LAPORAN TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA

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

Disusun oleh:

Ilham Bintang Nurmansyah 13520102

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

1. Penjelasan Algoritma *Divide and Conquer*

Algoritma yang saya lakukan yaitu pertama tama melakukan konversi dari array bertipe numpy array, menjadi python list biasa. Lalu mencari titik dengan absis paling kecil dan besar, menarik garis diantara kedua titik terebut dan membagi himpunan koordinat menjadi 2, yaitu set 1 (sisi atas/kiri) dan set 2 (yaitu sisi kanan/bawah). Setelah itu, melakukan divide and conquer pada masing masing set tersebut secara rekursif. Untuk divide and conquer pada masing masing set titik, algoritmanya kurang lebih melakukan return bila isi dari list points (jumlah titik pada sisi yang akan dicari) sudah 0, artinya sudah tidak ada lagi titik diluar itu. Bisa isi dari list points belum 0, maka akan mencari titik terjauh dari garis A dan B yaitu titik C ,lalu titik C dihapus dari himpunan titik dan dimasukkan pada himpunan solusi, setelah itu himpunan titik dibagi 2 lagi seperti saat pertama, sehingga himpunan titik dibagi 2 berdasarkan garis AC dan garis BC dan hanya diambil himpunan titik yang akan dicari berikutnya (bisa atas/kiri atau bawah/kanan) lalu fungsi ini dipanggil kembali hingga himpunan titik yang akan dicari isinya kosong. Setelah semua fungsi rekursif selesai, maka array jawaban yang isinya merupakan himpunan titik hull sudah terisi. Setelah itu urutan himpunan titik hull di sort agar tidak ada garis yang saling menabrak. Lalu setiap titik dipasangkan agar menjadi garis dan dimasukkan kedalam array baru yang merupakan hasil akhir berbentuk pasangan titik yang membentuk convex hull.

2. Kode program

myConvexHull.py

```
src > myConvexHull.py > myConvexHull
    You, 2 minutes ago | 1 author (You)
1   import math
2   # main function, untuk menkonversi numpy array ke array biasa, mencari min max, mencari titik hull dan dipasangkan menjadi garis
3   def myConvexHull(unconv):
4       solution = []
5       solutionPair = []
6       points = convertToArr(unconv)
7       minimum,maximum = findExtreme(points)
8       solution.append(minimum)
9       solution.append(maximum)
10
11      kiri,kanan = divide(points,minimum,maximum)
12      hullSet1(kiri,minimum,maximum,solution)
13      hullSet2(kanan,minimum,maximum,solution)
14
15      x = sum(point[0] for point in solution)/len(solution)
16      y = sum(point[1] for point in solution)/len(solution)
17      solution.sort(key = lambda point: math.atan2(point[0]-x,point[1]-y))      You, 13 hours ago • commit convexHull
18
19      for i in range(len(solution)):
20          if i == len(solution)-1:
21              solutionPair.append([solution[i],solution[0]])
22          else:
23              solutionPair.append([solution[i],solution[i+1]])
24      return solutionPair
25
26  # untuk mengkonversi numpy array ke array biasa
27  def convertToArr(points):
28      arr = []
29      for i in range(len(points)):
30          arr.append([float(points[i][0]),float(points[i][1])])
31      return arr
32
33  # untuk mencari titik maksimum dan minimum
34  def findExtreme(points):
35      minimum = points[0]
36      maximum = points[0]
37
38      for point in points:
```

src > myConvexHull.py > myConvexHull

```python
        if point[0] <= minimum[0]:
            minimum = point
        if point[0] >= maximum[0]:
            maximum = point
    return minimum, maximum

# untuk mencari determinan
def determinan(x1,x2,x3):
    return(x1[0]*x2[1]) + (x1[1]*x3[0]) + (x2[0]*x3[1]) - (x3[0]*x2[1]) - (x3[1]*x1[0]) - (x2[0]*x1[1])

# untuk membagi hull menjadi set 1 dan set 2 sesuai besar determinan (+ atau -)
def divide(points,minimum,maximum):
    set1 = []
    set2 = []

    for point in points:
        if point!=minimum and point!=maximum:
            if(determinan(minimum,maximum,point) > 0):
                set1.append(point)
            if(determinan(minimum,maximum,point) < 0):
                set2.append(point)
    return set1,set2

# untuk mencari titik terjauh dari garis
def pointDistace(points,minimum,maximum):
    jarak = 0
    index = 0

    for i in range(len(points)):
        jaraktmp = abs((points[i][0]-minimum[0])*(maximum[1]-minimum[1]) - (maximum[0]-minimum[0])*(points[i][1]-minimum[1]))
        if jaraktmp > jarak:
            jarak = jaraktmp
            index = i

    return points[index]

# untuk mencari titik hull dari set 1(atas) secara rekursif setelah di divide
def hullSet1(points,minimum,maximum,solution):
    if(len(points) == 0):
```

src > myConvexHull.py > myConvexHull

```python

# untuk mencari titik hull dari set 1(atas) secara rekursif setelah di divide
def hullSet1(points,minimum,maximum,solution):
    if(len(points) == 0):
        return
    else:
        pointmax = pointDistace(points,minimum,maximum)
        solution.append(pointmax)
        points.remove(pointmax)
        x1,a = divide(points,minimum,pointmax)
        x2,b = divide(points,pointmax,maximum)

        hullSet1(x1,minimum,pointmax,solution)
        hullSet1(x2,pointmax,maximum,solution)

# untuk mencari titik hull dari set 2(bawah) secara rekursif setelah di divide
def hullSet2(points,minimum,maximum,solution):
    if(len(points) == 0):
        return
    else:
        pointmax = pointDistace(points,minimum,maximum)
        solution.append(pointmax)
        points.remove(pointmax)
        a,x1 = divide(points,minimum,pointmax)
        b,x2 = divide(points,pointmax,maximum)

        hullSet2(x1,minimum,pointmax,solution)
        hullSet2(x2,pointmax,maximum,solution)
```

Visualizer.ipynb:

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import myConvexHull

data = datasets.load_iris()


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

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(3):
    bucket = df[df['Target'] == i]
    bucket= bucket.iloc[:,[2,3]].values
    hull= myConvexHull(bucket)

    plt.scatter(bucket[:,0],bucket[:,1], label=data.target_names[i])

    for x in range(0,len(hull)):
        listX = [hull[x][0][0],hull[x][1][0]]
        listY = [hull[x][0][1],hull[x][1][1]]
        plt.plot(listX,listY,color=colors[i])
plt.legend()
```

✓ 0.2s Python

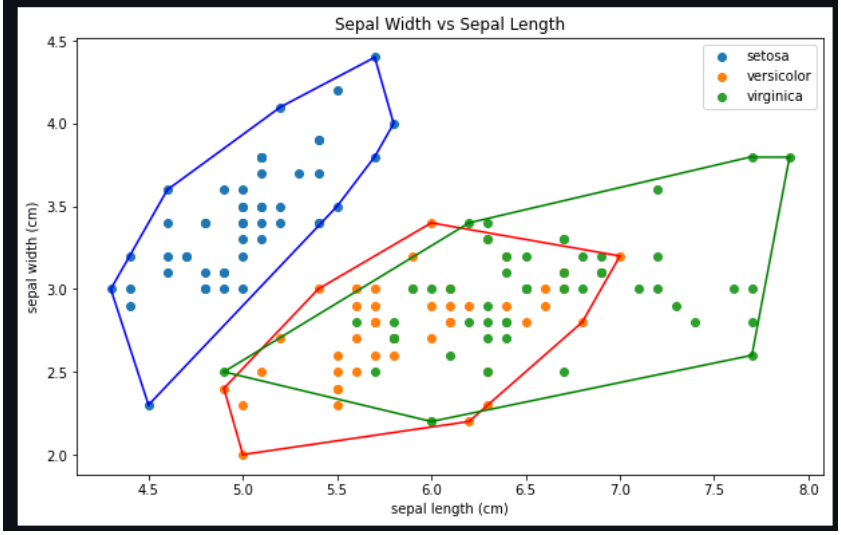
3. Screenshot Input/Output Program

a. Sepal-length vs Sepal-width

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import myConvexHull

data = datasets.load_iris()


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

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
for i in range(3):
    bucket = df[df['Target'] == i]
    bucket= bucket.iloc[:,[0,1]].values
    hull= myConvexHull(bucket)

    plt.scatter(bucket[:,0],bucket[:,1], label=data.target_names[i])

    for x in range(0,len(hull)):
        listX = [hull[x][0][0],hull[x][1][0]]
        listY = [hull[x][0][1],hull[x][1][1]]
        plt.plot(listX,listY,color=colors[i])
plt.legend()
```

✓ 0.2s

```
(150, 5)
<matplotlib.legend.Legend at 0x1f26e6473d0>
```

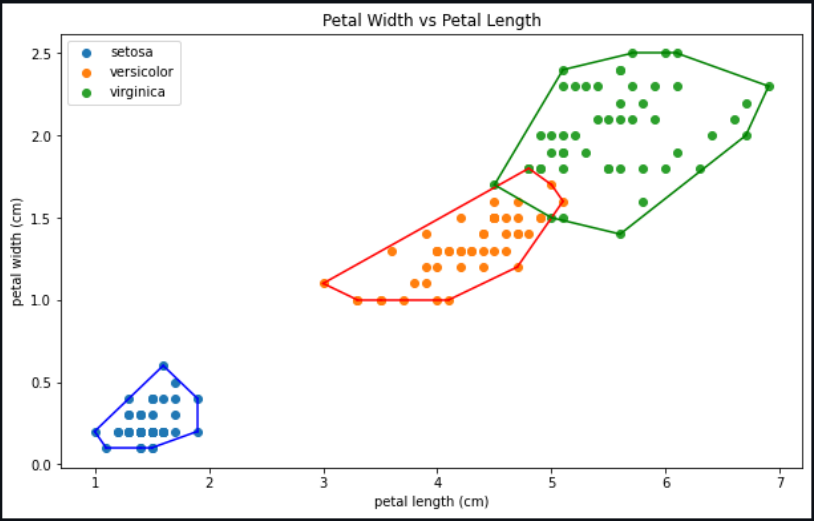


b. Petal-Width vs Petal-Length

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import myConvexHull

data = datasets.load_iris()


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

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(3):
    bucket = df[df['Target'] == i]
    bucket= bucket.iloc[:,[2,3]].values
    hull= myConvexHull(bucket)

    plt.scatter(bucket[:,0],bucket[:,1], label=data.target_names[i])

    for x in range(0,len(hull)):
        listX = [hull[x][0][0],hull[x][1][0]]
        listY = [hull[x][0][1],hull[x][1][1]]
        plt.plot(listX,listY,color=colors[i])
plt.legend()
```

✓ 1.7s

```
(150, 5)
<matplotlib.legend.Legend at 0x1244e9173a0>
```

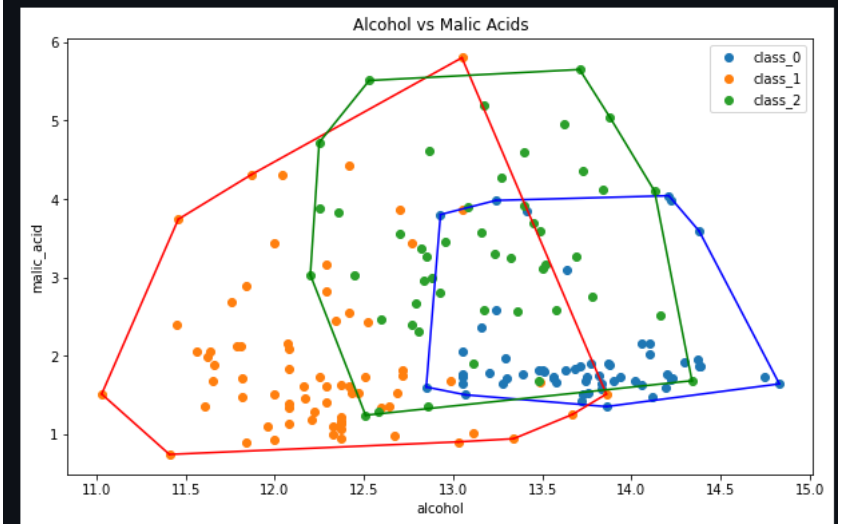


c. Alcohol vs Malic Acid (bonus)

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import myConvexHull

data = datasets.load_wine()


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

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Alcohol vs Malic Acids')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
for i in range(3):
    bucket = df[df['Target'] == i]
    bucket= bucket.iloc[:,[0,1]].values
    hull= myConvexHull(bucket)

    plt.scatter(bucket[:,0],bucket[:,1], label=data.target_names[i])

    for x in range(0,len(hull)):
        x = [hull[x][0][0],hull[x][1][0]]
        y = [hull[x][0][1],hull[x][1][1]]
        plt.plot(x,y,color=colors[i])
plt.legend()
```

✓ 0.2s

```
(178, 14)
<matplotlib.legend.Legend at 0x12450afcfd0>
```



4. **Link Github Source Code:**
   https://github.com/Hambinn/Tucil-2-Stima