Laporan Tugas Kecil 1 IF2211 Strategi Algoritma
Semester II tahun 2021/2022

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

Dipersiapkan oleh:

Nama: Jeremy Rionaldo Pasaribu

NIM: 13520082

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

Daftar Isi

BAB 1: Algoritma *Divide and Conquer*

Misalkan S adalah himpunan titik sebanyak n dengan n > 1. Berikut algoritma *dive and conquer* secara garis besar:

1. Urutkan himpunan titik dalam *S* berdasarkan nilai absis yang menaik. Jika terdapat nilai absis yang sama, urutkan berdasarkan nilai ordinat yang menaik.
2. Cari titik dalam *S* yang memiliki absis minimum (misal p_1) serta titik yang memiliki absis maksimum (misal p_n). Misalkan g adalah garis yang menghubungkan p_1 dan p_n.
3. Bagi himpunan titik dalam *S* selain p_1 dan p_n menjadi dua bagian yaitu S_1 (kumpulan titik di sebelah kiri garis g) dan S_2 (kumpulan titik di sebelah kanan garis g). Jika terdapat titik yang dilalui garis g, titik tersebut diabaikan.
4. Dalam sebuah bagian (S_1 dan S_2), terdapat dua kemungkinan:
   - Jika S_1 kosong (tidak terdapat titik), p_1 dan p_n menjadi titik terluar yang membentuk *convex hull*
   - Jika S_1 tidak kosong, cari titik yang memiliki jarak terjauh dari titik p_1 dan p_n (misal p_{max}). Jika tedapat beberapa titik memiliki jarak yang sama, pilih titik yang membentuk sudut paling besar terhadap p_1 dan p_n.
5. Setelah mendapatkan p_{max}, tentukan kumpulan titik yang berada di sebelah kiri garis p_1 p_{max} sebagai bagian S_1 yang baru, dan di sebelah kanan garis p_{max} p_n sebagai bagian S_2 yang baru.
6. Lakukan kembali langkah 4 dan 5 sehingga mendapatkan semua titik terluar yang membentuk *convex hull*.
7. Kembalikan semua pasangan titik terluar yang membentuk *convex hull*.

BAB 2: *Source Program* dalam bahasa Python

0.1. Pustaka *myConvexHull*

```
import math

def CheckLocation(p1, p2, p3):
    """
    Mengembalikan hasil perhitungan determinan dari 3 titik.
    Fungsi memeriksa apakah titik p3 berada di sebelah kiri atau
    sebelah kanan suatu garis yang dibentuk oleh p1 dan p2.
    """
    return p1[0] * p2[1] + p3[0] * p1[1] + p2[0] * p3[1] - p3[0] * p2[1] -
p2[0] * p1[1] - p1[0] * p3[1]

def DivideArea(p1, pn, S, S1, S2):
    """
    I.S.: S1 dan S2 belum terdefinisi
    I.F.: S1 berisi titik yang berada di bagian kiri dari garis yang dibentuk
oleh p1 dan pn. S2 berisi titik yang berada di bagian kanan dari garis yang
dibentuk oleh p1 dan pn.
    """
    for p in S:
        det = CheckLocation(p1, pn, p)
        if det > 0:
            S1.append(p)
        elif det < 0:
            S2.append(p)


def Angle(d1, d2):
    """
    Mengembalikan nilai sudut yang diapit dari garis d1 dan d2.
    """
    if(d1 == 0 or d2 == 0):
        return 0
    else:
        if(d1 > d2):
            return math.acos(d2/d1)
        else:
            return math.acos(d1/d2)


def FurthestDistance(S, p1, pn):
    """
    Mengembalikan titik yang memiliki jarak terjauh dari p1 dan pn.
    """
    maxd1 = math.dist(S[0], p1)
    maxd2 = math.dist(S[0], pn)
    index = 0
    for i in range(1, len(S)):
        d1 = math.dist(S[i], p1)
```

```
        d2 = math.dist(S[i], pn)
        if (d1 + d2 > maxd1 + maxd2):
            index = i
            maxd1 = d1
            maxd2 = d2
        elif (d1 + d2 == maxd1 + maxd2):
            if (Angle(d1, d2) > Angle(maxd1, maxd2)):
                index = i
                maxd1 = d1
                maxd2 = d2
    return S.pop(index)


def ConvexHull(S):
    """
    Fungsi utama pembuatan Convex Hull dengan algoritma Divide and Conquer
    """
    hull = []
    S1 = []
    S2 = []
    S = sorted(S, key=lambda k: [k[0], k[1]])
    p1 = S.pop(0)
    pn = S.pop(-1)
    for p in S:
        det = CheckLocation(p1, pn, p)
        if det > 0:
            S1.append(p)
        elif det < 0:
            S2.append(p)
    FindHull(S1, p1, pn, hull)
    FindHull(S2, pn, p1, hull)
    return hull


def FindHull(S, p1, pn, hull):
    """
    I.S.: S berisi titik yang berada di bagian kiri atau kanan dari garis yang
dibentuk oleh p1 dan pn.
    F.S.: hull berisi titik-titik terluar yang membentuk Convex Hull dari S.
    """
    if(len(S) == 0):
        hull.append([p1, pn])
    else:
        S1 = []
        S2 = []
        pmax = FurthestDistance(S, p1, pn)
        for p in S:
            det1 = CheckLocation(p1, pmax, p)
            if det1 > 0:
                S1.append(p)
            det2 = CheckLocation(pmax, pn, p)
            if det2 > 0:
```

```
                S2.append(p)
        FindHull(S1, p1, pmax, hull)
        FindHull(S2, pmax, pn, hull)
```

BAB 3: Hasil Pengujian

3.1. Data Set Iris (Petal-length vs Petal-width)

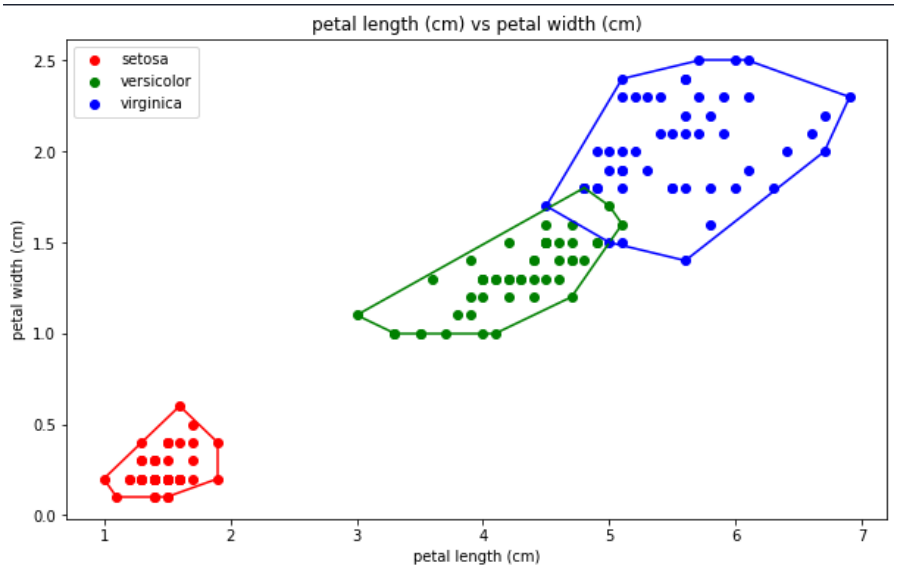
3.1.1. *Input* program

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

data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
colors = ["red","green","blue","pink","black","orange","purple","beige","brown","gray",
"cyan","magenta"]
plt.figure(figsize=(10, 6))
plt.title(data.feature_names[2] + ' vs ' + data.feature_names[3])
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:, [2, 3]].values
  hull = ConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i], color=colors[i])
  for simplex in hull:
    x = [simplex[0][0], simplex[1][0]]
    y = [simplex[0][1], simplex[1][1]]
    plt.plot(x, y, color=colors[i])
plt.legend()
```

Gambar 1. Input program visualisasi data set iris (petal-length vs petal-width)

3.1.2. *Output* program



Gambar 2. Output program visualisasi data set iris (petal-length vs petal-width)

3.2. Data Set Iris (Setal-length vs Setal-width)

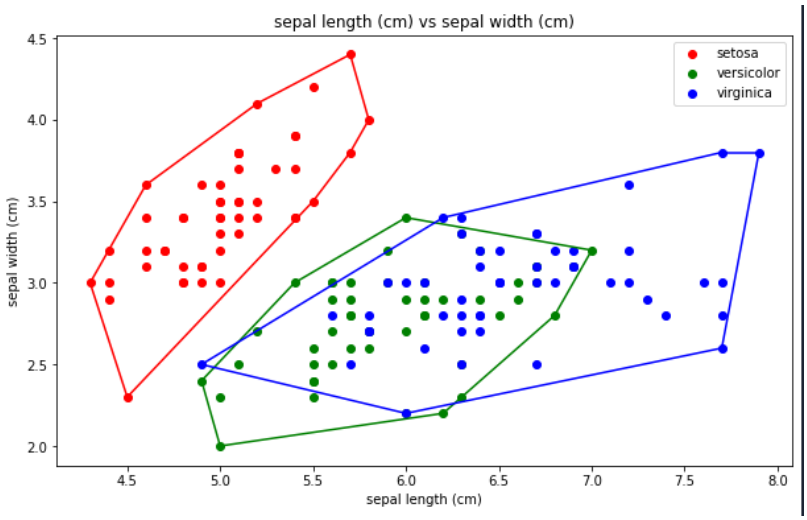
3.2.1. *Input* program

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

data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
colors = ["red","green","blue","pink","black","orange","purple","beige","brown","gray",
"cyan","magenta"]
plt.figure(figsize=(10, 6))
plt.title(data.feature_names[0] + ' vs ' + data.feature_names[1])
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
print(data.target_names)
for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:, [0, 1]].values
  hull = ConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i], color=colors[i])
  for simplex in hull:
    x = [simplex[0][0], simplex[1][0]]
    y = [simplex[0][1], simplex[1][1]]
    plt.plot(x, y, color=colors[i])
plt.legend()
```

Gambar 3. Input program visualisasi data set iris (sepal-length vs sepal-width)

3.2.2. *Output* program



Gambar 4. Output program visualisasi data set iris (sepal-length vs sepal-width)

3.3. Data Set Diabetes (Malic-acid vs Ash)

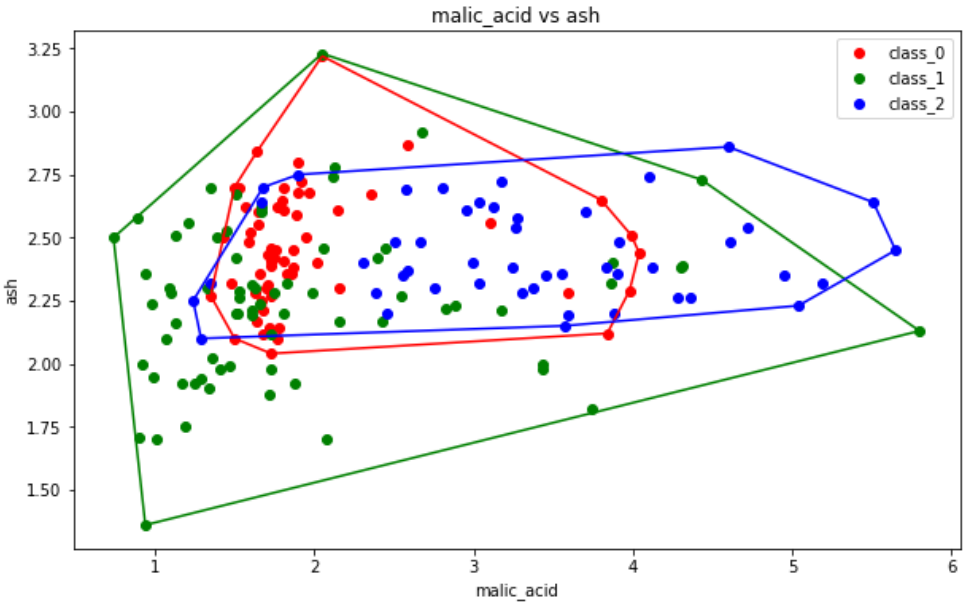
3.3.1. *Input* program

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

data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
colors = ["red","green","blue","pink","black","orange","purple","beige","brown","gray",
"cyan","magenta"]
plt.figure(figsize=(10, 6))
plt.title(data.feature_names[1] + ' vs ' + data.feature_names[2])
plt.xlabel(data.feature_names[1])
plt.ylabel(data.feature_names[2])
for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:, [1, 2]].values
  hull = ConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i], color=colors[i])
  for simplex in hull:
    x = [simplex[0][0], simplex[1][0]]
    y = [simplex[0][1], simplex[1][1]]
    plt.plot(x, y, color=colors[i])
plt.legend()
```

Gambar 5. Input program visualisasi data set diabetes (malic-acid vs ash)

3.3.2. *Output* program



Gambar 6. Output program visualisasi data set diabetes (malic-acid vs ash)

3.4. Data Set Breast Cancer (Mean-radius vs Mean-texture)

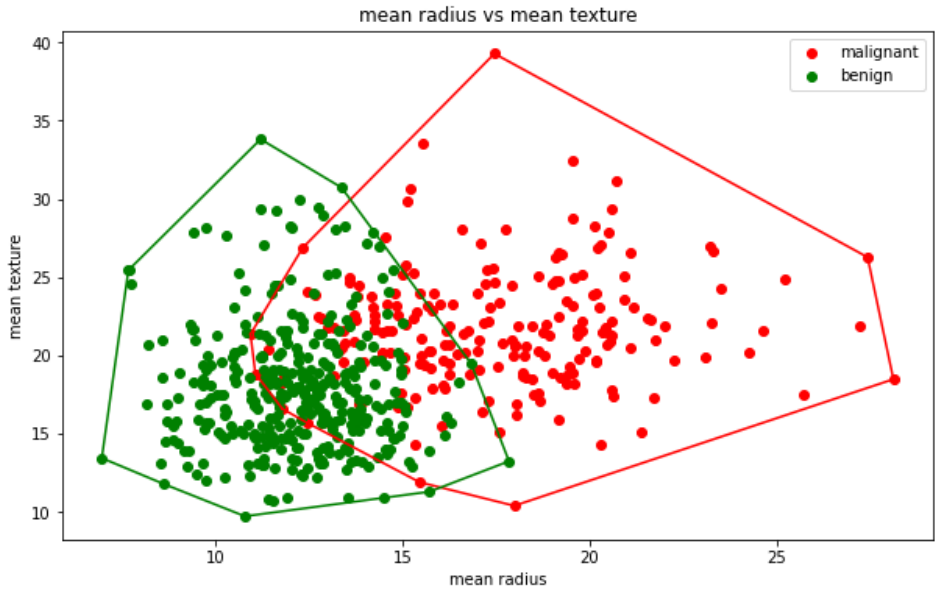
3.4.1. *Input* program

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

data = datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
colors = ["red","green","blue","pink","black","orange","purple","beige","brown","gray",
"cyan","magenta"]
plt.figure(figsize=(10, 6))
plt.title(data.feature_names[0] + ' vs ' + data.feature_names[1])
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 = ConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i], color=colors[i])
  for simplex in hull:
    x = [simplex[0][0], simplex[1][0]]
    y = [simplex[0][1], simplex[1][1]]
    plt.plot(x, y, color=colors[i])
plt.legend()
```

Gambar 7. Input program visualisasi data set breast cancer (mean-radius vs mean-texture)

3.4.2. *Output* program



Gambar 8. Output program visualisasi data set breast cancer (mean-radius vs mean-texture)

3.5. Data Set Digits (Pixel-0-2 vs Pixel-0-3)

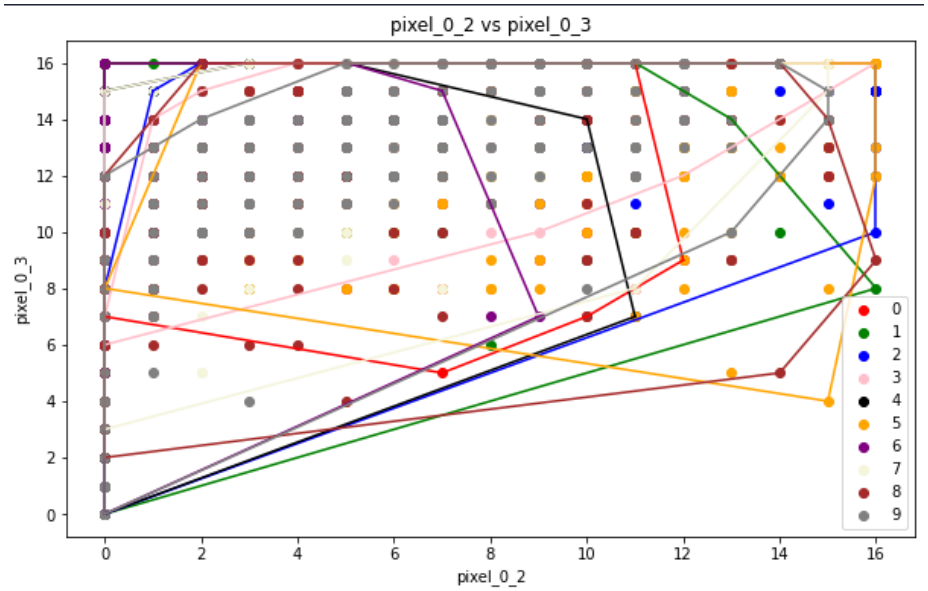
3.5.1. *Input* program

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

data = datasets.load_digits()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
colors = ["red","green","blue","pink","black","orange","purple","beige","brown","gray",
"cyan","magenta"]
plt.figure(figsize=(10, 6))
plt.title(data.feature_names[2] + ' vs ' + data.feature_names[3])
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:, [2, 3]].values
  hull = ConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i], color=colors[i])
  for simplex in hull:
    x = [simplex[0][0], simplex[1][0]]
    y = [simplex[0][1], simplex[1][1]]
    plt.plot(x, y, color=colors[i])
plt.legend()
```

Gambar 9. Input program visualisasi data set digits (pixel_0_2 vs pixel_0_3)

3.5.2. *Output* program



Gambar 10. Output program visualisasi data set digits (pixel_0_2 vs pixel_0_3)

Lampiran

1. Alamat Kode Program (*Github*)

https://github.com/JeremyRio/StimaTucil2

2. Tabel Pengujian

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	√	