```
#Definimos nuestra amtriz B2:
B2 = np.array([[0,1,0],[0,0,1],[1,0,0]])
print("Matriz B2 = ")
print(B2)
```

```
Matriz B2 =
[[0 1 0]
 [0 0 1]
 [1 0 0]]
```

```
#Definimos nuestra amtriz B3:
B3 = np.array([[2,0,0,0],[1,2,0,0],[0,1,3,0],[0,0,1,3]])
print("Matriz B3 = ")
print(B3)
```

```
Matriz B3 =
[[2 0 0 0]
 [1 2 0 0]
 [0 1 3 0]
 [0 0 1 3]]
```

```
[ ]  # Aplicamos la biblioteca scipy.linalg.eig()
     eigvals, eigvecs = la.eig(B1)
     print('Eigenvalores:', eigvals)
     print('Eigenvectores:', eigvecs)
```

```
Eigenvalores: [11.+0.j  1.+0.j  2.+0.j]
Eigenvectores: [[ 0.          0.          1.        ]
 [ 0.4472136   0.89442719  0.        ]
 [ 0.89442719 -0.4472136   0.        ]]
```

```
[ ]  # Ahora vamos con B2
     eigvals, eigvecs = la.eig(B2)
     print('Eigenvalores:', eigvals)
     print('Eigenvectores:', eigvecs)
```

```
Eigenvalores: [-0.5+0.8660254j -0.5-0.8660254j  1. +0.j       ]
Eigenvectores: [[ 0.57735027+0.j   0.57735027-0.j  -0.57735027+0.j ]
 [-0.28867513+0.5j -0.28867513-0.5j -0.57735027+0.j ]
 [-0.28867513-0.5j -0.28867513+0.5j -0.57735027+0.j ]]
```

```
[ ]  # Ahora vamos con B3
     eigvals, eigvecs = la.eig(B3)
     print('Eigenvalores:', eigvals)
     print('Eigenvectores:', eigvecs)
```

```
Eigenvalores: [3.+0.j 3.+0.j 2.+0.j 2.+0.j]
Eigenvectores: [[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  2.56395025e-16]
 [ 0.00000000e+00  0.00000000e+00  5.77350269e-01 -5.77350269e-01]
 [ 0.00000000e+00  6.66133815e-16 -5.77350269e-01  5.77350269e-01]
 [ 1.00000000e+00 -1.00000000e+00  5.77350269e-01 -5.77350269e-01]]
```

Ahora pasamos al ejercicio 5.

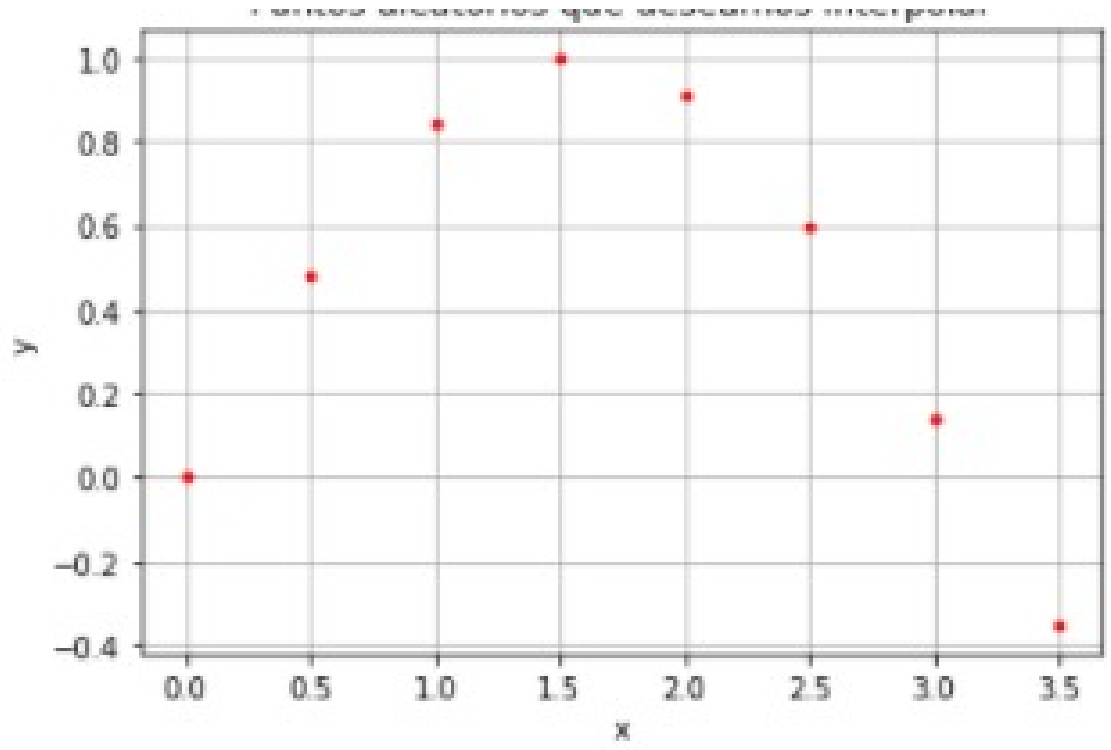
Ejercicio 5: Se tienen los siguientes 8 puntos:

$$
\begin{array}{c}
(x, y) \\
(0.0, 0.0), \\
(0.5, 0.47942), \\
(1.0, 0.84147), \\
(1.5, 0.99749), \\
(2.0, 0.90930), \\
(2.5, 0.59847), \\
(3.0, 0.14112), \\
(3.5, -0.35078)
\end{array}
$$

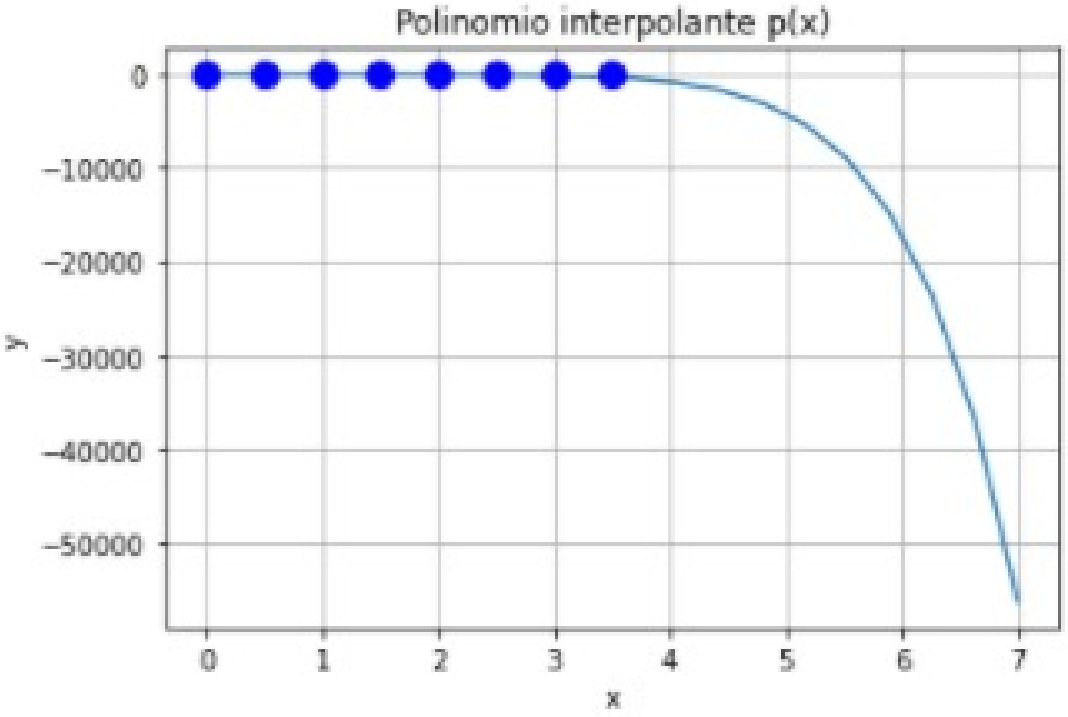
Encuentre un polinomio interpolante $p(x)$ que pase por los 8 puntos.

Graficamos nuestros datos y después vemos las gráficas:

```
#Ahora graficamos nuestros datos:
# graficamos los puntos en rojo con puntos ('r.')
N = 8
x = np.arange(0,4)
y = np.arange(-0.5,1)
plt.plot(0,0,'r.')
plt.plot(0.5,0.47942,'r.')
plt.plot(1,0.84147,'r.')
plt.plot(1.5,0.99749,'r.')
plt.plot(2,0.90930,'r.')
plt.plot(2.5,0.59847,'r.')
plt.plot(3,0.14112,'r.')
plt.plot(3.5,-0.35078,'r.')
plt.grid()
plt.title('Puntos aleatorios que deseamos interpolar')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Después de la interpolación, llegamos al siguiente gráfico:



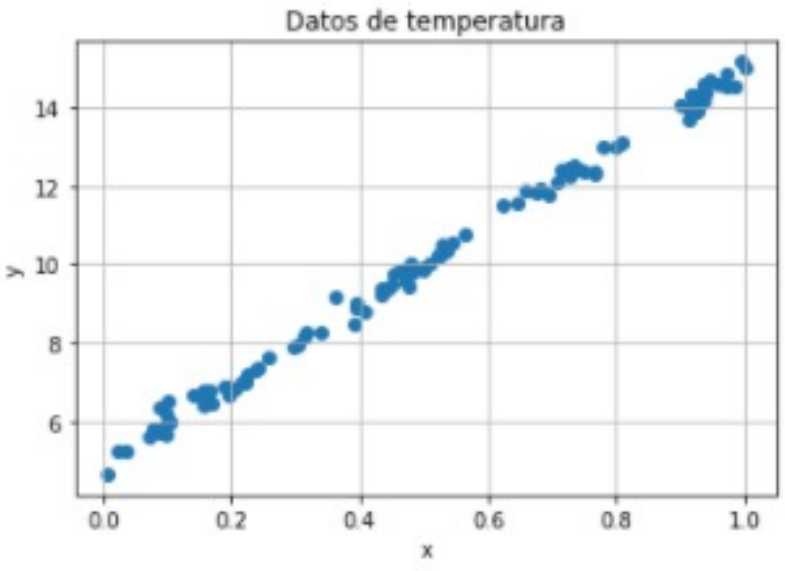
Llegamos al último ejercicio de la actividad 7, que consiste en hacer una regresión lineal con nuestros datos de las temepraturas anteriores.

Ejercicio 6.

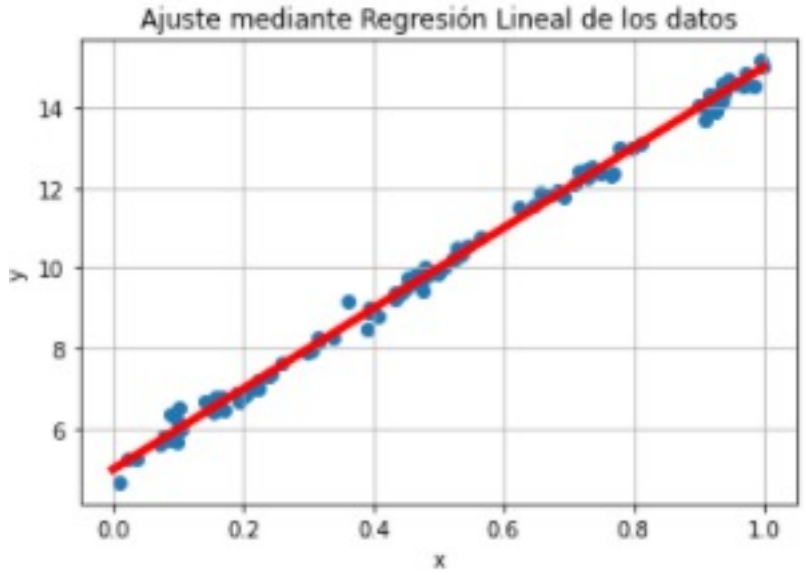
Regrese a su modelo de análisis de series de tiempo. Haga una descomposición de su serie de tiempo de las temperaturas (Tmax y Tmin), y realice una regresión lineal sobre la serie de Tendencia de Temp $(T = a_0 + a_1 t)$. Encuentre si la pendiente es positiva o negativa de la tendencia (signo de a_1), es decir si las Temperaturas (Tmax, Tmin) están subiendo o bajando en el periodo analizado.

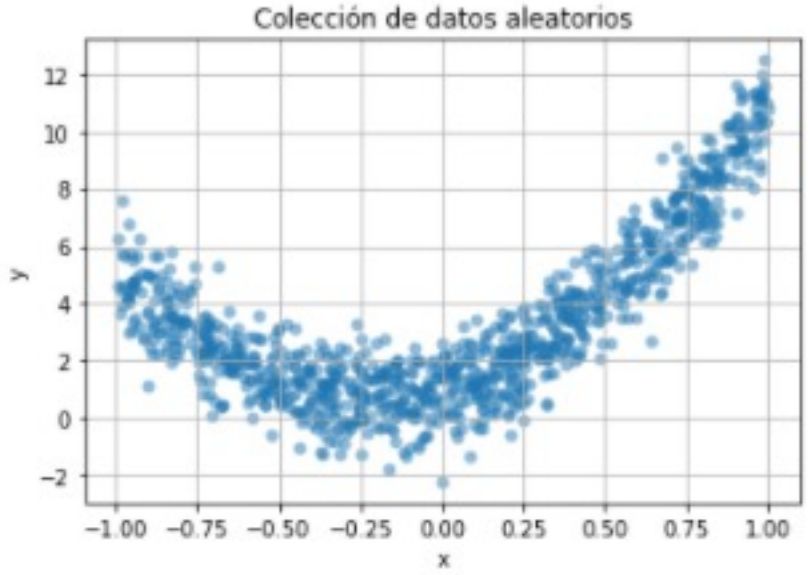
NOTA: Pueden trabajar en sus Notebooks de la Actividad 5 o 6 y hacer el ajuste de regresión lineal allá donde tienen todo cargado en la memoria y luego copiar unas celdas de texto de esa Notebook y agregarlas al final de su Notebook de la Actividad 7. Así no tienen que cargar a la memoria todas las operaciones.

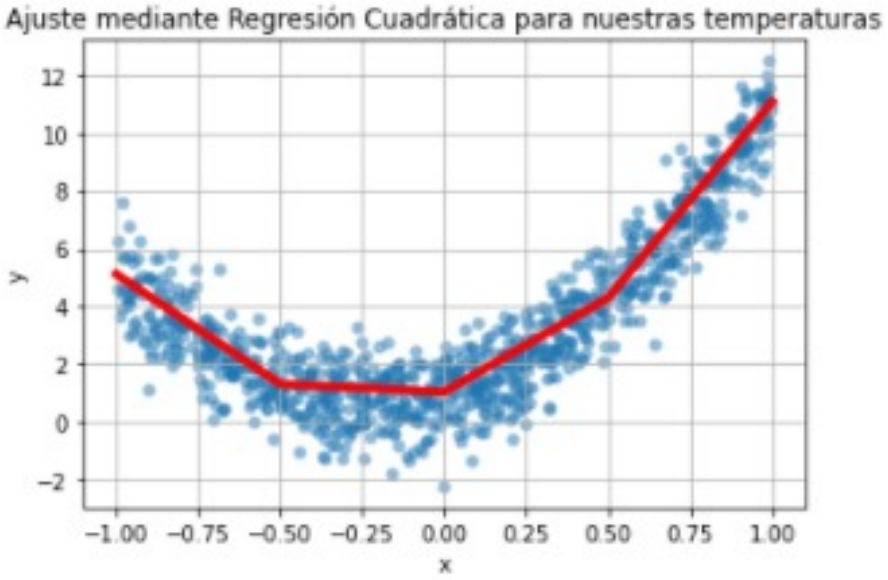
```
# Vamos a hacer una regresión lineal con nuestros datos de temperatura
# Coeficientes de la recta
a0 = 5
a1 = 10
# Número de puntos entre (10,40)
N = 100
x = np.random.rand(100)
# Ruido gaussiano
noise = 0.2*np.random.randn(100)
# Agregamos el ruido a la recta
y = a0 + a1*x + noise

plt.scatter(x,y);
plt.grid()
plt.title('Datos de temperatura')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Obtuvimos las siguientes gráficas:







Concluimos las actividades de esta semana.

4 Conclusión

Como conclusión, podemos ver que es fundamental el álgebra lineal en la física computacional, ya que con esta podemos modelar muchas situaciones de la vida real y poder jugar con los datos de formas en las que podamos usar los datos y sus predicciones a nuestro favor. Es por eso que haber realizado esta actividad me gusto mucho por que se relaciona con lo que me quiero de dedicar terminando la universidad, a al ciencia de datos, a lo que se relaciona todo esto de la estadística, el álgebra lineal y la física computacional.

5 Apéndice

1. **¿Qué te pareció el tema de Análisis de Series de Datos?**
   Me pareció muy bien la actividad, ya que como en un futuro me gustaría enfocarme en la ciencia de datos, esta actividad se relaciona mucho con ello, es por eso le pongo mucha atención y hago todo lo posible por aprender acerca de estos temas.

2. **¿Cómo estuvo el reto?**
   Estuvo algo difícil, ya que no pude compilar en ciertas ocasiones y fue muy difícil encontrar el error, pero al final todo salió bien y me gusto la actividad. Si tuviera que agregar algo, es que no tenía un concocimiento sólido en el álgebra lineal, ya que considero que el docente que me impartio la materia no lo hizo de la mejor formar posible, por ser principios de la pandemia.

3. **¿ ¿Qué se te dificultó más??**
   El repasar los temas de álgebra lineal y encontrar algunas cosas que no tenía tan sólida mi formación.

4. **¿Qué te aburrió?**
   Nada en concreto, todo creo que fue de suma importancia y que en todo momento todo fue muy activo y dinámico.

5. **¿Qué recomendarías para mejorar la primera Actividad??**
   Nada en especial, ya que me gusto mucho como esta implementada esta actividad, pero si vieran un poco más a fondo la ciencia de datos, sería mejor para mí.

6. **¿Que grado de complejidad le asignarías a esta Actividad? (Bajo, Intermedio, Avanzado)**
   Le asignaría un mediano, porque todavía no consideró que la actividad sea demasiado difícil como para que se vuelva intensivo el ejercicio, pero no es lo más fácil del mundo ya que debemos de mover los datos y ajustarlos a nuestro modelo, y prácticamente llevar el álgebra lineal al medio computacional.

6 Bibliografía

(a) *Linear Algebra with SciPy - Mathematical Python. (s. f.). Mathematical Python. Recuperado 7 de marzo de 2021, de https://www.math.ubc.ca/%7Epwalls/math-python/linear-algebra/linear-algebra-scipy/.*

(b) *Solving Linear Systems - Mathematical Python. (s. f.). Mathematical Python. Recuperado 7 de marzo de 2021, de https://www.math.ubc.ca/%7Epwalls/math-python/linear-algebra/solving-linear-systems/.*

(c) *Eigenvalues and Eigenvectors - Mathematical Python. (s. f.). Mathematical Python. Recuperado 7 de marzo de 2021, de https://www.math.ubc.ca/%7Epwalls–/math-python/linearalgebra/eigenvalues-eigenvectors/.*

(d) *Applications - Mathematical Python. (s. f.). Mathematical Python. Recuperado 7 de marzo de 2021, de https://www.math.ubc.ca/%7Epwalls/math-python/linear-algebra/applications/.*

(e) *Briega, L. R. E. (2015, 14 junio). Algebra Lineal con Python. Matemáticas, análisis de datos y python. https://relopezbriega.github.io/blog/2015/06/14/algebra-lineal-con-python/*