**LAPORAN TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA**

**IMPLEMENTAI CONVEX HULL UNTUK VISUALISASI TES LINEAR SEPARABILITY DATASET DENGAN ALGORITMA DIVIDE AND CONQUER**

Logo

Description automatically generated

Oleh :

Ilham Prasetyo Wibowo 13520013

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2022**

1. DESKRIPSI ALGORITMA

Himpunan titik pada sebuah bidang planar disebut convex jika untuk sembarang dua titik pada bidang tersebut semua segmen garis yang berakhir di titik tersebut berada pada himpunan tersebut. Convex hull dari sekumpulan titik S adalah sebuah himpunan terkecil (convex terkecil) yang mengandung semua titik pada S.

Program menerima sebuah larik (array) yang berisi titik-titik dalam ruang dua dimensi yang akan dicari convex hull-nya. Kemudian akan dicari garis yang membagi dua dari kumpulan titik. Garis tersebut dibentuk dari titik yang paling kiri (nilai x terkecil) dengan garis yang paling kanan(nilai x terbesar).

Chart, scatter chart

Description automatically generated

Ilustrasi mencari garis yang akan membagi titik menjadi dua bagian

Setelah garis ditemukan, kumpulan titik akan dibagi menjadi dua yaitu titik yang berada di atas garis, dan titik yang berada dibawah garis. Setelah itu akan dicari masing masing convex hull dari bagian dan akan digabungkan.

Pencarian convex hull dilakukan secara rekursif. Basis rekursif adalah jika tidak ada lagi titik di atas garis. Atau titik di atas garis hanya tersisa satu.

Chart, scatter chart

Description automatically generated

Ilustrasi ketika sudah mencapai basis rekursi pertama

Chart, scatter chart

Description automatically generated

Ilustrasi Ketika mencapai basis rekursi kedua

Ketika sudah mencapai basis, program akan mengembalikan sebuah tuple dengan dua elemen yaitu index titik yang membentuk garis . Ketika program mencapai basis rekursi kedua, program akan mengembalikan kedua garis misal AB, dan BC.

Jika masih terdapat lebih dari satu titik yang ada diatas garis, program akan mencari titik yang paling jauh dari garis. Misal garis AB, dan terdapat titik terjauh X. Program akan mencari titik titik yang berada di atas garis AX dan XB. kemudian secara rekursif akan mencari convex hull masing masing titik dan kedua hasil akan digabung.

1. SOURCE CODE

Berikut kode program yang telah ditulis.

#file myConvexHull.py

import numpy as np

from numpy.linalg import norm

def myConvexHull(points):

    #get index of max value in points

    max\_idx = np.argmax(points, axis=0)[0]

    #get index of min value in points

    min\_idx = np.argmin(points, axis=0)[0]

#divide points into two part with a line shaped by point in min\_idx and max\_idx

    #get index of values above the line

    index\_above = partition\_above(points,points[min\_idx],points[max\_idx])

    #get index of values above the line

    index\_below = partition\_above(points,points[max\_idx],points[min\_idx])

    #search convex hull above line

    above\_convex = get\_convex\_hull(points,index\_above,points[min\_idx],

points[max\_idx],[min\_idx,max\_idx])

    #search convex hull below line

    below\_convex = get\_convex\_hull(points,index\_below,points[max\_idx],

points[min\_idx],[max\_idx,min\_idx])

    #combine both convex hulls

    trueConvex = above\_convex + below\_convex

    return np.array(trueConvex)

def get\_convex\_hull(points,index\_A,pA,pB,pLine):

    #there isnt any points on left/right side of pLine

    if (len(index\_A) == 0):

        return [pLine]

    #only one point on left/right side of pLine

    #return line ab, bc

    elif (len(index\_A) == 1) :

        return [[pLine[0],index\_A[0]],[index\_A[0],pLine[1]]]

    #still more than one points

    #search for a points farthest from line pApB

    else :

        farthest\_idx = get\_farthest\_point(points,pA,pB,index\_A)

        farthest = points[farthest\_idx]

    #index of points on left side of line pA-farthestpoint

    convex\_left = partition\_above\_2(points,pA,farthest,index\_A)

    #index of points on left side of line farthestpoint,pB

    convex\_right = partition\_above\_2(points,farthest,pB,index\_A)

    #search convex hull left side of line pA-farthest recursively

    left\_hull = get\_convex\_hull(points,convex\_left,pA,

farthest,[pLine[0],farthest\_idx])

    #search convex hull left side of line farthest-pB recursively

    right\_hull = get\_convex\_hull(points,convex\_right,farthest,pB,

[farthest\_idx,pLine[1]])

    #combine the results from left\_hull and right\_hull

    ultimate\_hull = left\_hull + right\_hull

    #return

    return ultimate\_hull

#search points by ALL points in point

def partition\_above(points,pA,pB):

    points\_result = []

    for i in range(len(points)):

        #Determinant

        x = pA[0]\*pB[1] + points[i][0]\*pA[1] + pB[0]\*points[i][1] –

points[i][0]\*pB[1] - pB[0]\*pA[1] - pA[0]\*points[i][1]

        if (x > 0):

            #point[i] is above line pApB

            points\_result.append(i)

    #returns indices of points in list

    return points\_result

#search points only by array of index ind

def partition\_above\_2(points,pA,pB,ind):

    points\_result = []

    for i in ind:

        if (points\_equal(points[i],pA)):

            continue

        elif (points\_equal(points[i],pB)):

            continue

        else :

            #determine whether the point is above or below line pA-pB

            x = (pA[0]\*pB[1]) + (points[i][0]\*pA[1]) +

(pB[0]\*points[i][1]) - (points[i][0]\*pB[1]) - (pB[0]\*pA[1]) - (pA[0]\*points[i][1])

            if (x > 0):

                #point[i] is above line pApB

                points\_result.append(i)

    #returns indices of points in list

    return points\_result

def points\_equal(pA,pB):

    if (pA[0] == pB[0]):

        if (pA[1] == pB[1]):

            return True

    return False

#get the farthest point from line pA-pB

def get\_farthest\_point(points,pA,pB,idx):

    ires = 0

    dres = 0

    for i in idx:

        d = abs(np.cross(pB-pA,points[i]-pA)/norm(pB-pA))

        if (dres < d):

            ires = i

            dres = d

    return ires

1. HASIL PENGUJIAN

**IRIS DATASET**

Chart, scatter chart

Description automatically generated

Chart

Description automatically generated

**WINE DATASET**

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated

**BREAST CANCER DATASET**

Chart, scatter chart

Description automatically generated

Chart, line chart

Description automatically generated

1. TAUTAN DRIVE PROGRAM

<https://github.com/ilhamwibowo/Tucil-2-STIMA>

Table

Description automatically generated