LAPORAN TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA

Implementasi Convex Hull untuk Visualisasi Tes Linear Separability Dataset dengan Algoritma Divide and Conquer

NIM	: 13520085
Nama	: Ubaidillah Ariq Prathama
Kelas	: K01

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

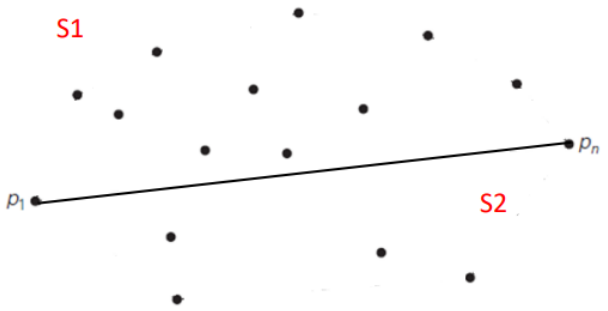
2021

A. Algoritma Divide and Conquer pada Convex Hull

Convex hull merupakan persoalan klasik dalam geometri komputasional. Persoalan ini digambarkan secara sederhana dalam ruang dimensi dua (bidang) sebagai mencari subset dari himpunan titik pada bidang tersebut sedemikian rupa sehingga jika titik-titik tersebut dijadikan poligon maka akan membentuk poligon yang konveks. Suatu poligon dikatakan konveks jika digambarkan garis yang menghubungkan antar titik maka tidak ada garis yang memotong garis yang menjadi batas luar poligon. Definisi lain dari convex hull adalah poligon yang disusun dari subset titik sedemikian sehingga tidak ada titik dari himpunan awal yang berada di luar poligon tersebut (semua titik berada di batas luar atau di dalam area yang dilingkupi oleh poligon tersebut).

Terdapat beberapa algoritma untuk mengerjakan convex hull. Salah satunya adalah algoritma divide and conquer. Adapun Langkah-langkah dari algoritma ini adalah :

1. Urutkan titik secara membesar berdasarkan sumbu-x, misal titik paling kiri adalah p_1 dan titik paling kanan adalah p_n.
2. Garis yang menghubungkan p_1 dan p_n (p_1p_n) membagi S menjadi dua bagian yaitu S1 (kumpulan titik di sebelah kiri atau atas garis p_1p_n) dan S2 (kumpulan titik di sebelah kanan atau bawah garis p_1p_n).

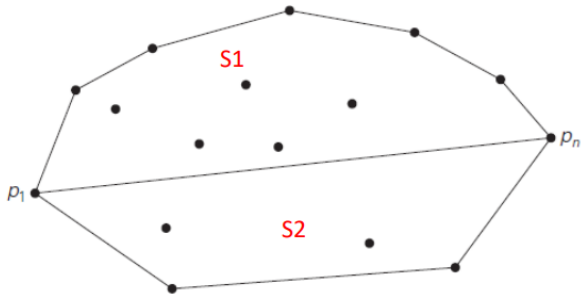


Untuk memeriksa apakah sebuah titik berada di sebelah kiri (atau atas) suatu garis yang dibentuk dua titik, gunakan penentuan determinan :

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3$$

3. Semua titik pada S yang berada pada garis p_1p_n (selain titik p_1 dan p_n) tidak mungkin membentuk convex hull, sehingga bisa diabaikan dari pemeriksaan.

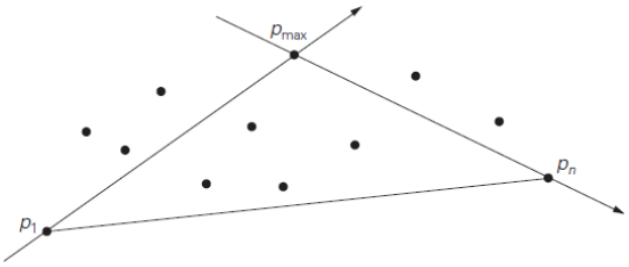
4. Kumpulan titik pada S1 bisa membentuk convex hull bagian atas, dan kumpulan titik pada S2 bisa membentuk convex hull bagian bawah.



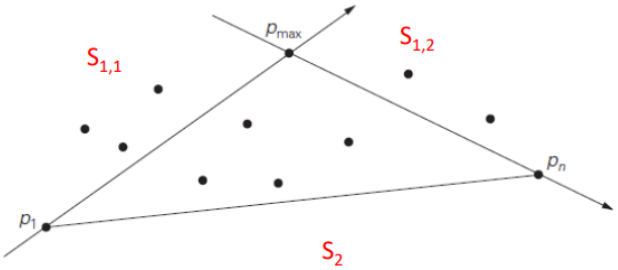
5. Untuk sebuah bagian (misal S1), terdapat dua kemungkinan :

   • Jika tidak ada titik lain selain S1, maka titik p_1 dan p_n menjadi pembentuk convex hull bagian S1

   • Jika S1 tidak kosong, pilih sebuah titik yang memiliki jarak terjauh dari garis $p_1 p_n$ (misal p_{max}). Jika terdapat beberapa titik dengan jarak yang sama, pilih sebuah titik yang memaksimalkan sudut $p_{max}\ p_1\ p_n$ Semua titik yang berada di dalam daerah segitiga $p_{max}\ p_1\ p_n$ diabaikan untuk pemeriksaan lebih lanjut



6. Tentukan kumpulan titik yang berada di sebelah kiri garis $p_1 p_{max}$ (menjadi bagian S1,1), dan di sebelah kanan garis $p_1 p_{max}$ (menjadi bagian S1,2)



7. Lakukan hal yang sama (butir 5 dan 6) untuk bagian S2, hingga bagian ‘kiri’ dan ‘kanan’ kosong.
8. Kembalikan pasangan titik yang dihasilkan.

Algoritma ini memiliki kompleksitas $O(N \log N)$ dimana N merupakan banyaknya titik. Untuk mengurutkan semua titik dibutuhkan kompleksitas $O(N \log N)$ dengan asumsi sorting dapat dilakukan dengan efektif. Untuk memeriksa apakah titik berada di sisi mana diperlukan $O(N)$ dengan N merupakan banyaknya titik yang dicek. Untuk mencari titik yang memiliki jarak terjauh juga diperlukan $O(N)$ dengan N merupakan banyaknya titik yang dicek. Penggabungan hanya dilakukan dengan memasukkan titik ke dalam sebuah array atau set sehingga dapat dilakukan dalam $O(1)$.

B. Code Dalam Bahasa Python

1. Code untuk library convexHull.py

```
import math
import matplotlib.pyplot as plt

#Algorithm
myHull = []
hullPoints = set()

def section(start, end, test) :
    det = start[0]*end[1] + test[0]*start[1] + end[0]*test[1] -
test[0]*end[1] - end[0]*start[1] - start[0]*test[1]
    if det > 1e-9 :
        return 1
    elif det < -1e-9 :
        return -1
    else :
        return 0

def distance(start, end, test) :
    if end[0] == start[0] :
        return abs(test[0] - end[0])
    else :
        a = (end[1]-start[1])/(end[0]-start[0])
        b = -1
        c = start[1] - start[0]*a
        return abs(a*test[0] + b*test[1] + c)/math.sqrt(a*a + b*b)
```

```
def divide(oldArr, start, end, orientation) :
    newArr = []
    for elmt in (oldArr) :
        if section(start, end, elmt) == orientation :
            newArr.append(elmt)
    return newArr

def myConvexHull(oldArr, start, end, orientation) :
    newArr = divide(oldArr, start, end, orientation)
    maxDist = -1
    maxDistPos = []

    if len(newArr) == 0 :
        myHull.append([start[0],start[1],end[0],end[1]])
        hullPoints.add(tuple(start))
        hullPoints.add(tuple(end))
        return

    for elmt in (newArr) :
        dist = distance(start, end, elmt)
        if maxDist < dist :
            maxDist = dist
            maxDistPos = elmt

    myConvexHull(newArr, start, maxDistPos, -section(start, maxDistPos, end))
    myConvexHull(newArr, maxDistPos, end, -section(maxDistPos, end, start))
    return


#Show result
def displayConvexHull(df, data, x, y) :
    plt.figure(figsize = (10, 6))
    colors = ['b','r','g']
    plt.title('Convex Hull')
    plt.xlabel(data.feature_names[x-1])
    plt.ylabel(data.feature_names[y-1])
    for i in range(len(data.target_names)):
        myHull.clear()
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:,[x-1,y-1]].values
        bucket = sorted(bucket, key=lambda x:x[0])
        start = bucket[0]
        end = bucket[len(bucket) - 1]
        myConvexHull(bucket, start, end, 1)
        myConvexHull(bucket, start, end, -1)
```

```
        print("Set of hull points : ")
        print(hullPoints)
        for j in range (len(bucket)) :
            plt.scatter(bucket[j][0], bucket[j][1],
label=data.target_names[i], color = colors[i])
        for j in range (len(myHull)) :
            plt.plot([myHull[j][0],myHull[j][2]], [myHull[j][1],
myHull[j][3]], colors[i])
    plt.show()
```

2. Code untuk main.py

```
import convexHull as ch
import pandas as pd
from sklearn import datasets

print("Available dataset")
print("1. Iris Dataset")
print("2. Wine Dataset")
print("3. Breast Cancer Dataset")
inputDataset = int(input("Input dataset number : "))

if inputDataset == 1 :
    data = datasets.load_iris()
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
    print("1. Sepal Length")
    print("2. Sepal Width")
    print("3. Petal Length")
    print("4. Petal Width")
    inputX = int(input("Input x coordinate number : "))
    inputY = int(input("Input y coordinate number : "))
    ch.displayConvexHull(df, data, inputX, inputY)

elif inputDataset == 2 :
    data = datasets.load_wine()
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
    print("1. Alcohol")
    print("2. Malic acid")
    print("3. Ash")
    print("4. Alcalinity of ash")
    print("5. Magnesium")
    print("6. Total phenols")
```

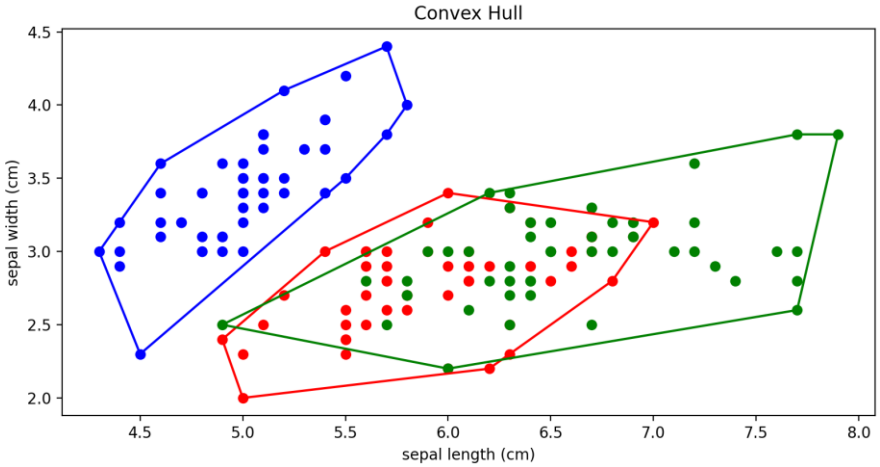
```
    print("7. Flavanoids")
    print("8. Nonflavanoid phenols")
    print("9. Proanthocyanins")
    print("10. Color intensity")
    print("11. Hue")
    print("12. OD280/OD315 of diluted wines")
    print("13. Proline")
    inputX = int(input("Input x coordinate number : "))
    inputY = int(input("Input y coordinate number : "))
    ch.displayConvexHull(df, data, inputX, inputY)

elif inputDataset == 3 :
    data = datasets.load_breast_cancer()
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
    print("1. Radius")
    print("2. Texture")
    print("3. Perimeter")
    print("4. Area")
    print("5. Smoothness")
    print("6. Compactness")
    print("7. Concavity")
    print("8. Concave Points")
    print("9. Symmetry")
    print("10. Fractal Dimension")
    inputX = int(input("Input x coordinate number : "))
    inputY = int(input("Input y coordinate number : "))
    ch.displayConvexHull(df, data, inputX, inputY)
else :
    print("Invalid Input.")
```

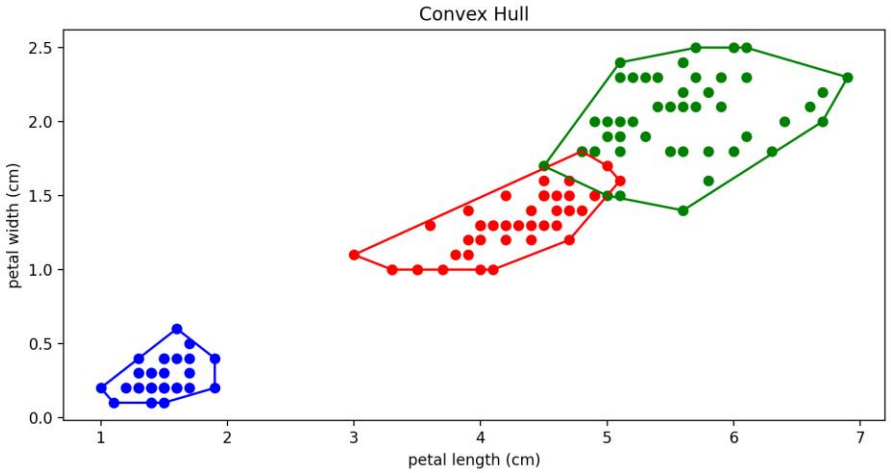
C. Testing Program

- Interface input/output program

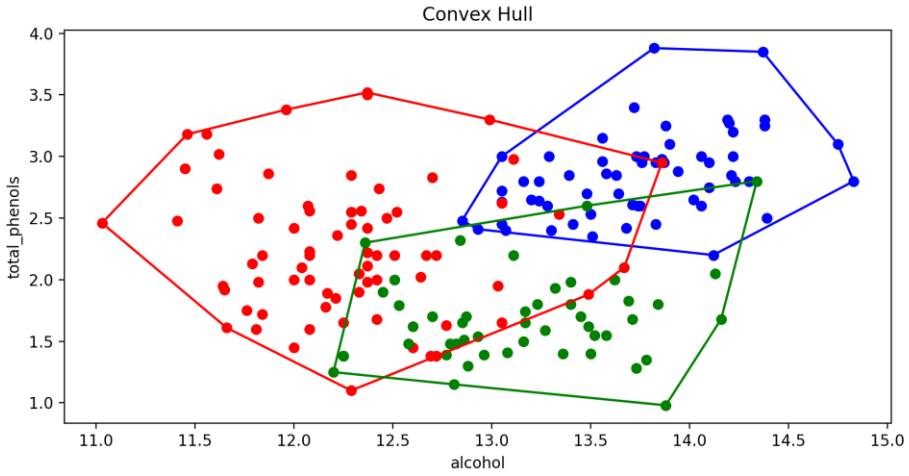
```
PS C:\Bismillah Nugas\Tucil2_13520085> python -u "c:\Bismillah Nugas\Tucil2_13520085\src\main.py"
Available dataset :
1. Iris Dataset
2. Wine Dataset
3. Breast Cancer Dataset
Input dataset number : 1
Available data :
1. Sepal Length
2. Sepal Width
3. Petal Length
4. Petal Width
Input x coordinate number : 3
Input y coordinate number : 4
Set of hull points :
{(1.9, 0.4), (1.0, 0.2), (1.9, 0.2), (1.1, 0.1), (1.5, 0.1), (1.6, 0.6)}
Set of hull points :
{(1.9, 0.4), (5.1, 1.6), (4.1, 1.0), (1.0, 0.2), (1.9, 0.2), (1.1, 0.1), (1.5, 0.1), (3.0, 1.1), (3.3, 1.0), (4.7, 1
.2), (5.0, 1.7), (1.6, 0.6), (4.8, 1.8)}
Set of hull points :
{(5.7, 2.5), (6.1, 2.5), (6.9, 2.3), (6.7, 2.0), (5.1, 1.6), (5.0, 1.5), (1.1, 0.1), (1.5, 0.1), (5.0, 1.7), (5.6, 1
.4), (4.1, 1.0), (4.7, 1.2), (4.5, 1.7), (4.8, 1.8), (1.9, 0.4), (1.0, 0.2), (1.9, 0.2), (1.6, 0.6), (3.0, 1.1), (3.
3, 1.0), (5.1, 2.4)}
```

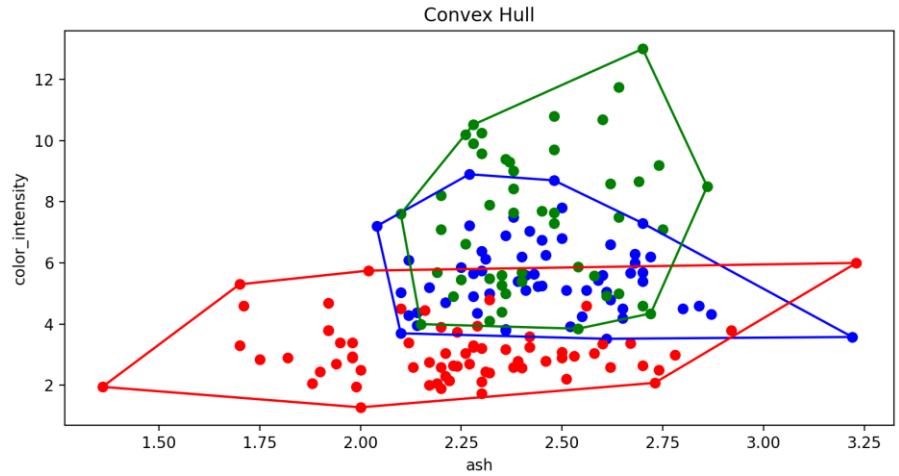
- Testing dataset
  1. Iris Dataset



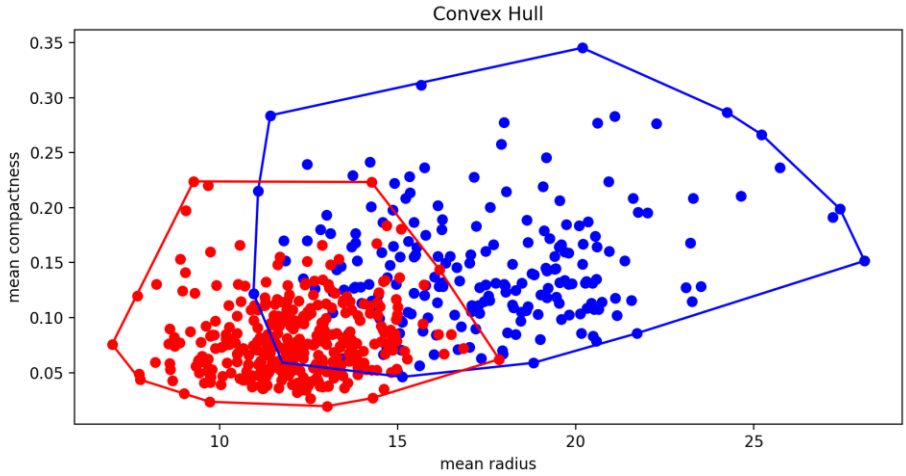


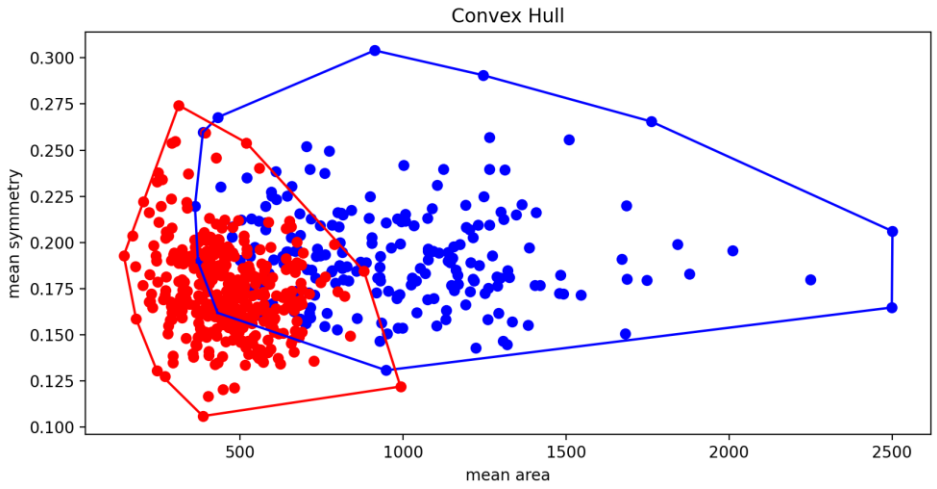
  2. Wine Dataset





3. Breast Cancer Dataset





D. Link Source Code

Berikut dilampirkan link menuju source code dan laporan https://github.com/ubaidalih/Tucil2_13520085

E. Lembar Checklist

Poin	Ya	Tidak
1.Program myConvexHull berhasil dibuat tanpa kesalahan	v	
2. Convex Hull yang dihasilkan sudah benar	v	
3. Pustaka myConvexHull dapat digunakan untuk menampilkan convex hull setiap label dengan warna yang berbeda.	v	
4. Bonus: program dapat menerima input dan menuliskan output untuk dataset lainnya	v	