Laporan Tugas Kecil 2 IF2211
Strategi Algoritma

Mencari Convex Hull dengan Algoritma Divide & Conquer

oleh
Aloysius Gilang Pramudya (13520147)

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

2022

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

A. Algoritma *Divide and Conquer*

Langkah-langkah algoritma *Divide and Conquer* yang digunakan pada program ini adalah sebagai berikut :

1. Kumpulan titik diurutkan berdasarkan nilai absis yang menaik, dan jika ada nilai absis yang sama, maka diurutkan dengan nilai ordinat yang menaik.
2. Cari titik ektrim yaitu titik paling kiri (min_extreme) dan titik paling kanan (max_extreme). Garis yang menghubungkan min_extreme dan max_extreme membagi kumpulan titik menjadi dua bagian yaitu S1 (kumpulan titik sebelah kiri atau atas garis) dan S2 (kumpulan titik di sebelah kanan atau bawah garis).
3. Kumpulan titik pada S1 membentuk convex hull bagian atas, dan kumpulan titik pada S2 membentuk convex hull bagian bawah. S adalah gabungan list S1 dan S2. Terapkan algoritma Divide and Conquer pada S1 dan S2 secara rekursif.
4. Pada S1 dan S2 jika tidak terdapat titik lagi maka *do nothing* (basis). Jika tersisa satu titik maka langsung tambahkan ke S (basis). Untuk semua titik pada list pilih titik dengan jarak terjauh dari garis yang dibentuk dari min_extreme dan max_extreme (new_max).
5. Tentukan kumpulan titik yang berada diluar daerah segitiga yaitu diluar garis min_extreme-new_max (menjadi bagian S1) dan garis new_max-max_extreme (menjadi bagian S2).
6. Lakukan hal yang sama (butir 4 dan 5) untuk bagian S2.
7. Kembalikan pasangan titik yang dihasilkan.

B. Source Code Program

Source Code untuk pustaka myConvexHull

```
import numpy as np
import numpy.linalg as nl


# menghitung jarak titik p3 dari garis yang dibentuk oleh titik p1 dan p2
def PointToLineDis(p1,p2,p3):
    p1 = np.asfarray(p1)
    p2 = np.asfarray(p2)
```

```
    p3 = np.asfarray(p3)
    d = nl.norm(np.cross(p2-p1, p1-p3))/nl.norm(p2-p1)
    return d

# mengurutkan list of point berdasarkan x kemudian y menaik
def sortPoints(list):
    return sorted(list , key=lambda x: [x[0], x[1]])

# mengurutkan list of point berdasarkan x kemudian y menurun
def sortPointsR(list):
    return sorted(list , key=lambda x: [x[0], x[1]], reverse=True)

# menghitung nilai determinan
def determinan(point, min_e, max_e):
    x1 = max_e[0] ; y1 = max_e[1] ; x2 = min_e[0] ; y2 = min_e[1] ; x3 = point[0] ; y3 = point[1]
    D = (x1*y2) + (x3*y1) + (x2*y3) - (x3*y2) - (x2*y1) - (x1*y3)
    return D

#return list of outermost point untuk bagian kiri/atas
def DAC (list, min_e, max_e):
    # basis
    if len(list) == 0:
        return []
    elif len(list) == 1 :
        if PointToLineDis(min_e,max_e,list) == 0.0 :
            return []
        else : return list
    else :
        max_distance = 0
        for point in list:
            d = PointToLineDis(min_e,max_e,point)
            if d > max_distance :
                max_distance = d
                new_max = point

        S1 = [] ; S2 = [] ; S = []

        for point in list :
            if determinan(point, min_e, new_max) < -0.1e-5 :
```

```
                S1.append(point)
            elif determinan(point,new_max,max_e) < -0.1e-5 :
                S2.append(point)

        S.extend(DAC(S1,min_e,new_max))
        S.extend([new_max])
        S.extend(DAC(S2,new_max,max_e))
        return S


#return list of outermost point untuk bagian kanan/bawah
def DAC2 (list, min_e, max_e):
    # basis
    if len(list) == 0:
        return []
    elif len(list) == 1 :
        if PointToLineDis(min_e,max_e,list) == 0.0 :
            return []
        else : return list
    else :
        max_distance = 0
        for point in list:
            d = PointToLineDis(min_e,max_e,point)
            if d > max_distance :
                max_distance = d
                new_max = point

        S1 = [] ; S2 = [] ; S = []

        for point in list :
            if determinan(point, min_e, new_max) > 0.1e-5 :
                S1.append(point)
            elif determinan(point,new_max,max_e) > 0.1e-5 :
                S2.append(point)

        S.extend(DAC2(S1,min_e,new_max))
        S.extend([new_max])
        S.extend(DAC2(S2,new_max,max_e))

        return S
```

```
# main function
def myConvexHull (list):
    # urutkan list of point berdasarkan x kemudian y menaik
    list = sortPoints(list)
    # cari titik paling kiri/kanan, bagi dua bagian list
    min_extreme = min(list, key=lambda x: [x[0], x[1]])
    max_extreme = max(list, key=lambda x: [x[0], x[1]])
    S1 = [] ; S2 = [] ; S = []
    for point in list :
        if determinan(point, min_extreme, max_extreme) > 0.1e-5 :
            S2.append(point)
        elif determinan(point,min_extreme,max_extreme) < -0.1e-5 :
            S1.append(point)
    # S berisi outermost point dengan urutan clockwise
    S.extend([min_extreme])
    S.extend(sortPoints(DAC(S1, min_extreme,max_extreme)))
    S.extend([max_extreme])
    S.extend(sortPointsR((DAC2(S2, min_extreme,max_extreme))))
    S.extend([min_extreme])
    return S
```

Source Code untuk pustaka main program

```
import convexHull as mch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
```

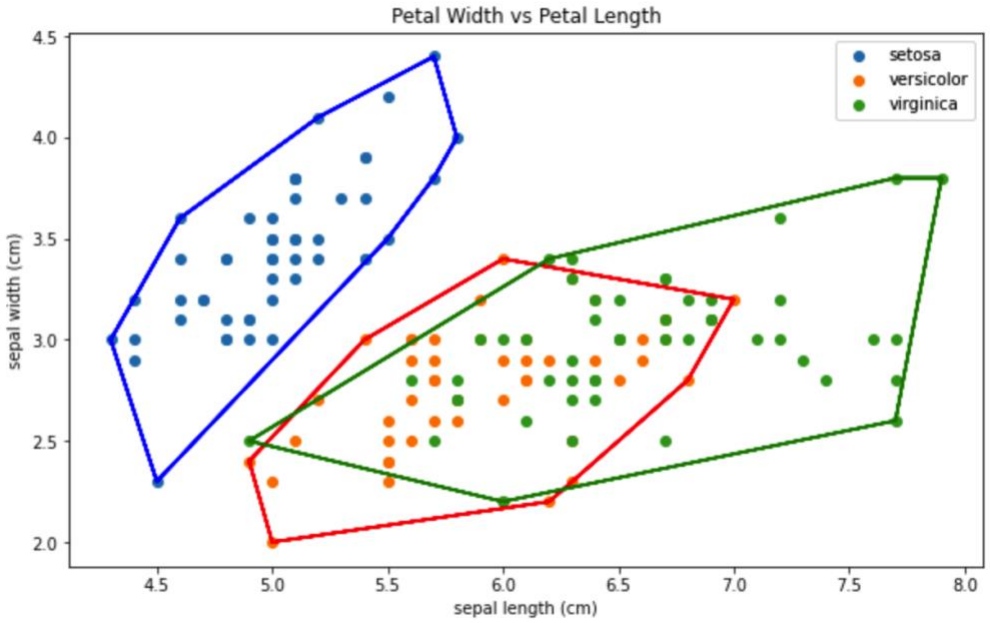
```
plt.xlabel(data.feature_names[0])          # axis point
plt.ylabel(data.feature_names[1])          # ordinate
for i in range(len(data.target_names)):    # akan ada 3 convex hull karena target[0..2]
   bucket = df[df['Target'] == i]          # pembagian data berdasarkan target
   bucket = bucket.iloc[:,[2,3]].values    # return 2d array [[a,b]] from attribute Petal width and Petal length
   hull = mch.myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide &
Conquer
   plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
   print(hull)
   x = [x[0] for x in hull]
   y = [x[1] for x in hull]
   for simplex in hull:          # visualizing convex hull
      plt.plot(x, y, colors[i])
plt.legend()
```

C. Screenshot hasil program

1. Test dataset iris (petal-length, petal-width)

```
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[0])           # axis point
plt.ylabel(data.feature_names[1])           # ordinate
for i in range(len(data.target_names)):     # akan ada 3 convex hull ka
    bucket = df[df['Target'] == i]          # pembagian data berdasarka
    bucket = bucket.iloc[:,[0,1]].values    # return 2d array [[a,b]] f
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil impleme
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    x = [x[0] for x in hull]
    y = [x[1] for x in hull]
    for simplex in hull:          # visualizing convex hull
        plt.plot(x, y, colors[i])
plt.legend()
```

```
Out[22]: <matplotlib.legend.Legend at 0x7fb12fb87790>
```

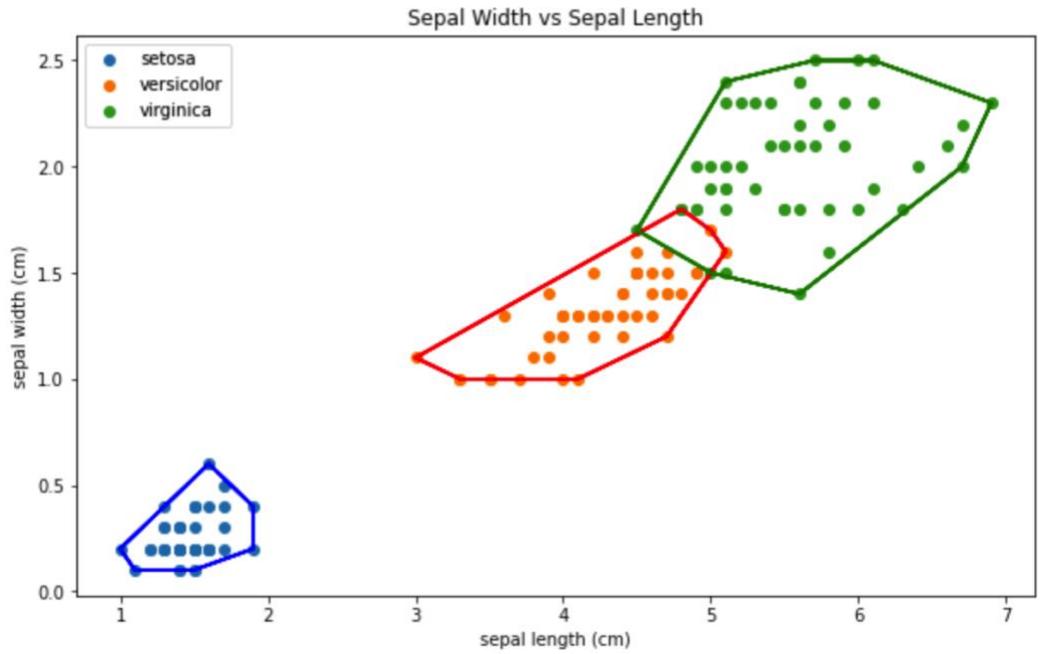


2. Test dataset iris (sepal-length, sepal-width)

```
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()

#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length')
plt.xlabel(data.feature_names[0])           # axis point
plt.ylabel(data.feature_names[1])           # ordinate
for i in range(len(data.target_names)):     # akan ada 3 convex hull kar
    bucket = df[df['Target'] == i]          # pembagian data berdasarkan
    bucket = bucket.iloc[:,[2,3]].values    # return 2d array [[a,b]] fr
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil impleme
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    x = [x[0] for x in hull]
    y = [x[1] for x in hull]
    for simplex in hull:          # visualizing convex hull
        plt.plot(x, y, colors[i])
plt.legend()
```

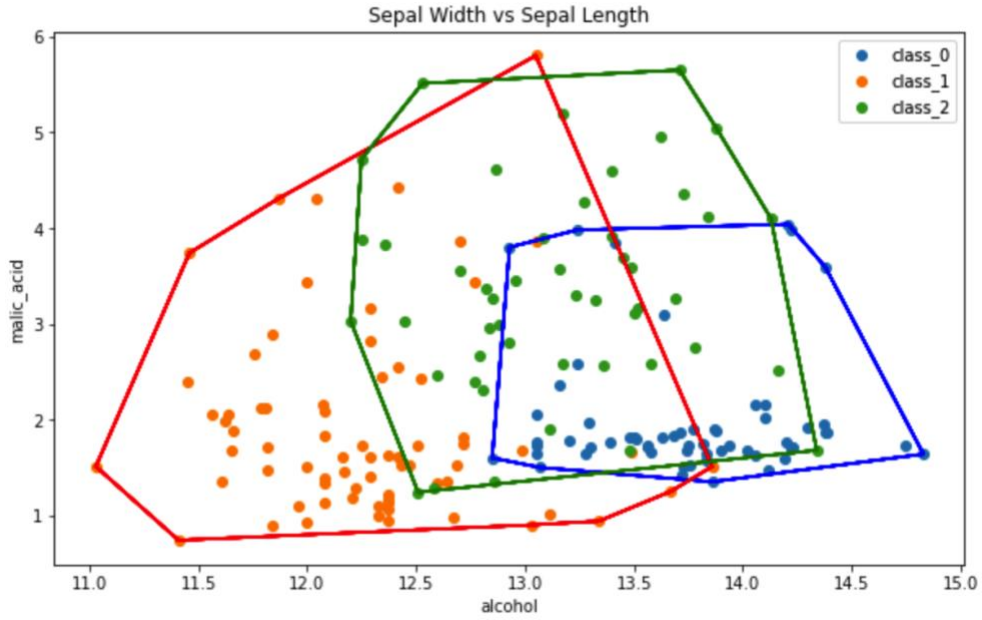
```
Out[23]: <matplotlib.legend.Legend at 0x7fb1312719d0>
```



3. Test dataset wine (malic_acid vs alcohol)

```
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('malic_acid vs alcohol')
plt.xlabel(data.feature_names[0])           # axis point
plt.ylabel(data.feature_names[1])           # ordinate
for i in range(len(data.target_names)):     # akan ada 3 convex hull ka
    bucket = df[df['Target'] == i]          # pembagian data berdasarka
    bucket = bucket.iloc[:,[0,1]].values    # return 2d array [[a,b]] f
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implem
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    x = [x[0] for x in hull]
    y = [x[1] for x in hull]
    for simplex in hull:          # visualizing convex hull
        plt.plot(x, y, colors[i])
plt.legend()
```

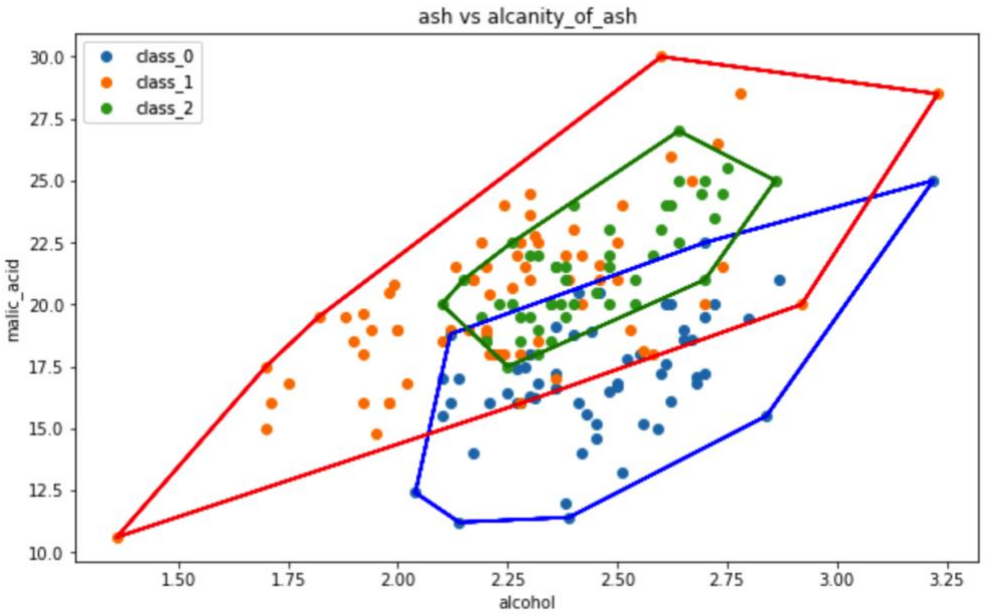
Out[24]: <matplotlib.legend.Legend at 0x7fb130ee33d0>



4. Test dataset wine (ash vs alcanity_of_ash) |

```
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('ash vs aclanity_of_ash')
plt.xlabel(data.feature_names[0])           # axis point
plt.ylabel(data.feature_names[1])           # ordinate
for i in range(len(data.target_names)):     # akan ada 3 convex hull karena target[0..2]
    bucket = df[df['Target'] == i]          # pembagian data berdasarkan target
    bucket = bucket.iloc[:,[2,3]].values    # return 2d array [[a,b]] from attribute Petal width and Petal length
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    x = [x[0] for x in hull]
    y = [x[1] for x in hull]
    for simplex in hull:          # visualizing convex hull
        plt.plot(x, y, colors[i])
plt.legend()
```

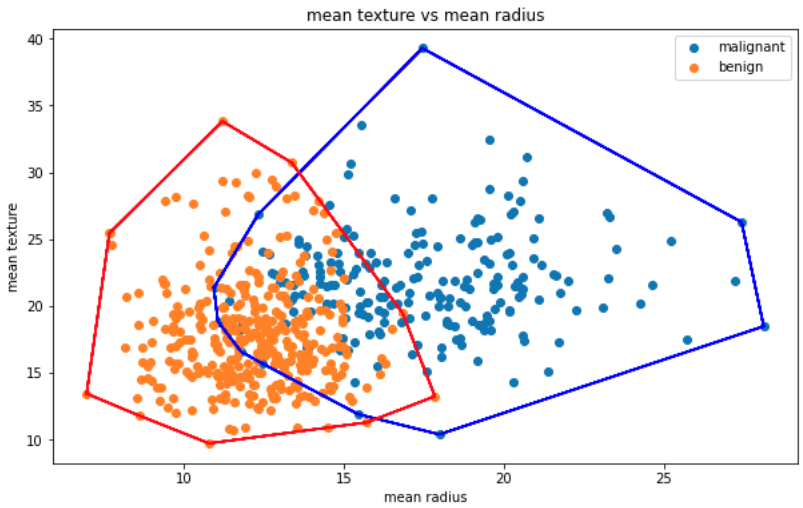
```
Out[35]: <matplotlib.legend.Legend at 0x7fb131c6d790>
```



5. Test dataset breast_cancer (mean texture, mean radius)

```
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('mean texture vs mean radius')
plt.xlabel(data.feature_names[0])           # axis point
plt.ylabel(data.feature_names[1])           # ordinate
for i in range(len(data.target_names)):     # akan ada 3 convex hull karena target[0..2]
    bucket = df[df['Target'] == i]          # pembagian data berdasarkan target
    bucket = bucket.iloc[:,[0,1]].values    # return 2d array [[a,b]] from attribute Petal width and Petal length
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    x = [x[0] for x in hull]
    y = [x[1] for x in hull]
    for simplex in hull:          # visualizing convex hull
        plt.plot(x, y, colors[i])
plt.legend()
```

```
Out[8]: <matplotlib.legend.Legend at 0x7fbc10a61f10>
```



D. Alamat drive yang berisi kode program

https://github.com/Aloysiusgilang/myConvexHull.git