**LAPORAN TUGAS KECIL IF2211 STRATEGI ALGORITMA SEMESTER II TAHUN 2021/2022**

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

Chart, scatter chart

Description automatically generated

**Disusun oleh:**

**13520156 Dimas Faidh Muzaki**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

# Algoritma *Divide and Conquer*

*Convex Hull* yang diimplementasikan di pustaka saya merupakan sebuah objek dari kelas bernama CHull yang memiliki dua data member yaitu *points* dan *simplices*. Dalam pembentukan objek kelas (konstruktor), kelas menerima sebuah argumen berupa matriks berukuran n x 2. Matriks tersebut merupakan dataset yang ingin kita uji *linear separability* nya. Setiap elemen pada matriks memiliki 2 elemen yang dapat direpresentasikan sebagai titik 2 dimensi. Data member *points* merupakan Salinan dari matriks masukan yang setiap elemennya dinomori sebagai identitas. Sementara itu, *simplices* merupakan sebuah list yang isinya adalah pasangan-pasangan nomor elemen points yang membentuk *Convex Hull.* Dalam mencari *Convex Hull*, pustaka saya memiliki algoritma *Divide and Conquer* sebagai berikut:

1. Data yang dimasukkan sebagai argument konstruktor elemennya masing-masing akan diberikan nomor identitas. Elemen data selanjutnya akan disebut sebagai titik
2. Titik yang sudah dinomori akan diurutkan. Titik memiliki atribut absis dan ordinat. Pengurutan dilakukan dengan abis menaik titik terlebih dahulu, untuk nilai absis yang sama akan diurutkan berdasarkan ordinat.
3. Dari titik-titik terurut tersebut akan diambil 2 titik ekstrem, indeks pertama dan terakhir, untuk dijadikan dua titik awal proses *divide and conquer.*
4. Selanjutnya adalah membagi *points* menjadi 2 himpunan. Himpunan pertama, s1, adalah titik-titik yang terletak disebelah kiri garis yang menghubungkan dua titik ekstrem. Sementara himpunan kedua, s2, adalah titik-titik yang terletak di sebelah kanan garis yang menghubungkan dua titik ekstrem tadi. Hal ini dilakukan dengan dibantu fungsi determinan pada file utils.
5. S1 dan s2 masing masing akan dimasukkan ke fungsi rekursif untuk mencari *simplices.* Mereka akan dimasukkan ke fungsi bersama dua titik ekstrem sebagai p1 dan p2
6. Di dalam fungsi ada dua kemungkinan kondisi:
7. Apabila himpunan titik yang masuk kosong. Maka p1 dan p2 adalah pembetuk *Convex Hull* dan nomor-nomor nya dimasukkan ke dalam *simplices* sebagai sebuah pasangan
8. Apabila himpunan tidak kosong, akan dipilih sebuah titik pn yang memiliki jarak terjauh dari garis p1p2. Jika terdapat beberapa titik dengan jarak yang sama, akan dipilih titik yang memaksimal sudut p1pnp2. Hal ini dapat dicapai dengan menggunakan fungsi farthest\_from\_line dan point\_distance\_to\_line
9. Selanjutnya akan dicari kumpulan titik yang berada di sebelah kiri garis p1pn dan di sebelah kanan garis pnp2
10. Untuk kedua kumpulan titik tadi akan dilakukan proses nomor 6 dan 7 sampai didapatkan kedua kumpulan titik kosong.
11. Tak lupa untuk s2 dilakukan proses nomor 6,7,8
12. Pada akhir proses rekursi akan didapatkan simplices yang berisi pasangan-pasangan titik pembentuk *Convex Hull.*

# *Source* Pusatakadalam Bahasa Python

File Chull.py

import copy  
  
from myConvexHull import utils  
  
  
def ConvexHull(arr):  
 ch = CHull(arr)  
 return ch  
  
  
class CHull:  
 def \_\_init\_\_(self, points):  
 self.simplices = []  
 self.points = [[0, 0, 0] for \_ in range(len(points))]  
 for i in range(len(points)):  
 self.points[i] = [points[i][0], points[i][1], i]  
 self.\_\_search\_simplices()  
  
 def \_\_search\_simplices(self):  
 # Himpunan titik-titik diurutkan berdasarkan absis lalu ordinat  
 sorted\_points = utils.sort\_coordinates(self.points)  
  
 # Dua titik terekstrem merupakan bagian dari ConvexHull  
 p\_start = sorted\_points.pop(0)  
 p\_end = sorted\_points.pop()  
 # self.simplices.append([p\_start[2], p\_end[2]])  
  
 # Membagi ke 2 himpunan, ini merupakan 2 himpunan kanan dan kiri pertama  
 s1 = utils.left\_set(copy.deepcopy(sorted\_points), p\_start, p\_end)  
 s2 = utils.right\_set(copy.deepcopy(sorted\_points), p\_start, p\_end)  
  
 # Masuk ke dalam fungsi yang rekurens untuk mencari simplex-simplex ConvexHull  
 # Menggunakan algoritma divide and conquer  
 self.\_\_dnc\_chull\_left(s1, p\_start, p\_end)  
 self.\_\_dnc\_chull\_right(s2, p\_start, p\_end)  
  
 def \_\_dnc\_chull\_left(self, points, p\_start, p\_end):  
 if len(points) == 0:  
 self.simplices.append([p\_start[2], p\_end[2]])  
 else:  
 p\_farthest = utils.farthest\_point(points, p\_start, p\_end)  
 s\_left = utils.left\_set(points, p\_start, p\_farthest)  
 s\_right = utils.left\_set(points, p\_farthest, p\_end)  
 self.\_\_dnc\_chull\_left(s\_left, p\_start, p\_farthest)  
 self.\_\_dnc\_chull\_left(s\_right, p\_farthest, p\_end)  
  
 def \_\_dnc\_chull\_right(self, points, p\_start, p\_end):  
 if len(points) == 0:  
 self.simplices.append([p\_start[2], p\_end[2]])  
 else:  
 p\_farthest = utils.farthest\_point(points, p\_start, p\_end)  
 s\_left = utils.right\_set(points, p\_start, p\_farthest)  
 s\_right = utils.right\_set(points, p\_farthest, p\_end)  
 self.\_\_dnc\_chull\_right(s\_left, p\_start, p\_farthest)  
 self.\_\_dnc\_chull\_right(s\_right, p\_farthest, p\_end)

File utils.py

import copy  
  
import numpy as np  
  
  
def sort\_coordinates(points):  
 return sorted(points, key=lambda k: [k[0], k[1]])  
  
  
def determinant(p\_start, p\_end, p):  
 p\_s = copy.deepcopy(p\_start)  
 p\_e = copy.deepcopy(p\_end)  
 pp = copy.deepcopy(p)  
  
 p\_s[2] = 1  
 p\_e[2] = 1  
 pp[2] = 1  
  
 p\_s = np.array(p\_s)  
 p\_e = np.array(p\_e)  
 pp = np.array(pp)  
  
 mat = [p\_s, p\_e, pp]  
 return np.linalg.det(mat)  
  
  
def point\_distance\_to\_line(p\_start, p\_end, p):  
 p\_start, p\_end, p = to\_np\_arr(p\_start, p\_end, p)  
 return np.abs(np.cross(p\_end - p\_start, p\_start - p)) / np.linalg.norm(p\_end - p\_start)  
  
  
def farthest\_point(points, p\_start, p\_end):  
 p = None  
 for point in points:  
 if p is None:  
 p = point  
 d\_p = point\_distance\_to\_line(p\_start, p\_end, p)  
 d\_point = point\_distance\_to\_line(p\_start, p\_end, point)  
 if d\_point > d\_p:  
 p = point  
 elif d\_point == d\_p and angle(p\_start, point, p\_end) > angle(p\_start, p, p\_end):  
 p = point  
 return p  
  
  
def angle(p1, p2, p3):  
 a, b, c = to\_np\_arr(p1, p2, p3)  
  
 ba = a - b  
 bc = c - b  
  
 cosine\_angle = np.dot(ba, bc) / (np.linalg.norm(ba) \* np.linalg.norm(bc))  
 return np.arccos(cosine\_angle)  
  
  
def to\_np\_arr(p1, p2=None, p3=None):  
 p1 = np.array([p1[0], p1[1]])  
 if p2 is None:  
 return p1  
 p2 = np.array([p2[0], p2[1]])  
 if p3 is None:  
 return p1, p2  
 p3 = np.array([p3[0], p3[1]])  
 return p1, p2, p3  
  
  
def left\_set(points, p\_start, p\_end):  
 s = []  
 for p in points:  
 if determinant(p\_start, p\_end, p) > 0 and p[2] != p\_start[2] and p[2] != p\_end[2]:  
 s.append(p)  
 return s  
  
  
def right\_set(points, p\_start, p\_end):  
 s = []  
 for p in points:  
 if determinant(p\_start, p\_end, p) < 0 and p[2] != p\_start[2] and p[2] != p\_end[2]:  
 s.append(p)  
 return s

# *Screenshot* Penggunaan Pustaka

Terdapat 3 kali uji pustaka menggunakan 2 dataset yaitu iris dan breast\_cancer dari sklearn. Pustaka digunakan pada file program utama, \_\_main\_\_.py. Pengguna akan dapat memilih dataset dan variable x dan y yang akan digunakan. Hal ini dilakukan dengan menggunakan fungsi showConvexHull. Secara keseluruhan, program utama mengguanakan package pandas, matplotlib, dan sklearn.

1. Iris (Sepal length-Sepal width)

A screenshot of a computer

Description automatically generated with medium confidence

Chart, scatter chart

Description automatically generated

1. Iris (Petal length – Petal width)

Text

Description automatically generated

Chart

Description automatically generated

1. Breast Cancer (Mean radius – Mean texture)

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

# *Check Point*

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

# Link Repositori

<https://github.com/maspaitujaki/ConvexHullPython>