Laporan Tugas Kecil 2

IF2211 Strategi Algoritma

Algoritma Divide and Conquer dengan Aplikasinya

Logo

Description automatically generated

Disusun oleh :

Bariza Haqi 13520018

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

1. Langkah Algoritma *Divide and Conquer*

Untuk langkah awalnya pilih dataset yang akan dicari *convex hull*-nya.

Berikut langkah-langkah algoritma *divide and conquer* pada *convex hull*:

1. Ambil 2 titik terjauh dari dataset yaitu p1 dengan nilai x terkecil dan p2 dengan nilai x terbesar
2. Jika kedua titik tersebut memiliki nilai x yang sama, maka ambil titik y terkecil dan y terbesar. Kedua titik tersebut merupakan hasil *convex hull-*nya
3. Jika kedua titik memiliki nilai x yang berbeda, bagi menjadi 2 bagian list titik dengan batas garis yang dibuat dari titik p1 dan p2
4. Pada list titik bagian 1, jika kosong maka p1 dan p2 akan membentuk garis *convex hull*.
5. Jika list titik tidak kosong maka ambil titik terjauh yaitu p3 dari garis p1 dan p2. Jika terdapat 2 titik dengan jarak sama maka ambil titik yang memaksimalkan sudut p3p1p2 dengan melihat nilai cos terkecil.
6. Buat sebuah list yang berisi titik yang memiliki determinan positif dari list titik dengan titik p1 dan p3 untuk mencari titik diluar segitiga p1p2p3 dari garis p1 dan p3. Jika list tersebut tidak kosong, ulangi Langkah ke-5 untuk list tersebut sampai kosong.
7. Lakukan Langkah ke-6 secara bersamaan untuk list titik dengan titik p3 dan p2. Gabungkan hasil semua titik terjauh yang didapat sehingga tersusun searah jarum jam
8. Ulangi Langkah ke-4 untuk list titik bagian 2
9. Gabungkan hasil titik pada list bagian 1 dan 2
10. Source Code Algoritma *Divide and Conquer*

Untuk penggunaan algoritma *divide and conquer* sendiri memakai bahasa Python. Berikut adalah source code program untuk membuat *convex hull* dengan algoritma

*divide and conquer.*

import numpy as np

import math

def getDetThree(p1,p2,p3):

    #mengembalikan determinan dari 3 titik

    det=round(p1[0]\*p2[1] + p3[0]\*p1[1] + p2[0]\*p3[1] - p3[0]\*p2[1] - p2[0]\*p1[1] - p1[0]\*p3[1],5) ##menggunakan round karena python membaca nilai det mendekati 0 saat hasil seharusnya 0 untuk titik yang segaris

    return det

def getJarak(p1,p2,p3):

    #mengembalikan jarak p3 ke garis p1 dan p2

    #x1 tidak boleh sama dengan x2

    m=(p2[1]-p1[1])/(p2[0]-p1[0])

    jarak=abs((-m\*p3[0]+1\*p3[1]+(m\*p1[0]-p1[1]))/math.sqrt(1+m\*\*2))

    return jarak

def getCosSudut(p1,p2,p3):

    #mengembalikan cos sudut dari p3p1p2

    #titik p1!=p2 dan titik p1!=p3

    a=math.sqrt((p3[0]-p2[0])\*\*2+(p3[1]-p2[1])\*\*2)

    b=math.sqrt((p3[0]-p1[0])\*\*2+(p3[1]-p1[1])\*\*2)

    c=math.sqrt((p2[0]-p1[0])\*\*2+(p2[1]-p1[1])\*\*2)

    cos=(b\*\*2+c\*\*2-a\*\*2)/(2\*b\*c)

    return cos

def getListDet(S,p1,p2):

    #mengembalikan list titik dengan determinan positif dari garis p1 dan p2

    list=[]

    for i in range(len(S)):

        det=getDetThree(p1,p2,S[i])

        if (det>0):

            list.append(S[i])

    return list

def getListConvex(ListS,p1,p2):

    #mengembalikan list point yang membentuk convex

    #ListS tidak boleh kosong

    list=[]

    list1=[]

    list2=[]

    list3=[]

    #cari pmax yaitu titik terpanjang dari garis p1 dan p2

    jarak=getJarak(p1,p2,ListS[0])

    indexPoint=0

    for i in range (1,len(ListS)):

        jarakbaru=getJarak(p1,p2,ListS[i])

        if (jarakbaru>jarak): #pilih titik dengan jarak terpanjang dari garis

            jarak=jarakbaru

            indexPoint=i

        elif (jarakbaru==jarak): # pilih titik yang membentuk sudut p1 terbesar dilihat dari nilai cos sudut terkecil

            cossudutjarak=getCosSudut(p1,p2,ListS[indexPoint])

            cossudutjarakbaru=getCosSudut(p1,p2,ListS[i])

            if (cossudutjarakbaru<cossudutjarak):

                jarak=jarakbaru

                indexPoint=i

    pmax=ListS[indexPoint]

    #cari Listconvex untuk garis p1pmax dan pmaxp2

    ListBagian1=getListDet(ListS,p1,pmax)

    print(pmax)

    print(ListBagian1)

    if (len(ListBagian1)!=0):

        list1=getListConvex(ListBagian1,p1,pmax)

    list2.append(pmax)

    ListBagian2=getListDet(ListS,pmax,p2)

    if (len(ListBagian2)!=0):

        list3=getListConvex(ListBagian2,pmax,p2)

    list=list1+list2+list3 #susun list sehingga memutar searah jarum jam

    return list

def ConvexHull(T):

    S=T.tolist() #ubah T dari numpy array menjadi list

    List=[]

    p1=S[0]

    p2=S[0]

    for i in range(1,len(S)):

        if (S[i][0]>p2[0]):

            p2=S[i]

        if (S[i][0]<p1[0]):

            p1=S[i]

    if (p2[0]==p1[0]):#jika x2-x1=0

        #ambil 2 titik dengan jarak y terpanjang

        minY=S[0]

        maxY=S[0]

        if (len(S)>1):

            for i in range(1,len(S)):

                if (S[i][1]>p2[1]):

                    maxY=S[i]

                if (S[i][1]<p1[1]):

                    minY=S[i]

            List.append(minY)

            List.append(maxY)

    else:#x2-x1!=0

        #bagi menjadi 2 bagian dari garis p1 dan p2

        ListS1=getListDet(S,p1,p2)

        ListS2=getListDet(S,p2,p1)

#susun list sehingga memutar searah jarum jam

        if (len(ListS1)!=0 and len(ListS2)!=0):

            ListConvexS1=getListConvex(ListS1,p1,p2)

            ListConvexS2=getListConvex(ListS2,p2,p1)

            List.append(p1)

            List+=ListConvexS1

            List.append(p2)

            List+=ListConvexS2

        elif (len(ListS2)!=0):

            ListConvexS2=getListConvex(ListS2,p1,p2)

            List.append(p1)

            List.append(p2)

            List+=ListConvexS2

        elif (len(ListS1)!=0):

            ListConvexS1=getListConvex(ListS1,p1,p2)

            List.append(p1)

            List+=ListConvexS1

            List.append(p2)

        else: #ListS1 kosong dan ListS2

            List.append(p1)

            List.append(p2)

    List=listToNpArray(List)

    return List

def listToNpArray(List):

    #membuat array menjadi susunan [[x1,x2],[y1,y2]] dan berbentuk numpy array

    list=[]

    ListX=[]

    ListY=[]

    for i in range (len(List)):

        if (i<len(List)-1):

            X=[List[i][0],List[i+1][0]]

            Y=[List[i][1],List[i+1][1]]

            ListX.append(X)

            ListY.append(Y)

        else: #i=index terakhir list, buat garis dari titik index terakhir dengan titik index pertama

            X=[List[i][0],List[0][0]]

            Y=[List[i][1],List[0][1]]

            ListX.append(X)

            ListY.append(Y)

    list.append(ListX)

    list.append(ListY)

    List=np.array(list)

    return List

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

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 = ConvexHull(bucket)

    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target\_names[i])

    for index in range(len(hull[0])):

        plt.plot(hull[0][index], hull[1][index], colors[i])

plt.legend()

1. Screenshot Input dan Output Program
2. Dataset iris dengan atribut sepal width dan sepal length

Text

Description automatically generated with medium confidence

1. Dataset iris dengan atribut petal width dan petal length

Graphical user interface, text

Description automatically generated

1. Dataset wine dengan atribut alcohol dan malic acid

Graphical user interface

Description automatically generated with medium confidence

1. Dataset breast\_cancer dengan atribut mean radius dan mean texture

Graphical user interface

Description automatically generated with low confidence

1. Alamat drive

https://github.com/barizahaqi/Tucil-2-Strategi-Algoritma

|  |  |  |
| --- | --- | --- |
| 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 | ✓ |  |