CS 472 Assignment 3: Convex Hull Brute Force

Codey Sivley

For Dr. Lewis Spring 2022

Feb 28, 2022

Intro:

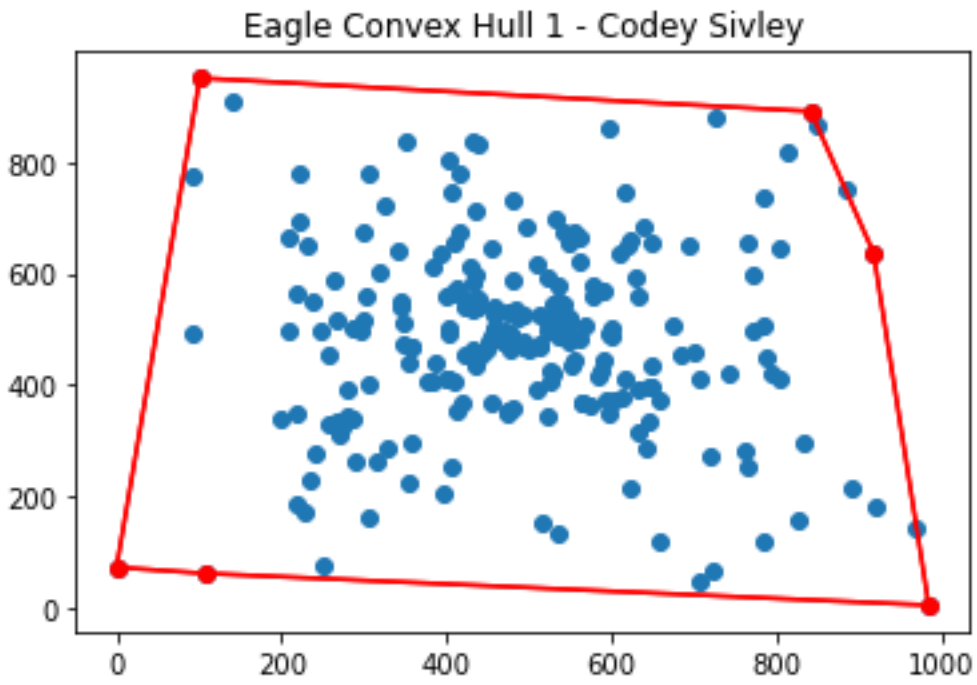
This problem was previously assigned as an exam question in our OOP class, so I had some related code in Python. As such, I pulled up that old code and made modifications to make it work with lists of tuples.

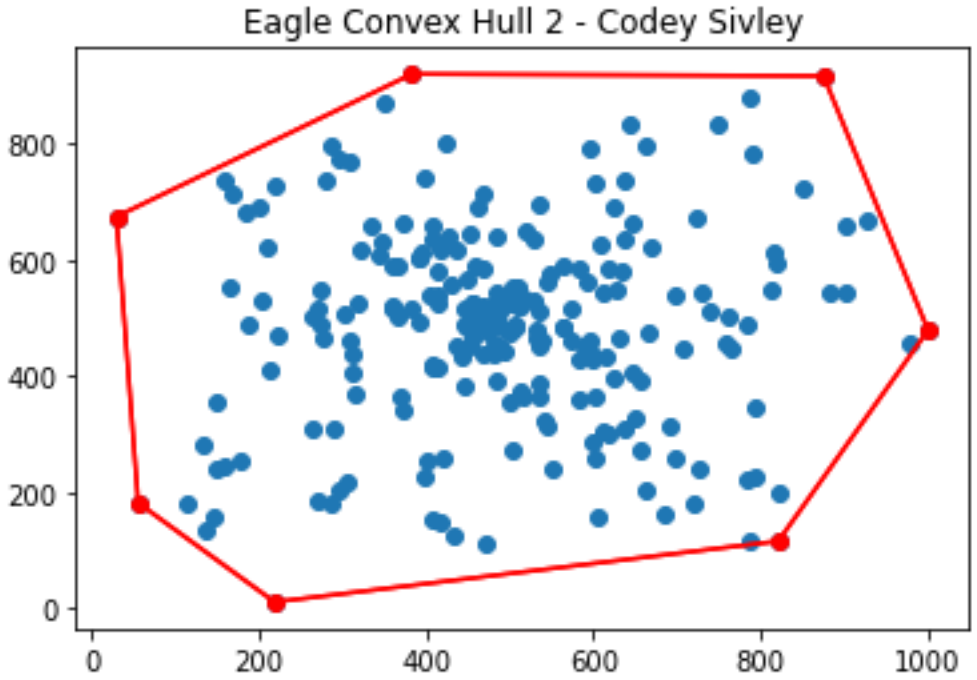
Functions employed are:

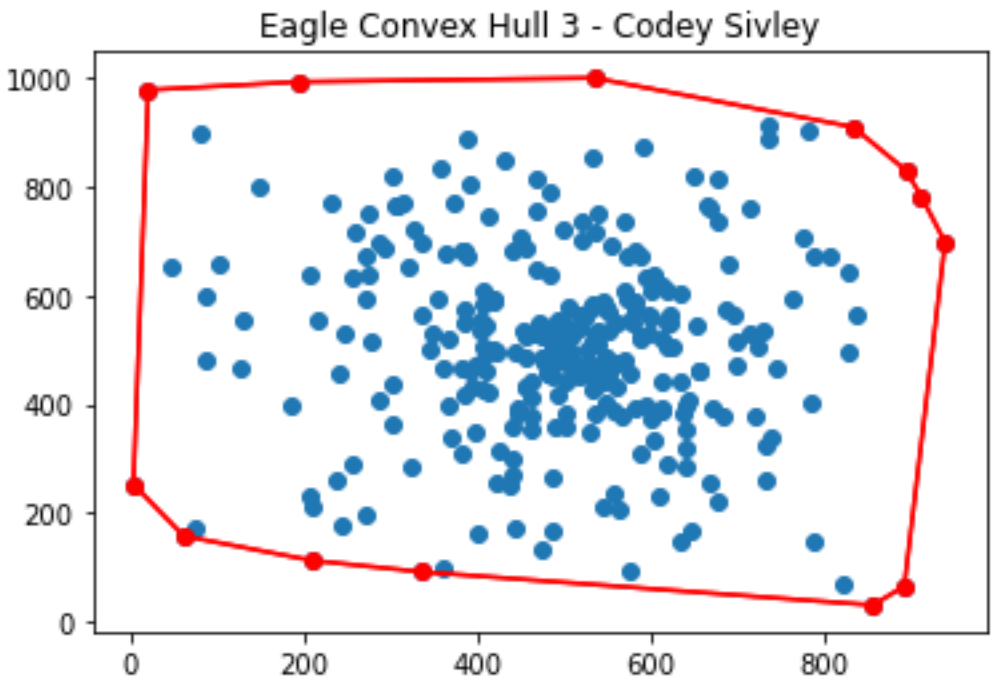
- def convexHullTest(lineStart, lineEnd, testPoint)
    - this function does the basic operation for “General Equation of a Line”, ax+by=c.
- def convexHullRoutine(listOfPoints, hull)
    - given a list of points and an empty set, this function pushes any coordinates it finds to be edge points to the hull set.
    - Iterates over the list of points, calling convexHullTest each time.
        - O(n^3) worst-case
    - Skips iteration if any of the points being checked are identical
    - Earlier versions stored the calculated value of convexHullTest in an array, then checked each list member to see if any were positive, then checked again to see if any were negative.
        - Very wasteful. N^3 datapoints stored, and (n^3)*2 passes to check all points parity.
        - Why bother storing them if we only need to check the parity?
    - New version has a positive flag and a negative flag.
        - Each call to convexHullTest stores in the same memory location. (as opposed to list n^3 long)
        - Simply checks parity and sets corresponding flag to positive.
        - Then, if both positive and negative flags are tripped, breaks out of loop.
        - Saves (n^3)-1 memory locations, and lots of calls to convexHullTest().
- def importPoints(theFile)
    - Simple import of tuples generated by random point generator function
- randomPointsGenCenterWeighted.py
    - Pure random number generation led to unpleasantly rectangular results.
    - Reworked random point generator script to generate points in a number of passes, each pass with an increased offset from the centerpoint.
    - Results in a more pleasant “shotgun” spread pattern.

Results:

In order to check results, added a little matplotlib magic. Following are three plots of runs on center-weighted datasets.


Eagle Convex Hull 1 - Codey Sivley
0
200
400
600
800
0
200
400
600
800
1000


Eagle Convex Hull 2 - Codey Sivley
0
200
400
600
800
0
200
400
600
800
1000



Source Code:

Code is copy-pasted as plain text.

convexHull.py:

```
# -*- coding: utf-8 -*-

"""

Codey Sivley

For Algorithms CS472

Dr. Lewis

Spring 2022


Convex Hull Brute Force

with added Graphing

"""


import matplotlib.pyplot as plt
```

```
def convexHullTest(lineStart, lineEnd, testPoint):
    #input: Three xy points as tuples
    #output: int result of ax+by-c
    x0 = lineStart[0]
    y0 = lineStart[1]
    x1 = lineEnd[0]
    y1 = lineEnd[1]
    x2 = testPoint[0]
    y2 = testPoint[1]

    lineY = y1-y0
    lineX = x0-x1
    c = (x0*y1)-(x1*y0)

    #below, x2 & y2 are the point being tested,
    #lineX and lineY are properties of the two line endpoints
    #if all returns are either >= 0 or <=0, lineStart and lineEnd are
    #on the convex hull.

    return ((lineY * x2) + (lineX * y2) - c)

########################################

def convexHullRoutine(listOfPoints, hull):
    #input: points as list of tuples, empty set
    #output: fills set with points on convexHull

    #flags to break out of loop, saves lots of processes and memory
    positiveFlag = 0
```

```
negativeFlag = 0
breakFlag = 0

for i in listOfPoints:
    for j in listOfPoints:
        if (i == j):
            continue

        for k in listOfPoints:
            if ((k == i) or (k == j)):
                continue
            parity = convexHullTest(i,j,k)
            if parity > 0:
                positiveFlag = 1
            elif parity < 0:
                negativeFlag = 1
            else:
                pass #don't need to worry about zero

            if (positiveFlag and negativeFlag):
                positiveFlag = 0
                negativeFlag = 0
                breakFlag = 1
                break
        if breakFlag:
            breakFlag = 0
            continue
        #if we made it here, we have an edge.
        hull.add(i)
```

```
            hull.add(j)
            print("Found Edge: " + str(i) + ", " + str(j))

            #plot stuff
            x, y = zip(i,j)
            plt.plot(x,y, c='red')

            #reset for next pass
            positiveFlag = 0
            negativeFlag = 0
    print("Done, Found " + str(len(hull)) + " edge points.")

##########################

def importPoints(theFile):
    pointList = []
    rawData = open(theFile, "r")
    for x in rawData:
        pointList.append(eval(x))
    rawData.close()
    return pointList

##########################

#main starts here
hull = set() #hull is set to prevent duplicates
eaglePoints = importPoints("eaglePoints3.txt")

plt.scatter(*zip(*eaglePoints)) #plot all points as blue
```

```
convexHullRoutine(eaglePoints, hull) # find convex hull


plt.scatter(*zip(*hull), c='red') #color edge points red
plt.title("Eagle Convex Hull 3 - Codey Sivley")
plt.show()


#output results
with open("eagleHullResults3.txt", 'a') as results:
    for each in hull:
        results.write(str(each) + "\n")
```

randomPointsGenCenterWeighted.py:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 28 13:58:45 2022

@author: csivley
"""


#random points generator


import random


bounds = 1000
origin = int(bounds/2)
levels = 10
offset = int(origin/levels)
saturation = 0.0003
```

```
ppl = int((bounds*bounds*saturation)/levels)
pointsWritten = 0

file = open('eaglePoints3.txt', "a") #not using with open because bigO

for each in range (levels):
    for i in range (ppl):
        currentRange = offset * (each + 1)
        randX = origin + (random.randint(-currentRange,currentRange))
        randY = origin + (random.randint(-currentRange,currentRange))
        dataPoint = (randX, randY)
        file.write(str(dataPoint) + "\n")
        pointsWritten = pointsWritten + 1

file.close()
print(str(pointsWritten) + " points written.")
```