Tugas Kecil 2 IF2211 Strategi Algoritma

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

Dibuat oleh:

Shadiq Harwiz 13520038

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

A. Algoritma *Divide and Conquer*

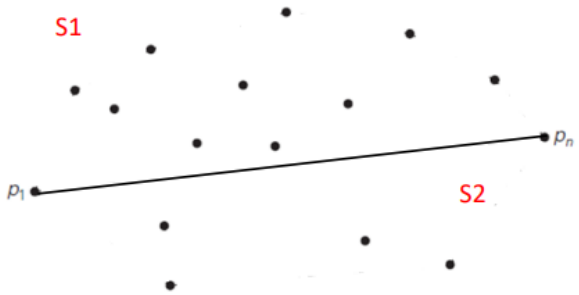
Divide and conquer adalah sebuah desain algoritma di dalam ilmu komputer. Dalam sebuah permasalahan, agoritma ini bekerja secara rekursif untuk membagi permasalahan tersebut menjadi sub-sub permasalahan sampai sub permasalahan ini dapat diselesaikan secara langsung. Solusi dari permasalahan kecil ini kemudian dikombinasikan dengan solusi sub permasalahan lain untuk menyelesaikan permasalahan yang lebih besar sampai keseluruhan permasalahan dapat diselesaikan.

Convex hull sebuah himpunan titik pada bidang planar adalah himpunan titik-titik yang telah membentuk sebuah poligon konveks yang melingkupi seluruh himpunan titik tersebut. Sebuah subset S dari bidang $\boldsymbol{R}$ disebut konveks jika dan hanya jika pada seluruh dua buah titik sembarang $p, q \in S$ dibentuk garis yang seluruhnya berada dalam S. Pencarian *convex* hull dari sebuah himpunan titik Q adalah mencari sebuah convex set terkecil yang memuaat seluruh titik pada Q. *Convex* hull dari sebuah himpunan titik Q pada n dimensi adalah seluruh irisan dari semua *convex set* yang mengandung Q. Terlebih lanjut, untuk N buah titik $p_1, p_2, \dots, p_N$. *Convex hull* merupakan himpunan *convex combination*. Teknik *divide and conquer* menjadi salah satu cara untuk menyelesaikan pencarian *convex hull*.

Gambar 1 Convex Set

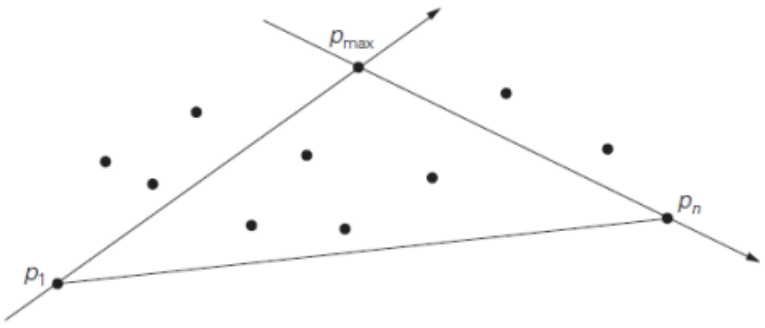
Langkah-langkah yang dilakukan dalam menyelesaikan persoalan *convex hull* dengan memanfaatkan strategi algorima *divide and conquer* adalah sebagai berikut.

1. Kumpulan titik sebanyak *n* dari himpunan *S* yang telah diterima terlebih dahulu diurutkan berdasarkan nilai absis yang menaik pada masing-masing titik.
2. Ambil titik dengan urutan pertama sebagai p_1 dan titik dengan urutan terakhir sebagai p_n.
3. Hubungkan kedua titik p_1 dan p_n menjadi sebuah garis p_1p_n sehingga akan membagi *S* menjadi dua bagian, yaitu S_1 (kumpulan titik yang berada di kiri garis p_1p_n) dan S_2 (kumpulan titik yang berada di kiri garis p_np_1)



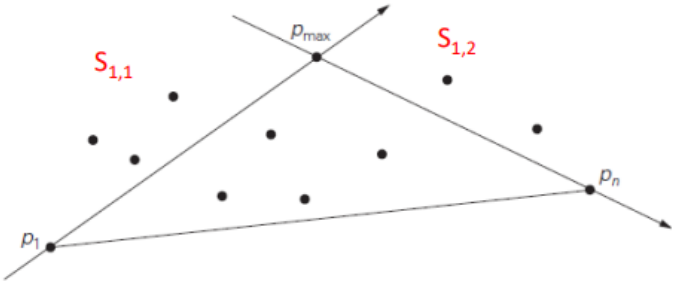
Gambar 2 ilustrasi pada nomor 3

4. Lakukan pemeriksaan apakah sebuah titik pada S berada di kiri atau di kanan garis p_1p_n, untuk melakukan pemeriksaan tersebut dapat memanfaatkan konsep determinan yang dikemas dalam bentuk persamaan $(y_2 - y_1)x + (x_1 - x_2)y + (x_2 \times y_1 - x_1 \times y_2)$, dengan terdapat sebuah titik (x, y) yang akan dicari untuk mengetahu apakah titik tersebut berada di kiri atau di kanan sebuah garis $\big((x_1, y_1), (x_2, y_2)\big)$. Jika bernilai positif, titik tersebut berada di sebelah kanan garis $\big((x_1, y_1), (x_2, y_2)\big)$ yang dikumpulkan menjadi bagian S_1 dan jika bernilai negatif, titik tersebut berada di sebelah kiri garis $\big((x_1, y_1), (x_2, y_2)\big)$ yang dikumpulkan menjadi bagian S_2.
5. Apabila tidak ada titik yang berada di sebelah kiri garis p_1p_n, maka titik p_1 dan p_n menjadi bagian dari $convex\ hull$. Sebaliknya, akan dicari satu buah titik diantara titik-titik yang berada di sebelah kiri garis p_1p_n yang memiliki jarak tegak lurus terjauh terhadap garis p_1p_n sebagai titik p_2. Apabila terdapat beberapa titik yang sama, pilih titik yang pertama kali telah ditemukan sebagai titik p_2. Akibatnya, akan terbentuk sebuah daerah segitiga $p_1p_np_2$



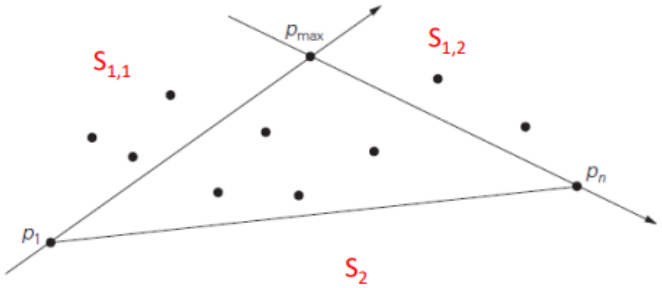
Gambar 3 ilustrasi pada nomor 5

6. Lakukan pemeriksaan dan kumpulkan titik-titik yang berada di sebelah kiri garis p_1p_2 menjadi bagian $S_{1,1}$ dan di sebelah kiri garis p_2p_n menjadi bagian $S_{1,2}$.



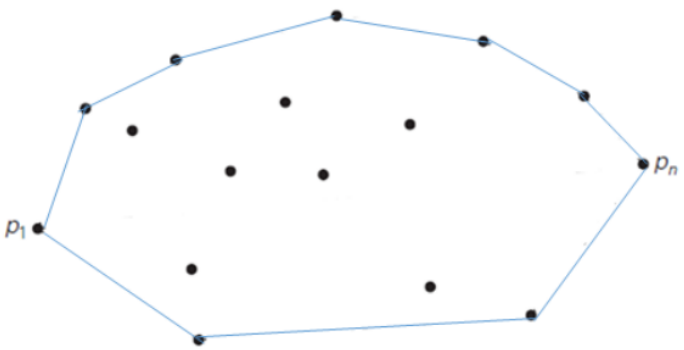
Gambar 4 ilustrasi pada nomor 6

7. Lakukan hal yang sama pada butir nomor 5 dan 6 untuk $S_{1,1}, S_{1,2}$, dan S_2 hingga tidak ada lagi yang dapat dilakukan untuk pemeriksaan dan pengumpulan titik-titik S.



Gambar 5 ilustrasi pada nomor 7

8. Kembalikan titik-titik yang dihasilkan.



Gambar 6 kumpulan titik-titik yang dihasilkan yang telah saling dipasangkan sebagai garis

B. Source Program

Dalam mengimplementasi Convex Hull untuk Visualisasi Tes Linear Separability Dataset dengan memanfaatkan strategi algoritma *divide and conquer*, penulis membuat program dalam bahasa python.

1. MyConvexHull.py

```
import math

#Fungsi untuk melakukan sorting
def sort_key(koordinat):
    return koordinat[0]

#Fungsi menghitung jarak tegak lurus antara titik dan garis
def jarak_tegaklurus(A,B,x,y) :
    x1, y1 = A
    x2, y2 = B
    a = y2 - y1
    b = x1 - x2
    c = x2*y1 - x1*y2
    d = (abs((a*x + b*y + c))) / (math.sqrt(a*a + b*b))
    return(d)

#Fungsi untuk menentukan dimana posisi suatu titik berada terhadap garis AB
def status_sebuah_titik(A,B,koordinasi):
    x1, y1 = A
    x2, y2 = B
    x, y = koordinasi
    a = y2 - y1
    b = x1 - x2
    c = x2*y1 - x1*y2
    f = a*x + b*y + c
    if f < 0:
        return 'kiri'
    elif f > 0:
        return 'kanan'
    else:
        return 'satu-garis'

def menemukan_banyak_titik_terluar(partisi,A,B,solusi_titik):
    if len(partisi) == 0:
        if ( ( A in solusi_titik) == False ) :
            solusi_titik.append(A)
        if ( ( B in solusi_titik) == False ) :
            solusi_titik.append(B)
        return
    else : #mencari titik yang berada paling luar ( panjang tegak lurus terbesar
terhadap garis AB )
        jarak_terjauh = -1
        C = None
        for koordinasi in partisi:
            x, y = koordinasi
            f = jarak_tegaklurus(A,B,x,y)
            if f > jarak_terjauh:
                jarak_terjauh = f
                C = koordinasi
        x,y = C
```

```
        #Hapus titik C pada partisi
        partisi.remove(C)

        #Menyeleksi titik-titik yang tersedia untuk disimpan pada tempatnya
        ACkiri = []
        for koordinasi in partisi:
            koordinasi_side = status_sebuah_titik(A,C,koordinasi)
            if koordinasi_side == 'kiri':
                ACkiri.append(koordinasi)
        CBkiri = []
        for koordinasi in partisi:
            koordinasi_side = status_sebuah_titik(C,B,koordinasi)
            if koordinasi_side == 'kiri':
                CBkiri.append(koordinasi)


        #Penerapan Divide and Conquer
        menemukan_banyak_titik_terluar(ACkiri,A,C,solusi_titik)
        menemukan_banyak_titik_terluar(CBkiri,C,B,solusi_titik)

def convex_hull(bucket):
    # buat array solusi
    solusi_titik = []

    # Ubah type dri numarr ke list
    koordinat = bucket.tolist()

    #Lakukan pengurutan titik berdasarkan koordinat X dari mengecil ke membesar
    koordinat.sort(key=sort_key, reverse=False)

    #Ambil titik koordinat pada elemen array pertama dan terakhir untuk
membentuk garis AB
    A = koordinat[0]
    B = koordinat[-1]

    #Inisiasi array untuk bagian kumpulan titik-titik yang berada bada sisi kiri
AB atau kiri BA
    ABkiri = []
    BAkiri = []

    #Menyeleksi titik-titik yang tersedia untuk disimpan pada tempatnya
    for koordinasi in koordinat:
        koordinasi_side = status_sebuah_titik(A,B,koordinasi)
        if koordinasi_side == 'kiri':
            ABkiri.append(koordinasi)
        elif koordinasi_side == 'kanan':
            BAkiri.append(koordinasi)
        else:
            pass

    #Penerapan Divide and Conquer
    menemukan_banyak_titik_terluar(ABkiri,A,B,solusi_titik)
    menemukan_banyak_titik_terluar(BAkiri,B,A,solusi_titik)

    #return(solusi_garis)
    return(solusi_titik)
```

2. Main.py

```
while (True) :
    #Meminta input dari user terkait dataset yang akan diolah
    data = None
    print("Kumpulan dataset : ")
    print("1. Breast Cancer")
    print("2. Iris")
    print("3. Wine")
    correct_input_j  = False
    while ( correct_input_j == False ) :
        j = int(input("Silakan pilih dataset yang akan digunakan (0 Jika ingin
keluar) >> "))
        if ( (j < 0 or j > 3) == False ) :
            correct_input_j = True

    if ( j == 1 ) :
        data = datasets.load_breast_cancer()
    elif ( j == 2) :
        data = datasets.load_iris()
    elif ( j == 3 ) :
        data = datasets.load_wine()
    else :
        exit()

    length = len(data.feature_names)
    for i in range(length):
        print(i+1, '.', data.feature_names[i])

    x = None
    y = None
    correct_input_x = False
    correct_input_y = False

    while ( correct_input_x == False ) :
        x = int(input("Silakan pilih informasi atribut yang akan digunakan
sebagai sumbu X (0 Jika
        if ( (x < 0 or x > length) == False ) :
            correct_input_x = True

    while ( correct_input_y == False ) :
        y = int(input("Silakan pilih informasi atribut yang akan digunakan
sebagai sumbu Y (0 Jika
        if ( (y < 0 or y > length) == False ) :
            if ( y == x ) :
                correct_input_opsi = False
                while ( correct_input_opsi == False ) :
                    print("Atribut yang dipilih telah dipilih sebelumnya sebagai
sumbu X, tetap in
                    opsi = str(input(">> "))
                    if ( opsi == "Y" ) :
                        correct_input_y = True
                        correct_input_opsi = True
                    elif ( opsi == "N" ) :
                        correct_input_y = False
                        correct_input_opsi = True
                    else :
                        correct_input_opsi = False
            else :
                correct_input_y = True
```

```
    #create a DataFrame
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
    print("Berikut isi dari atribut informasi yang telah dipilih")
    print(df.iloc[0:, [x-1,y-1]])

    #visualisasi hasil ConvexHull
    plt.figure(figsize = (10, 6))
    colors = ['b','r','g','c','m','y','k']
    plt.title(f'{data.feature_names[x-1]} vs {data.feature_names[y-1]}')
    plt.xlabel(data.feature_names[x-1])
    plt.ylabel(data.feature_names[y-1])
    for i in range(len(data.target_names)):
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:,[x-1,y-1]].values

        #gunakan library yg telah dibuat
        test1 = MCH.convex_hull(bucket)

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

        # Mengolompokkan titik menjadi per titik pada sumbu x dan sumbu y
        x_values = [float(aw[0]) for aw in test1]
        y_values = [float(aw[1]) for aw in test1]
        x_values.append(test1[0][0])
        y_values.append(test1[0][1])

        # Membuat garis
        plt.plot(x_values, y_values, colors[i])

    plt.legend()
    plt.show()

    correct_input_quit = False
    while ( not correct_input_quit ) :
        print("Ingin tetap melanjutkan? (Y/N)")
        quit = str(input(">> "))
        if ( quit == 'N' ) :
            exit()
        elif ( quit == 'Y' ) :
            correct_input_quit = True
        else :
            correct_input_quit = False
```

C. Alamat Kode Program

Program dapat diunduh dari alamat berikut:

https://drive.google.com/drive/folders/1R3NU-4R0aAFGzqbXvf5gf6UzQK409GI2?usp=sharing

Atau dapat melalui

https://github.com/shdiqq/Tucil2_13530038.git

D. Tabel Penilaian

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	☒	☐