Elementos de Cálculo Numérico / Cálculo Numérico

Primer Cuatrimestre 2020

Séptimo ejercicio computacional
Lunes 29/06/20 al Lunes 06/07/20
Recuerde subir el archivo en formato ejercicioX_NOMBREAPELLIDO.py
Recuerde enviar su código al hacer consultas

En este ejercicio aplicaremos el método de cuadrados mínimos para realizar un ajuste de una curva a un cojunto de punto con valores $\{(x_i, y_i)\}_{i=1}^N$. Recordemos que el problema de cuadrados mínimos consiste en encontrar un vector de coeficientes c tal que

$$Ac = Y \tag{1}$$

donde la matriz A está construida en base a una presunta relación entre los valores de x y de y, e Y es tal que $Y_i = y_i$. Para poder aplicar el método de cuadrados mínimos necesitamos que la relación entre los coeficientes c_i y los valores de y_i sea lineal, aunque la dependencia con los valores de x_i no lo sea. Podemos escribir entonces que

$$A_{ij} = \phi_j(x_i) \tag{2}$$

de forma que $y_i = \sum_{j=1}^M c_j \phi_j(x_i) = \sum_{j=1}^M A_{ij} c_j$. Para encontrar los valores de c_j, buscaremos resolver la ecuación

$$A^T A c = A^T Y \tag{3}$$

que nos dará los valores de c tales que $||Ac - y||$ sea mínimo. Llamamos a la matriz $A^T A = B$.

A Construya un par de vectores X e Y tal que X conste de 10 valores equiespaciados entre 0 y π, y $Y = \sin(X)$

B Construya una función que reciba un vector con valores `X`, un número entero `n` y un argumento `tipo` y construya una matriz `A` tal que:

  1 Si `tipo` es `polinomial` entonces `A[i][j] = X[i]**j`, donde `j` se mueve entre 0 y `n`.

  2 Si `tipo` es `senoidal` entonces `A[i][j] = sin((j+1)*X[i])`, donde `j` se mueve entre 0 y `n`.

  Puede usar el siguiente modelo:

```
def matriz_A(x,n,tipo):
    nx = # COMPLETAR #
    A = np.zeros((nx,n))
    if tipo=='polinomial':
```

```
        for i in range(nx):
            for j in range(n):
                A[i][j] = ## COMPLETAR ##
    elif tipo=='senoidal':
        for i in range(nx):
            for j in range(n):
                A[i][j] = ## COMPLETAR ##
    return(A)
```

Checkee esta función, usando por ejemplo, `n` $= 3$, `x = np.array([0,2])` para ambos valores de `tipo`.

C Construya una función que reciba los vectores `x` e `y`, un número de coeficientes `n` y un argmento `tipo` como en el punto A, que construya la matriz y resuelva el problema de cuadrados mínimos. Puede usar el modelo:

```
def cuadrados(x,y,n,tipo):
    A = matriz_A(x,n,tipo)
    B = ## COMPLETAR ##
    c = ## COMPLETAR ##
    return(c)
```

Checkee esta función usando, por ejemplo `x=np.array([0,1,2]), y = np.array([0,1,4])`, `n=2` y `tipo='polinomial'`. El resultado debería ser muy próximo a $c = (0, 0, 1)$. *¿Por qué?*

Considere usar las funciones de `numpy` y `numpy.linalg`: `np.dot`, `npl.inv`, `np.transpose`.

D Considerando `n=3`, encuentre los coeficientes correspondientes a ambos tipos posibles para los valores de `X` e `Y` del punto A. Interpretelos, ¿tienen sentido para usted?

E Construya una funcion que reciba un argumento `tipo` y un vector de coeficientes `c`, y retorne una función que calcule el valor de la aproximación por cuadrados mínimos para el dado tipo. Esta función deberá recibir como argumento un array `z` de valores y devolver un array `w` con los valores de ajuste en cada valor de `z`. Puede usar el siguiente modelo:

```
def genera_ajustador(c,tipo):
    if tipo=='polinomial':
        def function(z):
            w = 0
            for j in range(len(c)):
                w += ## COMPLETAR ##
            return(## COMPLETAR ##)
    elif tipo=='senoidal':
        def function(z):
            w = 0
```

```
            for j in range(len(c)):
                w += ## COMPLETAR
            return(## COMPLETAR ##)
    return(## COMPLETAR ##)
```

Emplee esta función para graficar el resultado de los ajustes realizados en el paso previo, tomando 50 puntos equiespaciados en el rango de `X`, graficando como puntos los valores de `X` e `Y`, y como lineas el resultado de ambos ajustes.

F Empleando los datos del problema 7, realize un ajuste considerando a `X` como el logaritmo del área de un tract y a `Y` como el logaritmo de la población, empleando `tipo='polinomial'` y `n` iguales a 2,4,6 y 8 (grados impares). ¿Cuál de estos ajustes representa mejor los datos?

G **Extra 1:** Construya una función que reciba la función ajustadora `ajustador`, y dos vectores `X` e `Y` y calcule el error de ajuste como la norma 2 entre el valor generado por la función ajustadora y el valor original. Puede usar el siguiente modelo:

```
def calcula_error(X,Y,ajustador):
    Y_ajustador = ## COMPLETAR ##
    error = ## COMPLETAR ##
    return(error)
```

Calcule el error de ajustar los valores de `X`, e `Y` propuestos en el paso F para cada `n` considerado y grafique el error en función de n. ¿Cuál ajuste tiene menor error? ¿Se condice con lo que visualmente observó en el punto anterior?

H **Extra 2** *Si aun le quedan ganas de seguir* Construya ahora una función que reciba los vectores `X`, `Y`, el argumento `tipo` y el número de coeficientes `n` y calcule, para cada `i` en `range(len(X))`

1 La funcion ajustadora considerando todos los datos excepto el `i`-esimo.

2 El error de la función ajustadora en el valor `i`-esimo que no considero para determinar los coeficientes

Teniendo el error de predicción para cada valor, la función debe promediar estos valores y retornar ese error promediado. Este error nos permite evaluar la calidad del modelo considerado. Puede usar el siguiente modelo:

```
def error_calidad_modelo(X,Y,n,tipo):
    error_valores = []
    for i in range(## COMPLETAR ##):
        X_noi = ## COMPLETAR ##
        Y_noi = ## COMPLETAR ##
        c = cuadrados(X_noi,Y_noi,n-1,tipo)
        ajustador = genera_ajustador(c,tipo)
        error = calcula_error(X[#COMPLETAR#],Y[#COMPLETAR#],ajustador)
```

```
        error_valores.append(error)
    error_medio = np.mean(#COMPLETAR#)
    return(error_medio)
```

¿Qué modelo tiene menor error bajo este cálculo? Grafique este nuevo error en función de n, y grafique un error en función del otro.

El error del punto F representa cuan bien ajusta el modelo a los datos, y decrece a medida que agregamos más y más coeficientes (aumentamos `n`), acercandonos al polinomio interpolador (que tendría error cero). En cambio, el error en el punto G representa cuan bien nos permite predecir nuevos valores el modelo que consideramos, a partir de los datos a los que tenemos acceso. En este segundo error tendremos que el polinomio interpolador será muy malo, ya que ajusta perfectamente los datos que consideramos, sin capacidad de extrapolar a nuevos datos.