TUCIL 2 IF2211 STRATEGI ALGORITMA

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

Panawar Hasibuan 13517129

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021/2022

Daftar Isi

0.1 Algoritma devide and conquer

Algoritma Divide and Conquer adalah algoritma pemecahan masalah dengan cara membagi masalah kedalam bagian-bagian kecil, kemudian menyelesaikan masalah tersebut dari bagian yang paling rendah / bawah.

Library ini adalah library pembentuk *convex hull* yang menerima masukan kumpulan tuple dua dimensi yang merepresentasikan titik pada bidang dua dimensi (titik absis dan ordinat). Suatu poligon disebut *convex* jika tidak ada sisi yang berpotongan dengan segmen garis yang dihubungkan oleh dua titik sudut lainnya.

Adapun proses yang dilakukan untuk mencari bidang *convex* semua titik dari masukan library ini berada di dalam suatu bidang *convex* yang sudut-sudut pembentuknya adalah himpunan bagian dari titik-titik masukan (disebut *convex hull*) adalah sebagai berikut.

- Himpunan solusi adalah himpunan titik yang menjadi penyusun *convex hull*
- Semua masukan akan diurutkan menaik berdasarkan absis, untuk absis yang sama diurutkan berdasarkan ordinatnya.
- Titik terkecil (P1) dan terbesar (P2) hasil sorting dimasukkan kedalam himpunan solusi (karna semua titik lainnya berada diantara dua titik ini, jelas kedua titik ini adalah bagian penyusun *convex hull* yang dicari)
- Sebuah garis l dibentuk dari menghubungkan kedua titik, algoritma *devide and conquer* dimulai
- Titik-titik lainnya dikelompokkan menjadi dua kelompok berdasarkan posisinya terhadap garis l: $y = f(x)$
  - titik Px_1, y_1 dimasukkan kedalam kelompok 1 jika $y_1 > f(x_1)$
  - titik Qx_2, y_2 dimasukkan kedalam kelompok 2 jika $y_2 > f(x_2)$
  - titik Rx_2, y_2 tidak dimasukkan kedalam kelompok manapun jika $y_2 = f(x_2)$ karna untuk kasus ini, titik tersebut berada digaris l sehingga sudah pasti tidak termasuk penyusun *convex hull*
- Langkah awal dilakukan masing-masing pada kelompok 1 dan kelompok 2 dengan memasukkan P1 pada kelompok 1 dan P2 pada kelompok 2
- Metode pengurutan dilakukan bergantian terhadap absis dan ordinat.
- Langkah berhenti jika pada suatu kelompok titik hanya ada 3 titik

0.2 Source Program

Kode Program

Algorithm 1: MyConvexHull.py

```
"""
# implementasi convex hull dengan algoritma devide and conquer
# @author Panawar Hasibuan
# @email 13517129@std.stei.itb.ac.id
"""
import numpy as np
```

```
7
8  class MyConvexHull:
9      """
10     data: numpy.array(N,2)
11     """
12     def __init__(self, bucket):
13         self.bucket = np.copy(bucket)
14         x = np.copy(self.bucket)
15         x = x[x[:,1].argsort()]
16         self.sorted = x[x[:,0].argsort(kind='mergesort')]
17         self.simplices =self.hull(self.setIdx(self.sorted))
18
19     """
20     Algoritma Devide and Conquer
21     prekondisi, data terurut
22     return, set index penyusun convex hull
23     """
24     def DnC(self,data,state):
25         hull = np.unique([-99])
26         if len(data) > 3:
27             #rekurens
28             #simpan pembentuk convexhull
29             q = np.copy(data[0])
30             r = np.copy(data[-1])
31             hull = np.union1d(self.getIdx(q),self.getIdx(r))
32             #Karna state awal True, dan data terurut terhadap x
33             #urutkan data
34             if not state:
35                 #terurut terhadap x
36                 data = data[data[:,1].argsort()]
37                 data = data[data[:,0].argsort(kind='mergesort')]
38             else:
39                 #terurut terhadap y
40                 data = data[data[:,0].argsort()]
41                 data = data[data[:,1].argsort(kind='mergesort')]
42
43             if (r[0]==data[0,0] and r[1]==data[0,1]) or (q[0]==data[0,0] and q
                   [1]==data[0,1]): #r==data[0] or q==data[0]
44                 p = np.copy(data[-1])
45             else:
46                 p = np.copy(data[0])
47             first = np.array((p,q),dtype=int) #menampung titik kelompok 1
48             second = np.array((p,r),dtype=int) #menampung titik kelompok 2
49             for i in range(1,len(data)-1):
50                 if self.isBesideQ(data[i],p,q,r):
51                     first = np.vstack((first,np.copy(data[i])))
52                 elif self.isBesideR(data[i],p,q,r):
53                     second = np.vstack((second,np.copy(data[i])))
54             hull = np.union1d(hull,self.DnC(first,not state))
55             hull = np.union1d(hull,self.DnC(second,not state))
56         else:
57             #kondisi stop
58             for e in data:
59                 hull = np.union1d(hull,self.getIdx(e))
60         return hull
61
62     """
63     cek apakah  titik point terletak berlawanan dengan r terhadap garis yang
           menghubungkan p dan q
64     proses mendapat perhitungan di algoritma ini ada di laporan
65     """
66     def isBesideR(self,point,p,q,r):
```

```
67          x1,y1 = p[0],p[1]
68          x2,y2 = r[0],r[1]
69          if x1 == x2: #garis lurus horizontal
70              return (point[0]-x1)*(q[0]-x1) < 0
71          elif y1 == y2: #garis lurus vertikal
72              return (point[1]-y1)*(q[1]-y1) < 0
73          else:
74              fpoint=point[1]*(x2-x1)
75              xpoint=point[0]*(y2-y1)+y1*(x2-x1)-x1*(y2-y1)
76              fq=q[1]*(x2-x1)
77              xq=q[0]*(y2-y1)+y1*(x2-x1)-x1*(y2-y1)
78              return (fpoint-xpoint)*(fq-xq)<0
79
80      """
81      cek apakah  titik point terletak berlawanan dengan r terhadap garis yang
            menghubungkan p dan q
82      proses mendapat perhitungan di algoritma ini ada di laporan
83      """
84      def isBesideQ(self,point,p,q,r):
85          x1,y1 = p[0],p[1]
86          x2,y2 = q[0],q[1]
87          if x1 == x2: #garis lurus horizontal
88              return (point[0]-x1)*(r[0]-x1) < 0
89          elif y1 == y2: #garis lurus vertikal
90              return (point[1]-y1)*(r[1]-y1) < 0
91          else:
92              fpoint=point[1]*(x2-x1)
93              xpoint=point[0]*(y2-y1)+y1*(x2-x1)-x1*(y2-y1)
94              fr=r[1]*(x2-x1)
95              xr=r[0]*(y2-y1)+y1*(x2-x1)-x1*(y2-y1)
96              return (fpoint-xpoint)*(fr-xr)<0
97
98      """
99      Mengembalikan index dari data di self.bucket
100     mengembalikan -1 jika data tidak ada di self.bcuket
101     """
102     def getIdx(self,data):
103         for i in range(len(self.bucket)):
104             if self.bucket[i][0]== data[0] and self.bucket[i][1]==data[1]:
105                 return i
106                 break
107             elif i == len(self.bucket)-1:
108                 return -1
109
110     """
111     prekondisi: data terurut
112     membagi data menjadi dua, up dan down, dipisahkan oleh data[0] dan data
            [-1]
113     """
114     def devide(self,data,up,down):
115         x1,y1 = data[0,0],data[0,1]
116         x2,y2 = data[-1,0],data[-1,1]
117         i = 1
118         while i < len(data)-1:
119             p = data[i]
120             if x1 == x2: #garis lurus horizontal
121                 if (p[0]-x1) < 0:
122                     #up
123                     up = np.vstack((up,p))
124                 else:
125                     #down
126                     down = np.vstack((down,p))
```

```
127             elif y1 == y2: #garis lurus vertikal
128                 if (p[1]-y1) < 0:
129                     #up
130                     up = np.vstack((up,p))
131                 else:
132                     #down
133                     down = np.vstack((down,p))
134             else:
135                 fp=p[1]*(x2-x1)
136                 xp=p[0]*(y2-y1)+y1*(x2-x1)-x1*(y2-y1)
137                 if (fp-xp)<0:
138                     #up
139                     up = np.vstack((up,p))
140                 else:
141                     #down
142                     down = np.vstack((down,p))
143             i = i+1
144         up = np.vstack((up,data[-1]))
145         down = np.vstack((down,data[-1]))
146 
147     """
148     Mengembalikan himpunan index dari pembangun vertex hull
149     """
150     def setIdx(self,sort):
151         up = np.array(sort[0])
152         down = np.array(sort[0])
153         self.devide(sort,up,down)
154         return np.union1d(self.DnC(up,True),self.DnC(down,True))
155 
156     """
157     Mengembalikan ndarray index pembangun vertex hull, berbentuk simplices
158     """
159     def hull(self,set):
160         return set
```

Algorithm 2: Tucil2$_1$3517129.*ipynb*

```
#persiapkan data
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
data = datasets.load_iris()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
from myconvexhull import MyConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
for i in range(len(data.target_names)):
    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 simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

0.3 Skrinsut input/output

Berikut beberapa hasil skrinsut beberapa keluaran program

0.4 Dokumentasi

0.4.1 Alamat Drive

https://github.com/panawar/myConvexHull

0.4.2 Kesimpulan

Referensi

https://piptools.net/algoritma-divide-conquer/
diakses pada
https://www.artikata.com/arti-41099-convex+polygon.html
diakses pada:

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

Tabel 1: tabel uji