# LAPORAN TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA

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

# Logo Description automatically generated

Oleh

Firizky Ardiansyah

13520095

K02

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2022**

# BAB 1

# ALGORITMA *DIVIDE AND CONQUER*

Algoritma *divide and conquer* adalah algoritma penyelesaian komputasi dengan membagi sebuah persoalan menjadi beberapa subpersoalan, menyelesaikan subpersoalan tersebut, dan menggabungkan solusi subpersoalan menjadi solusi yang utuh. Prinsip *divide and conquer* banyak digunakan dalam persoalan *advanced data structure* dan geometri lanjutan. Biasanya, algoritma ini memiliki kompleksitas waktu lebih cepat dibandingkan algoritma *brute force* atau setidaknya sama dengan kompleksitas algoritma *brute force*. Salah satu penerapan strategi ini adalah dalam pencarian *convex hull*.

*Convex hull* adalah bidang convex terkecil yang dapat mengklosur kumpulan titik-titik. *Convex hull* merupakan salah satu objek geometris yang pencariannya dapat dilakukan menggunakan strategi *divide and conquer*. *Convex hull* memiliki banyak sekali penerapan salah satunya adalah dalam bidang statistik, *convex hull* digunakan untuk mengetes keterpisahan linear dari dua buah kumpulan data, yakni, dua buah kumpulan data bisa dipisahkan oleh sebuah garis linear jika dan hanya jika *convex hull* keduanya tidak saling beririsan. Pengetesan keterpisahan linear dua buah kumpulan data biasanya digunakan dalam penentuan kecocokan metode *machine learning* linear pada sekumpulan data tersebut.

Pada pembahasan ini, penulis mengimplementasikan algoritma *divide and conquer* untuk mengimplementasikan pencarian *convex hull* dalam memvisualisasikan *linear separability dataset*. Adapun langkah-langkah algoritma yang penulis rancang adalah sebagai berikut.

1. Memisahkan kumpulan data sesuai kategori.
2. Pencarian nilai ekstrem dari sekumpulan titik berdasarkan absis, jika memiliki absis yang sama, dicari ekstrem ordinat dari titik-titik tersebut. Misalkan ekstrem minimum adalah dan ekstrem maksimum adalah
3. Dibuat dua buah segmen berimpit dan . Definisikan region berarah, sebuah titik berada di atas segmen jika dan hanya jika absis dari lebih kecil dari dan representasi geometris titik berada di atas segmen . Jika absis lebih besar dibanding , titik berada di atas segmen hanya jika representasi geometrisnya ada di bawah segmen . Berikut adalah ilustrasinya

Bagian “atas”

Bagian “atas”

1. Untuk masing-masing segmen yang sudah dibentuk, bagi titik-titik menjadi dua bagian (bagian atas dan bawah relatif terhadap segmen). [proses *divide*]
2. Proses hanya titik-titik yang berada di atas segmen, dengan mencari titik yang memiliki jarak terjauh dari segmen yang dibentuk, jika dua buah titik jaraknya sama, dicari titik yang menghasilkan sudut paling besar. [proses *conquer*]
3. Jika tidak ada titik di bagian atas segmen, maka proses *conquer* subpersoalan ini sudah selesai.
4. Misalnya titik ekstrem yang telah dicari adalah dan segmen yang sedang diproses adalah , hapus segmen kemudian buat segmen baru dan , lalu lakukan kembali langkah 4, 5, 6.
5. Langkah *merge* sudah dilakukan bersamaan dengan proses *divide*, sehingga segmen-segmen akhir hasil algoritma ini adalah sebuah *convex hull* utuh dari kumpulan titik-titik tersebut. Lakukan algoritma ini pada kumpulan titik yang lain.

Adapun sebagai antarmuka program, disediakan dialog pada *command line interface* yang bisa memudahkan pengguna dalam menggunakan program. Disediakan empat buah pilihan kumpulan data, masing-masing kumpulan data bisa dicari *convex hull* dari titik-titik yang dibangun dari dua buah atribut kumpulan data tersebut. Pengguna dapat menentukan atribut mana yang akan digunakan sebagai pengujian.

# BAB 2

# *SOURCE PROGRAM*

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from sklearn import datasets  class bcolors:  # color untuk mewarnai command line  # diambil dari  # https://stackoverflow.com/questions/287871/how-to-print-colored-text-to-the-terminal  # oleh joeld  HEADER = '\033[95m'  OKBLUE = '\033[94m'  OKCYAN = '\033[96m'  OKGREEN = '\033[92m'  WARNING = '\033[93m'  FAIL = '\033[91m'  ENDC = '\033[0m'  BOLD = '\033[1m'  UNDERLINE = '\033[4m'  def openFile(id = 1):  if(id == 2):  data = datasets.load\_breast\_cancer()  print(bcolors.OKCYAN+"Loaded breast cancer dataset"+bcolors.ENDC)  elif(id==3):  data = datasets.load\_digits()  print(bcolors.OKCYAN+"Loaded digits dataset"+bcolors.ENDC)  elif(id==4):  data = datasets.load\_wine()  print(bcolors.OKCYAN+"Loaded wine dataset"+bcolors.ENDC)  else:  data = datasets.load\_iris()  print(bcolors.OKCYAN+"Loaded iris dataset"+bcolors.ENDC)  df = pd.DataFrame(data.data, columns=data.feature\_names)  df['Target'] = pd.DataFrame(data.target)  return data, df  def comp(p1, p2):  # komparasi 2 titik p1<p2 jika abis1<absis2,  # jika absisnya sama periksa ordinat11 < ordinat2  x1, y1 = p1  x2, y2 = p2  if(np.abs(x1-x2)<=1e-9):  return y1<y2  else:  return x1<x2  def minmax(points, iter, key = comp):  # mencari ekstrem dari sekumpulan titik  # iter adalah indeks subset titik yang dipilih dari points  # key adalah pembanding ekstrem  if(len(iter)):  mn = points[iter[0]]  mx = points[iter[0]]  idmn = iter[0]  idmx = iter[0]  for i in iter:  if(key(points[i], mn)):  mn = points[i]  idmn = i  if(key(mx, points[i])):  mx = points[i]  idmx = i  return idmn, idmx  def classifyRegion(point, p1, p2):  # mencari region dari point relatif  # terhadap segmen p1p2  x1, y1 = p1  x2, y2 = p2  xt, yt = point  # buat persamaan garis ay = bx + c  a = x2-x1  b = y2-y1  c = (x2-x1)\*y1 - (y2-y1)\*x1  # 1 is above  # 0 is in the line  # -1 is below  if(np.abs(a\*yt-b\*xt-c)<1e-9):  return 0  if(a\*yt > b\*xt + c):  return 1  else:  return -1  def getDist(point, p1, p2):  # mencari jarak point  # relatif terhadap segmen p1p2  x1, y1 = p1  x2, y2 = p2  xt, yt = point  # buat persamaan garis ay = bx + c  a = x2-x1  b = y2-y1  c = (x2-x1)\*y1 - (y2-y1)\*x1  denum = np.hypot(a, b)  num = np.abs(a\*yt - b\*xt - c)  if(denum <= 1e-9):  return 0  return num/denum  def dist(pt1, pt2):  # mencari jarak dua buah titik  x1, y1 = pt1  x2, y2 = pt2  return np.hypot((x1-x2), (y1-y2))  def getAngel(point, p1, p2, opp):  # mencari sudut dari p1, point, p2.  # opp adalah jarak point ke segmen p1p2  hyp1 = dist(p1, point)  hyp2 = dist(point, p2)  if(opp/hyp1 > 1 or opp/hyp2 > 1 or opp/hyp1 < -1 or opp/hyp2 < -1):  return None  return np.arccos(opp/hyp1) + np.arccos(opp/hyp2)  def mxNode(points, iter, p1, p2):  # mencari node dengan jarak terjauh dari  # segmen p1p2, jika ada dua titik yang  # jaraknya sama, dicari titik dengan sudut  # p1, point, p2 nya paling besar  def comp(pt1, pt2):  len1 = getDist(pt1, p1, p2)  len2 = getDist(pt2, p1, p2)  theta1 = getAngel(pt1, p1, p2, len1)  theta2 = getAngel(pt1, p1, p2, len2)  if(np.abs(len1-len2)<=1e-9):  if(theta1 == None or theta2 == None):  return p1[1] < p2[1]  return theta1 < theta2  else:  return len1<len2  \_, mx = minmax(points, iter, comp)  return mx  def Hull(points, iter, p1, p2, solutions):  # pencarian convexhull setelah titik-titik  # dibagi dua menjadi bagian atas dan bawah  if(len(iter)):  mx = mxNode(points, iter, points[p1], points[p2])  solutions.remove([p1, p2])  solutions += [[p1, mx], [mx, p2]]  s1 = np.array([i for i in iter if classifyRegion(points[i], points[p1], points[mx])==1])  s2 = np.array([i for i in iter if classifyRegion(points[i], points[mx], points[p2])==1])  solutions = Hull(points, s1, p1, mx, solutions)  solutions = Hull(points, s2, mx, p2, solutions)  return solutions  def MyConvexHull(points):  # membagi titik-titik menjadi atas dan bawah, kemudian  # masing-masing bagian akan dicari convex hullnya  iter = [\_ for \_ in range(len(points))]  mn, mx = minmax(points, iter)  solutions = [[mn, mx], [mx, mn]]  s1 = np.array([i for i in range(len(points)) if classifyRegion(points[i], points[mn], points[mx])==1])  s2 = np.array([i for i in range(len(points)) if classifyRegion(points[i], points[mn], points[mx])==-1])  solutions = Hull(points, s1, mn, mx, solutions)  solutions = Hull(points, s2, mx, mn, solutions)    return solutions  def plotHull(data, df, x=0, y=1):  # menerima datasets, kemudian mengambil kolom ke-x  # dan kolom ke-y untuk dijadikan data yang dicari  # convex hull-nya  plt.figure(figsize = (10, 6))  colors = ['blue','red','green','yellow', 'azure', 'lime', 'darkgreen', 'black', 'aqua','pink', 'crimson']  plt.title(data.feature\_names[x].title() + ' vs ' + data.feature\_names[y].title())  plt.xlabel(data.feature\_names[x])  plt.ylabel(data.feature\_names[y])  for i in range (len(data.target\_names)):  bucket = df[df['Target'] == i]  bucket = bucket.iloc[:,[x, y]].values  hull = MyConvexHull(np.array(bucket))  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target\_names[i])  for simplex in hull:  plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i%11])  plt.legend()  plt.show()  def interface():  # antarmuka program  print(bcolors.BOLD + "WELCOME!" + bcolors.ENDC)  print("This is a program to find a convex hull from")  print("a given dataset. The goal of this program is")  print(f"to visualize {bcolors.BOLD}linear separability of dataset{bcolors.ENDC}, so")  print(f"the dataset provided {bcolors.WARNING}required{bcolors.ENDC}")  print("a target classification in their attributes.\n\n")  print(bcolors.OKGREEN + "Below are sample datasets." + bcolors.ENDC)  print("You can choose one of them by specifying the index of dataset you wish to choose")  print("""  1. Iris  2. Breast Cancer  3. Digits  4. Wine  """)  def start():  # fungsi untuk melakukan pencarian convex hull berdasarkan spesifikasi  # yang diberikan user  try:  id = int(input("Type the index of dataset you wish to analyze: "))  if(id > 4 or id < -1):  print(bcolors.FAIL + "Your input is not valid" + bcolors.ENDC)  print(bcolors.WARNING + "Using default dataset..." + bcolors.ENDC)  data, df = openFile(1)  else:  data, df = openFile(id)  except:  print(bcolors.FAIL + "Your input is not valid" + bcolors.ENDC)  print("Using default dataset...")  data, df = openFile(1)    print("These are dataset's attribute you can choose")  for i in range(len(data.feature\_names)):  print(f"{i+1}. {data.feature\_names[i]}")    try:  x = int(input("Type index of attribute to be x-coordinate: "))  y = int(input("Type index of attribute to be y-coordinate: "))  mx = len(data.feature\_names)  if(x == y or x > mx or y > mx or x <= 0 or y <= 0):  print(bcolors.FAIL + "Your input is not valid" + bcolors.ENDC)  print("Using default attributes...")  plotHull(data, df, 0, 1)  else:  plotHull(data, df, x-1, y-1)  except:  print(bcolors.FAIL + "Your input is not valid" + bcolors.ENDC)  print("Using default attributes...")  plotHull(data, df, 0, 1)    if \_\_name\_\_ == '\_\_main\_\_':  interface()  start() |

# BAB 3

# TANGKAPAN LAYAR PENGETESAN

Pada bagian ini, akan ditampilkan beberapa tangkapan layar pada keempat kumpulan data yang tersedia. Masing-masing kumpulan data akan diuji dua kali dengan penggunaan atribut yang berbeda. Kumpulan data ini adalah kumpulan data *toy* yang tersedia pada pustaka *sklearn*. Terdapat *dataset default* dan atribut *default* jika pengguna tidak benar dalam memasukkan *input* atau mengosongkan *input*. *Dataset default* pada program ini adalah *dataset* Iris, sedangkan atribut *default* adalah atribut pertama sebagai absis, atribut kedua sebagai ordinat.

Text

Description automatically generated

Gambar Antarmuka Awal

1. Data Iris

|  |  |
| --- | --- |
| Text  Description automatically generated |  |

Gambar Pengujian *Dataset* Iris menggunakan Atribut *Default* (atribut pertama dan kedua)

|  |  |
| --- | --- |
|  |  |

Gambar Pengujian Dataset Iris pada Atribut Ketiga dan Keempat

1. Data Breast Cancer

|  |  |
| --- | --- |
|  |  |

Gambar Pengujian Dataset Breast Cancer pada Atribut *Default*

|  |  |
| --- | --- |
|  |  |

Gambar Pengujian Dataset Breast Cancer pada Atribut 11 dan 21

1. Data Digits

|  |  |
| --- | --- |
|  |  |
|  |

Gambar Pengujian Dataset Digits pada Atribut *default*

|  |  |
| --- | --- |
|  |  |
|  |

Gambar Pengujian Dataset Digits pada Atribut 2 dan 5

1. Data Wine

|  |  |
| --- | --- |
|  |  |

Gambar Pengujian Dataset Wine pada Atribut *Default*

|  |  |
| --- | --- |
|  |  |

Gambar Pengujian Dataset Wine pada Atribut 5 dan 13

# BAB 4

# DOKUMENTASI

Program implementasi *convex hull* dalam visualisasi *linear separability dataset* menggunakan algoritma *divide and conquer* lebih lengkap dapat diakses pada <https://github.com/firizky29/convex-hull>.

# BAB 5

# TABEL *CHECK LIST*

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| * 1. Pustaka *myConvexHull* berhasil dibuat dan tidak ada kesalahan | √ |  |
| * 1. *Convex hull* yang dihasilkan sudah benar | √ |  |
| * 1. Pustaka *myConvexHull* dapat digunakan untuk menampilkan *convex hull* setiap label dengan warna berbeda. | √ |  |
| * 1. **Bonus**: program dapat menerima input dan menuliskan output untuk dataset lainnya. | √ |  |