Algoritmo Envolvente Convexo

October 7, 2024

Universidad Autonnoma del Estado de Mexico

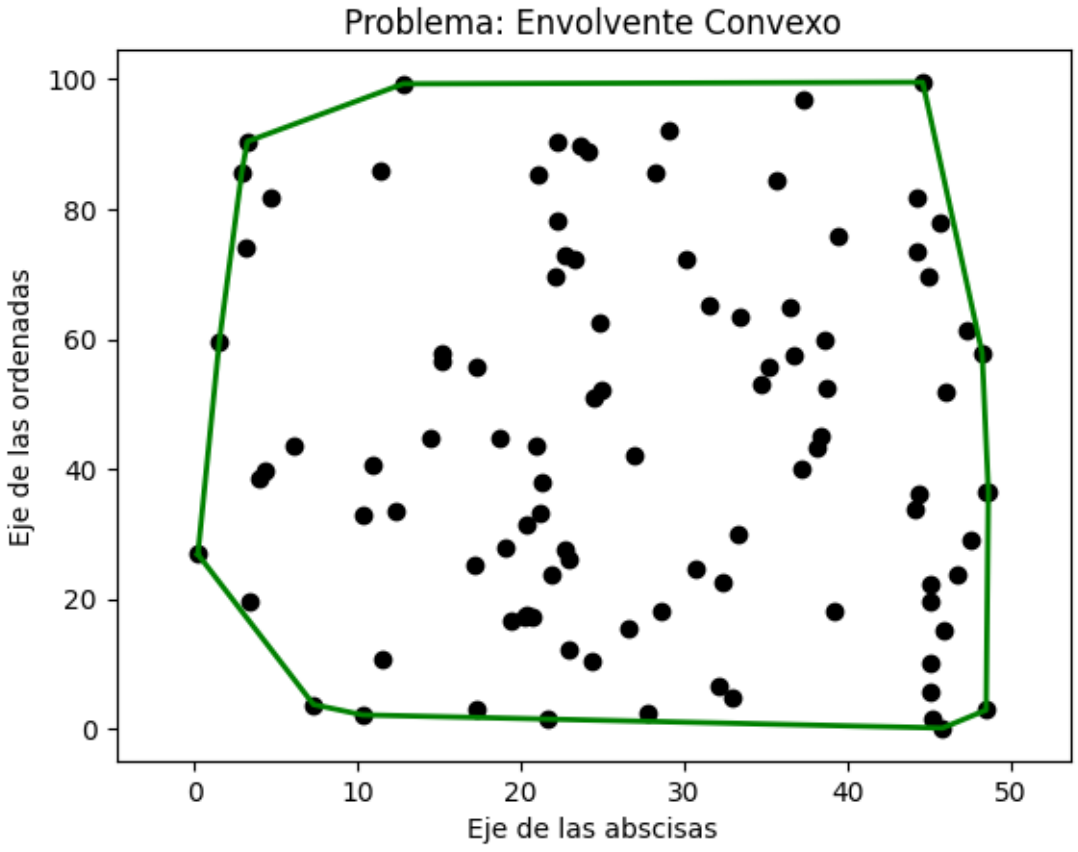
Raul Alejandro Calderon Hernandez

Envolvente Convexo

07 de Octubre de el 2024

[3]:
```
import random as rand
import numpy as np
import matplotlib.pyplot as plt

def turn_right():
    array = [coord_points[0], coord_points[1]]
    for i in range(2, len(coord_points)):
        array.append(coord_points[i])
        while len(array) > 2 and np.linalg.det([array[-3], array[-2],␣
 ↪array[-1]]) > 0:
            array.pop(-2)
    return array

def convex_hull():
    coord_points.sort()
    l_upper = turn_right()
    coord_points.reverse()
    l_lower = turn_right()
    l = l_upper + l_lower
    return l

def graph(convex_pol, coord_points):
    x_points = [i[0] for i in coord_points]
    y_points = [i[1] for i in coord_points]
    x_polygon = [i[0] for i in convex_pol]
    y_polygon = [i[1] for i in convex_pol]
    x_lim_der = max(x_points) + 5
    y_lim_sup = max(y_points) + 5
    x_lim_izq = min(x_points) - 5
    y_lim_inf = min(y_points) - 5
    # Asignación de los límites extremos
```

```
    plt.xlim(x_lim_izq, x_lim_der)
    plt.ylim(y_lim_inf, y_lim_sup)
    # Graficación
    plt.title('Problema: Envolvente Convexo')
    plt.xlabel('Eje de las abscisas')
    plt.ylabel('Eje de las ordenadas')
    plt.plot(x_points, y_points, 'ko')
    plt.plot(x_polygon, y_polygon, 'g-', linewidth = 2.0)
    plt.show()

num_points = 100
coord_points = []
for i in range(num_points): coord_points.append([rand.uniform(0, 50), rand.
 ↪uniform(0, 100), 1.0])
convex_pol = convex_hull()
graph(convex_pol, coord_points)
```



[4]:
```
import random as rand
import numpy as np
```

```
import matplotlib.pyplot as plt
```

importamos las 3 librerias necesarias para el codigo, como son random para poder tener datos aleatorios, numpy y matplotlib que nos ayudaran a graficar los datos que anteriormente obtuvimos

[8]:
```
def turn_right():
    array = [coord_points[0], coord_points[1]]
    for i in range(2, len(coord_points)):
        array.append(coord_points[i])
        while len(array) > 2 and np.linalg.det([array[-3], array[-2],␣
 ↪array[-1]]) > 0:
            array.pop(-2)
    return array
```

cuyo propósito parece ser determinar y retornar un subconjunto de puntos de coordenadas (coord_points) mientras mantiene alguna condición geométrica sobre las orientaciones de los puntos, utilizando el determinante para evaluar si se hace un "giro a la derecha". Este bucle while se ejecuta si la lista array tiene al menos tres puntos. np.linalg.det: Calcula el determinante de la matriz formada por los tres últimos puntos de array.

[9]:
```
def convex_hull():
    coord_points.sort()
    l_upper = turn_right()
    coord_points.reverse()
    l_lower = turn_right()
    l = l_upper + l_lower
    return l
```

se procesan los puntos en orden ascendente y descendente para construir las envolventes superior e inferior del conjunto de puntos. Con ayuda de las funciones de sort y reverse nos ayudan a obtener un mejor puntaje, y en si, una mejor disposicion de los puntos

[10]:
```
def graph(convex_pol, coord_points):
    x_points = [i[0] for i in coord_points]
    y_points = [i[1] for i in coord_points]
    x_polygon = [i[0] for i in convex_pol]
    y_polygon = [i[1] for i in convex_pol]

    x_lim_der = max(x_points) + 5
    y_lim_sup = max(y_points) + 5
    x_lim_izq = min(x_points) - 5
    y_lim_inf = min(y_points) - 5

    plt.xlim(x_lim_izq, x_lim_der)
    plt.ylim(y_lim_inf, y_lim_sup)

    plt.title('Problema: Envolvente Convexo')
    plt.xlabel('Eje de las abscisas')
```

```
    plt.ylabel('Eje de las ordenadas')
    plt.plot(x_points, y_points, 'ko')
    plt.plot(x_polygon, y_polygon, 'g-', linewidth=2.0)
    plt.show()
```

que se utiliza para graficar los puntos y el envolvente convexo generado por la función convex_hull()

Se crean dos listas, x_points y y_points, que contienen las coordenadas x e y de los puntos originales en coord_points, respectivamente.

De la misma manera, se extraen las coordenadas x e y de los puntos que forman el envolvente convexo en convex_pol.

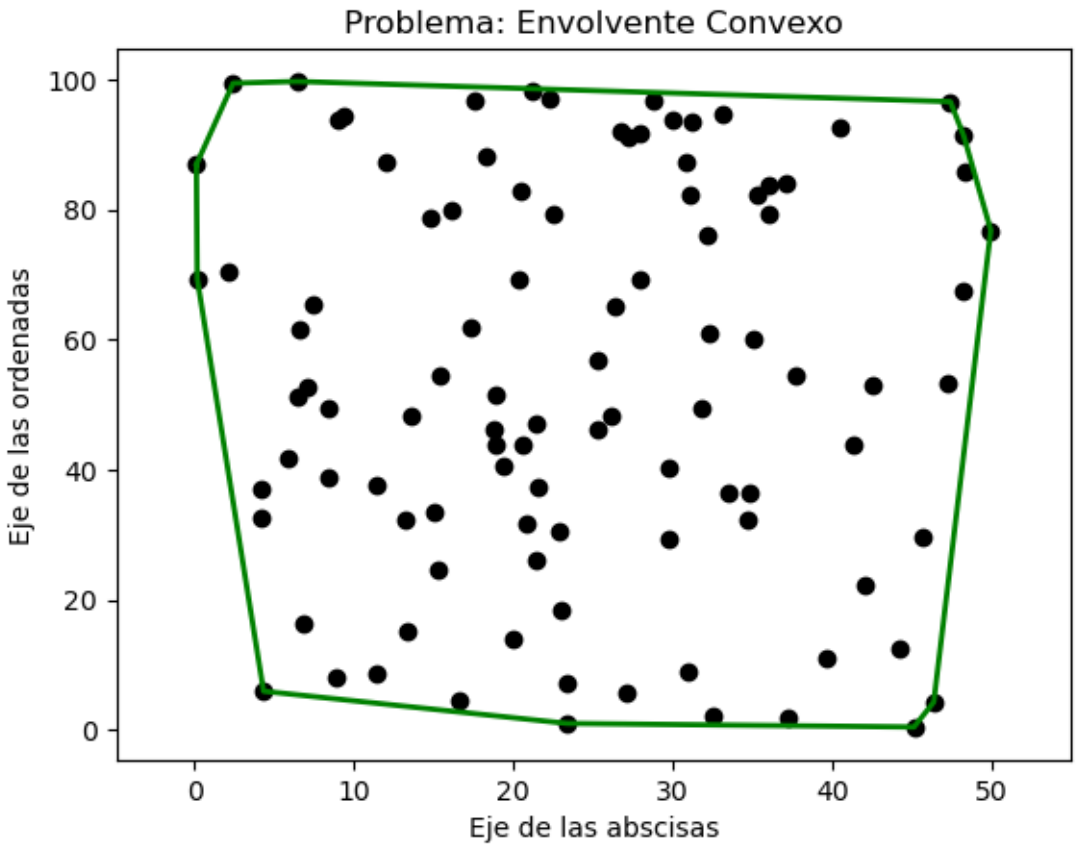
Se establecen los límites del gráfico añadiendo un pequeño margen de 5 unidades a los valores máximo y mínimo de las coordenadas x e y.

'ko': Se utiliza un formato que indica que los puntos deben ser dibujados con color negro ('k') y en forma de círculo ('o').

'g-': Indica que el polígono debe ser dibujado con una línea continua verde ('g').

configura el gráfico, establece los límites, etiqueta los ejes y muestra los puntos originales y el envolvente convexo en el gráfico.

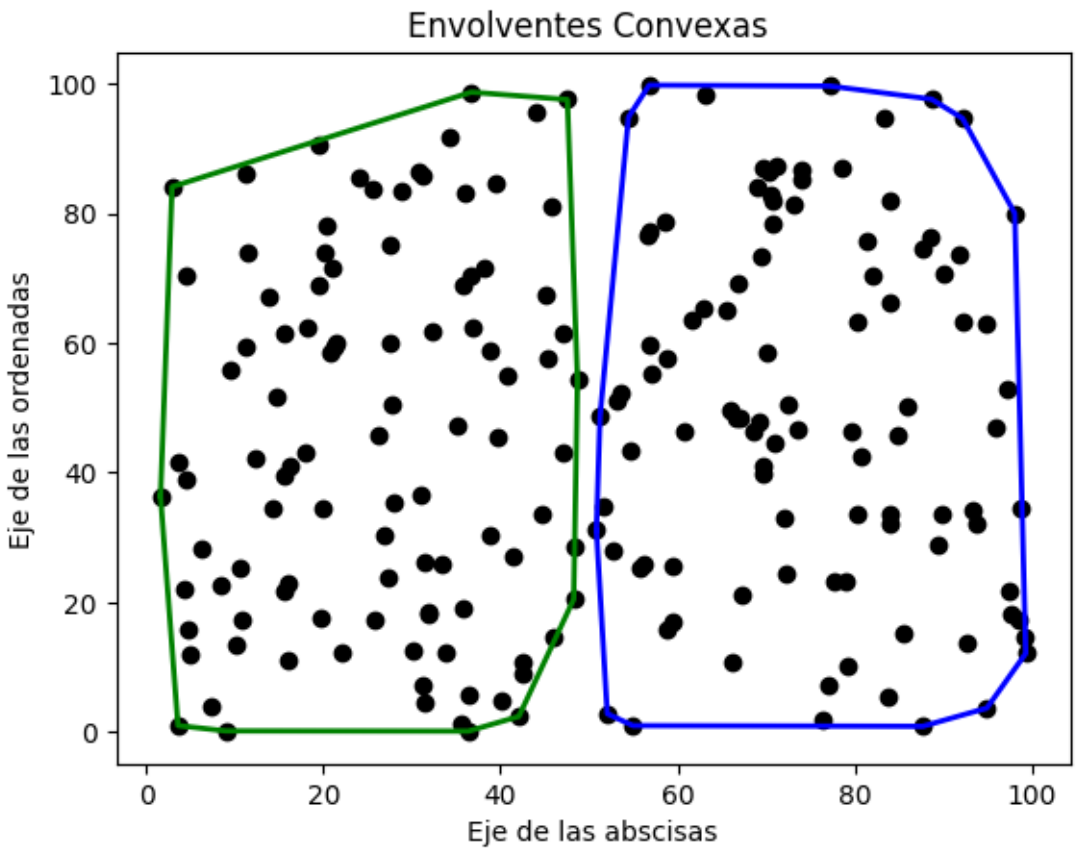
[11]:
```
num_points = 100
coord_points = []
for i in range(num_points): coord_points.append([rand.uniform(0, 50), rand.
 ↪uniform(0, 100), 1.0])

convex_pol = convex_hull()
graph(convex_pol, coord_points)
```



genera aleatoriamente 100 puntos, donde saca su envolvente convexo, esto quiere decir que hay una figura que podemos trazar, dentro de los puntos generados anteriormente, por medio de un for vamos generando aleatoriamente puntos y despues los empezaremos a graficar con ayuda de los metodos vistos anteriormente

[29]:
```
import random as rand
import numpy as np
import matplotlib.pyplot as plt
def turn_right(coord_points):
    array = [coord_points[0], coord_points[1]]
    for i in range(2, len(coord_points)):
        array.append(coord_points[i])
        while len(array) > 2 and np.linalg.det([array[-3], array[-2],␣
 ↪array[-1]]) > 0:
            array.pop(-2)
    return array

def convex_hull(coord_points):
    coord_points.sort()
    l_upper = turn_right(coord_points)
```

```
    coord_points.reverse()
    l_lower = turn_right(coord_points)
    l = l_upper + l_lower
    return l

def graph(convex_pol1, coord_points1, convex_pol2, coord_points2):
    x_points1 = [i[0] for i in coord_points1]
    y_points1 = [i[1] for i in coord_points1]
    x_polygon1 = [i[0] for i in convex_pol1]
    y_polygon1 = [i[1] for i in convex_pol1]
    x_points2 = [i[0] for i in coord_points2]
    y_points2 = [i[1] for i in coord_points2]
    x_polygon2 = [i[0] for i in convex_pol2]
    y_polygon2 = [i[1] for i in convex_pol2]
    x_lim_der = max(max(x_points1), max(x_points2)) + 5
    y_lim_sup = max(max(y_points1), max(y_points2)) + 5
    x_lim_izq = min(min(x_points1), min(x_points2)) - 5
    y_lim_inf = min(min(y_points1), min(y_points2)) - 5
    plt.xlim(x_lim_izq, x_lim_der)
    plt.ylim(y_lim_inf, y_lim_sup)
    plt.title('Envolventes Convexas ')
    plt.xlabel('Eje de las abscisas')
    plt.ylabel('Eje de las ordenadas')
    plt.plot(x_points1, y_points1, 'ko')
    plt.plot(x_polygon1, y_polygon1, 'g-', linewidth=2.0)
    plt.plot(x_points2, y_points2, 'ko')
    plt.plot(x_polygon2, y_polygon2, 'b-', linewidth=2.0)
    plt.show()

num_points = 100
coord_points = []
coord_points2 = []
for i in range(num_points): coord_points.append([rand.uniform(0, 50), rand.
 ↪uniform(0, 100), 1.0])
for i in range(num_points): coord_points2.append([rand.uniform(50, 100), rand.
 ↪uniform(0, 100), 1.0])

# Creación de los polígonos convexos para cada conjunto de puntos
convex_pol1 = convex_hull(coord_points1)
convex_pol2 = convex_hull(coord_points2)

# Graficación de ambos polígonos
graph(convex_pol1, coord_points1, convex_pol2, coord_points2)
```



Cambiamos el codigo para que se pueda visualizar las dos figuras dentro de un solo plano, donde cada figura contiene 50 puntitos, y que contienen un espacio de 50 px, donde ahora lo importante no es que tuviera dos cosas para un solo metodo sino que a la hora de graficar nos dara un resultado al que tenemos en pantalla

Observamos como los puntos de la figura dos, la figura azul se acercan a los de la figura 1, pero esto por como lo programamos, si le damos mucha mas posicion dentro del recuadro hasta se pueden interponer las figuras esto ya es dependiendo de los gustos y formas que deseemos podemos mejorar aun mas el codigo, optimizandolo, con una lista de figuras, y que de esta lista de figuras nos de la figura envolvente convexa, con los puntos mencionados anteriormente, y con sus respectivas posiciones, como un kmeans, pero sin tener tanto algoritmo de inteligencia artificial

Tambien podemos optimizar a la hora de graficar para que este nos devuelva el grafico, sin necesidad de aumentar los parametros o las listas que estos contienen pero asi funciona bastante bien

[ ]: