

Tugas Kecil 2 IF2211 Strategi Algoritma: Implementasi Convex Hull untuk Visualisasi Tes Linear Separability Dataset dengan Algoritma Divide and Conquer



Dibuat oleh:

Aditya Prawira Nugroho 13520049

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

2022

A. Algoritma

1. Divide and Conquer

Divide and conquer adalah sebuah algoritma yang secara garis besar membagi persoalan besar menjadi persoalan yang lebih kecil. Pada umumnya, algoritma divide and conquer memiliki 3 tahap. Tahap pertama adalah *divide* atau membagi, yaitu persoalan yang besar dibagi menjadi beberapa upa-persoalan yang memiliki kesamaan dengan permasalahan awal, tetapi memiliki ukuran yang lebih kecil. Kemudian, tahap kedua adalah *conquer*, yaitu tahap yang menyelesaikan masing-masing upa-persoalan yang akan langsung diselesaikan jika upa-persoalan merupakan basis dan secara rekursif jika upa-persoalan masih berukuran cukup besar. Tahap terakhir adalah *combine* atau menggabungkan, tahap ini menggabungkan semua solusi yang didapat dari penyelesaian upa-persoalan pada tahap sebelumnya menjadi satu kesatuan yang utuh.

2. Convex Hull dengan Algoritma Divide and Conquer

Untuk mencari *convex hull* dari sebuah himpunan titik, dapat diselesaikan dengan divide and conquer. Ide dasarnya adalah menggunakan algoritma *quicksort*. Kita definisikan S sebagai himpunan titik sebanyak n , dengan $n > 1$, yaitu titik $p_1(x_1, y_1)$ hingga $p_n(x_n, y_n)$ pada bidang kartesian dua dimensi. Kumpulan titik tersebut diurutkan berdasarkan nilai absis atau ordinat yang menaik. Kemudian, titik p_1 dan p_n akan menjadi titik ekstrim yang akan membentuk *convex hull* serta menjadi garis yang membagi kumpulan titik menjadi dua. Kumpulan titik bagian atas bisa membentuk *convex hull* bagian atas dan kumpulan titik bagian bawah bisa membentuk *convex hull* bagian bawah. Setelah itu, tentukan sebuah titik, misal p_{\max} , yang membentuk segitiga dan nilai sudut $p_{\max}p_1p_n$ paling besar di antara kumpulan titik lain. Jika titik p_{\max} sudah ditemukan, ulangi langkah pembagian seperti sebelumnya hingga tidak ada titik lagi. Ketika sudah tidak ada titik yang bisa dicari, kembalikan pasangan titik terakhir.

B. Source Program

1. convexHull.py

```
import myConvexHull.sort as s
import myConvexHull.mathematics as m

# Fungsi yang melakukan merge dua list indeks
def merge(listIndeks1, listIndeks2):
    listIndeksHasil = []
    # Cek apakah listIndeks1 atau 2 kosong
    if listIndeks1 != None:
        for i in range(0, len(listIndeks1)):
            # Cek jika listIndeks1[i] tidak ada duplikat
            if listIndeks1[i] not in listIndeksHasil:
                listIndeksHasil.append(listIndeks1[i])
    if listIndeks2 != None:
```

```

        for i in range (0, len(listIndeks2)):
            if listIndeks2[i] not in listIndeksHasil:
                listIndeksHasil.append(listIndeks2[i])
        return listIndeksHasil

# Fungsi mencari S1 dan S2 dan mengembalikan S1 dan S2 yang berupa list dari
# indeks titik
def findS1AndS2(listTitik, listIndeks, i1, i_n):
    S1 = []
    S2 = []

    for i in listIndeks:
        # Cek apakah indeks dari listIndeks sama dengan indeks maks dan min
        if i != i1 and i != i_n:
            if m.determinan(listTitik[i1][0], listTitik[i1][1],
listTitik[i_n][0], listTitik[i_n][1], listTitik[i][0],
listTitik[i][1]) > 0:
                S1.append(i)
            elif m.determinan(listTitik[i1][0], listTitik[i1][1],
listTitik[i_n][0], listTitik[i_n][1], listTitik[i][0],
listTitik[i][1]) < 0:
                S2.append(i)
    return S1, S2

# Fungsi yang mencari indeks titik terluar bagian atas secara rekursif
def searchHullUp(listTitik, listIndeksUp, i1, i2):
    # basis saat listIndeksUp kosong
    if len(listIndeksUp) < 1:
        hasil = [[i1, i2]]
        return hasil
    else:
        # Cari indeks titik maksimum
        idx_pmax = m.sudutMaks(i1, i2, listTitik, listIndeksUp)
        # Cari S1 dan S2 lagi untuk i1 idx_pmax dan i2 idx_pmax
        S11, unused = findS1AndS2(listTitik, listIndeksUp, i1, idx_pmax)
        S12, unused2 = findS1AndS2(listTitik, listIndeksUp, idx_pmax, i2)
        # bagian bawah tidak terpakai karena dia di dalam bangun

        return merge(searchHullUp(listTitik, S11, i1, idx_pmax),
searchHullUp(listTitik, S12, idx_pmax, i2))

# Fungsi yang mencari indeks titik terluar bagian bawah secara rekursif
def searchHullDown(listTitik, listIndeksDown, i1, i2):
    # basis saat listIndeksDown kosong
    if len(listIndeksDown) < 1:
        return [[i1, i2]]

```

```

else:
    # Cari indeks titik maksimum
    idx_pmax = m.sudutMaks(i1, i2, listTitik, listIndeksDown)
    # Cari S1 dan S2 lagi untuk i1 idx_pmax dan i2 idx_pmax
    unused, S11 = findS1AndS2(listTitik, listIndeksDown, i1, idx_pmax)
    unused2, S12 = findS1AndS2(listTitik, listIndeksDown, idx_pmax, i2)
    # bagian atas tidak terpakai karena dia di dalam bangun

    return merge(searchHullDown(listTitik, S11, i1, idx_pmax),
searchHullDown(listTitik, S12, idx_pmax, i2))

# Fungsi ConvexHull, mengembalikan list Indeks yang sudah berpasangan dan siap
untuk di-plot
def myConvexHull(listTitik):
    # Sorting titik dahulu
    listTitik = s.sortList(listTitik, 0)
    # Buat sebuah list indeks titik urut dari 0 hingga panjang listTitik
    listIndeks = []
    for i in range(len(listTitik)):
        listIndeks.append(i)
    # Simpan indeks awal dan indeks akhir
    iAwal = listIndeks[0]
    iAkhir = listIndeks[-1]
    # Delete indeks 0 dan indeks terakhir agar tidak terjadi pengulangan
    listIndeks.remove(listIndeks[0])
    listIndeks.remove(listIndeks[-1])
    # Cari S1 dan S2
    S1, S2 = findS1AndS2(listTitik, listIndeks, iAwal, iAkhir)
    # Lakukan rekursif
    hull = merge(searchHullUp(listTitik, S1, iAwal, iAkhir),
searchHullDown(listTitik, S2, iAwal, iAkhir))
    return hull

```

2. mathematics.py

```

import math

# Fungsi untuk menghitung determinan dari 3 titik
def determinan(x1, y1, x2, y2, x3, y3):
    # (x1, y1) membentuk garis dengan (x2, y2). (x3, y3) adalah titik maksimum
    baru
    return ((x1 * y2) + (x3 * y1) + (x2 * y3)) - ((x3 * y2) + (x2 * y1) + (x1
* y3))

# Fungsi menghasilkan jarak antar titik
def jarakDuaTitik(x1, y1, x2, y2):

```

```

    return ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5

# Fungsi mencari sudut dari tiga titik, yaitu sudut (x2, y2)
def sudutTigaTitik(x1, y1, x2, y2, x3, y3):
    # Menggunakan rumus vektor
    u = [(x1 - x2), (y1 - y2)]
    v = [(x1 - x3), (y1 - y3)]
    udotv = u[0] * v[0] + u[1] * v[1]
    panjangU = (u[0] ** 2 + u[1] ** 2) ** 0.5
    panjangV = (v[0] ** 2 + v[1] ** 2) ** 0.5
    return math.acos(udotv / (panjangU * panjangV))

# Fungsi yang mencari jarak maksimum, listTitik tidak kosong. Mengembalikan
indeks titik maksimum
def sudutMaks(i1, i_n, listTitik, listIndeks):
    maks = 0
    idx = 0
    for i in listIndeks:
        if (sudutTigaTitik(listTitik[i1][0], listTitik[i1][1],
listTitik[i_n][0], listTitik[i_n][1], listTitik[i][0],
listTitik[i][1])) > maks:
            maks = sudutTigaTitik(listTitik[i1][0], listTitik[i1][1],
listTitik[i_n][0], listTitik[i_n][1],
listTitik[i][0], listTitik[i][1])
            idx = i
    return idx

```

3. sort.py

```

# Fungsi untuk melakukan sorting list titik terurut dari terkecil ke besar
# Akan melakukan sorting berdasarkan Y jika byY = 1, X jika byY = 0
def sortList(listTitik, byY):
    temp = [[0, 0]]
    swapping = True
    # jika byY = 0, maka byX = 1, begitu pula sebaliknya
    byX = 1
    if byY == 1:
        byX = 0
    maxStep = len(listTitik)
    i = 0
    while i < maxStep and swapping:
        swapping = False
        for j in range(0, maxStep - i - 1):
            if listTitik[j][byY] > listTitik[j + 1][byY]:
                temp[0][byY] = listTitik[j][byY]
                temp[0][byX] = listTitik[j][byX]

```

```

        listTitik[j][byY] = listTitik[j + 1][byY]
        listTitik[j][byX] = listTitik[j + 1][byX]

        listTitik[j + 1][byY] = temp[0][byY]
        listTitik[j + 1][byX] = temp[0][byX]
        swapping = True

    return listTitik

```

4. main.py

```

import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import myConvexHull.convexHull as ch
import random

# inisiasi array warna
colors = ['darkviolet', 'cyan', 'yellow', 'lightcoral', 'b', 'r', 'g', 'deeppink',
'purple']
colorPicked = []

# Input dataset
print("List dataset yang tersedia: ")
print("1. iris ")
print("2. wine ")
print("3. breast cancer ")
inputData = int(input("Masukan nomor dataset yang Anda inginkan: "))
# Load dataset berdasarkan masukan pengguna
if inputData == 1:
    data = datasets.load_iris()
    print("Kolom yang ingin divisualisasikan: ")
    print("1. Sepal Width vs Sepal Length")
    print("2. Petal Width vs Petal Length")
    inputTitle = int(input("Masukan nomor pilihan: "))
    if inputTitle == 1:
        awal = 0
        akhir = 1
        title = "Sepal Width vs Sepal Length"
    else:
        awal = 2
        akhir = 3
        title = "Petal Width vs Petal Length"
elif inputData == 2:
    data = datasets.load_wine()
    print("Kolom yang ingin divisualisasikan: ")
    print("1. Alcohol vs Malic Acid")
    print("2. Ash vs Alcalinity of Ash")
    inputTitle = int(input("Masukan nomor pilihan: "))

```

```

if inputTitle == 1:
    awal = 0
    akhir = 1
    title = "Alcohol vs Malic Acid"
else:
    awal = 2
    akhir = 3
    title = "Ash vs Alcalinity of Ash"
else:
    data = datasets.load_breast_cancer()
    print("Kolom yang ingin divisualisasikan: ")
    print("1. Mean Radius vs Mean Texture")
    print("2. Mean Perimeter vs Mean Area")
    inputTitle = int(input("Masukan nomor pilihan: "))
    if inputTitle == 1:
        awal = 0
        akhir = 1
        title = "Mean Radius vs Mean Texture"
    else:
        awal = 2
        akhir = 3
        title = "Mean Perimeter vs Mean Area"

# Buat DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
pd.set_option("max_column", None)

#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
plt.title(title)
plt.xlabel(data.feature_names[awal])
plt.ylabel(data.feature_names[akhir])
for i in range(len(data.target_names)):
    colorNum = random.randint(0, 6)
    # Mencegah warna sama
    while colorNum in colorPicked:
        colorNum = random.randint(0, 6)
    colorPicked.append(colorNum)
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[awal,akhir]].values
    hull = ch.myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i],
color=colors[colorNum])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[colorNum])
plt.legend()
print("Apakah Anda ingin menyimpan diagram?")

```

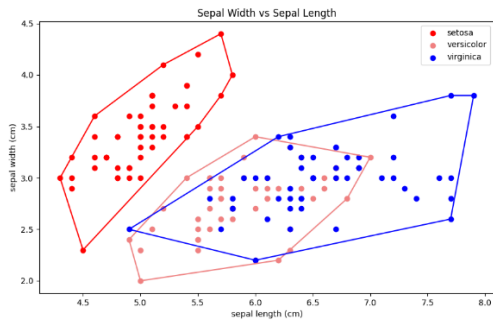

```

save = input("Input Y/N: ")
if (save == "Y" or save == 'y'):
    namaFile = input("Masukan nama file: ")
    plt.savefig("./src/" + namaFile + ".png")
    print("Gambar berhasil disimpan di folder test dengan nama file " +
namaFile + ".png")
plt.show()

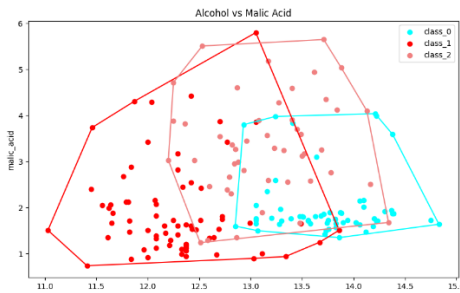
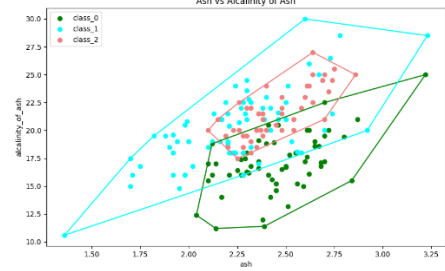
```

C. Input dan Output Program

1. Dataset Iris: Sepal Width vs Sepal Length

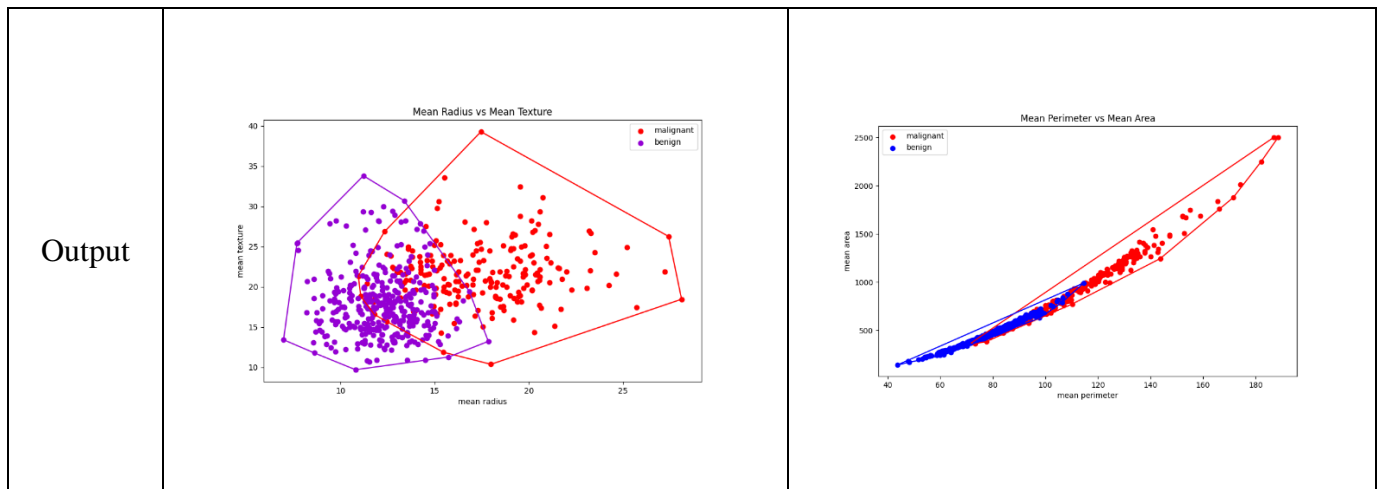
Dataset	Sepal Width vs Sepal Length	Petal Width vs Petal Length
Input	<pre> /Tugas/Tucil 2/Tucil-Stima-2/src/main.py" List dataset yang tersedia: 1. iris 2. wine 3. breast cancer Masukan nomor dataset yang Anda inginkan: 1 Kolom yang ingin divisualisasikan: 1. Sepal Width vs Sepal Length 2. Petal Width vs Petal Length Masukan nomor pilihan: 1 Apakah Anda ingin menyimpan diagram? Input Y/N: N </pre>	<pre> /Tugas/Tucil 2/Tucil-Stima-2/src/main.py" List dataset yang tersedia: 1. iris 2. wine 3. breast cancer Masukan nomor dataset yang Anda inginkan: 1 Kolom yang ingin divisualisasikan: 1. Sepal Width vs Sepal Length 2. Petal Width vs Petal Length Masukan nomor pilihan: 2 Apakah Anda ingin menyimpan diagram? Input Y/N: Y Masukan nama file: iris2 </pre>
Output		

2. Dataset Wine

Dataset	Alvohol vs Malic Acid	Ash vs Alcalinity of Ash
Input	<pre> /Tugas/Tucil 2/Tucil-Stima-2/src/main.py" List dataset yang tersedia: 1. iris 2. wine 3. breast cancer Masukan nomor dataset yang Anda inginkan: 2 Kolom yang ingin divisualisasikan: 1. Alcohol vs Malic Acid 2. Ash vs Alcalinity of Ash Masukan nomor pilihan: 1 Apakah Anda ingin menyimpan diagram? Input Y/N: y Masukan nama file: wine1 </pre>	<pre> /Tugas/Tucil 2/Tucil-Stima-2/src/main.py" List dataset yang tersedia: 1. iris 2. wine 3. breast cancer Masukan nomor dataset yang Anda inginkan: 2 Kolom yang ingin divisualisasikan: 1. Alcohol vs Malic Acid 2. Ash vs Alcalinity of Ash Masukan nomor pilihan: 2 Apakah Anda ingin menyimpan diagram? Input Y/N: y Masukan nama file: wine2 </pre>
Output		

3. Dataset Breast Cancer

Dataset	Alvohol vs Malic Acid	Ash vs Alcalinity of Ash
Input	<pre> /Tugas/Tucil 2/Tucil-Stima-2/src/main.py" List dataset yang tersedia: 1. iris 2. wine 3. breast cancer Masukan nomor dataset yang Anda inginkan: 3 Kolom yang ingin divisualisasikan: 1. Mean Radius vs Mean Texture 2. Mean Perimeter vs Mean Area Masukan nomor pilihan: 1 Apakah Anda ingin menyimpan diagram? Input Y/N: y Masukan nama file: bc1 </pre>	<pre> /Tugas/Tucil 2/Tucil-Stima-2/src/main.py" List dataset yang tersedia: 1. iris 2. wine 3. breast cancer Masukan nomor dataset yang Anda inginkan: 3 Kolom yang ingin divisualisasikan: 1. Mean Radius vs Mean Texture 2. Mean Perimeter vs Mean Area Masukan nomor pilihan: 2 Apakah Anda ingin menyimpan diagram? Input Y/N: y Masukan nama file: bc2 </pre>



D. Alamat Kode Program

Program dapat diunduh dari alamat berikut:

<https://drive.google.com/drive/folders/1z92jb5-vyeDSyHNI62aJERA8zencZv5?usp=sharing>

atau

<https://github.com/Adityapnn811/Tucil-Stima-2>

E. Tabel Penilaian

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