LAPORAN TUGAS PROGRAM

MACHINE LEARNING

PROBABILISTIK NEURAL NETWORK

Nama : I Putu Indra Aristya

NIM : 1301154219

Kelas : IF39-09

Mata Kuliah : Pembelajaran Mesin

Universitas Telkom

BANDUNG

2018

I. Deskripsi Masalah

Dalam tugas program ini, mahasiswa diminta untuk memvisualisasikan data latih yang digunakan menjadi bentuk *plot* untuk mengetahui persebaran data dan *error* yang ada. Selain itu mahasiswa juga diminta untuk melakukan prediksi klasifikasi terhadap data test yang diberikan. Jumlah data yang diberikan adalah 150 data latih dan 30 data tes. Setiap data yang ada, memiliki 3 atribut dan diklasifikasikan ke dalam 3 kelas, yaitu kelas 0, kelas 1 dan kelas 2. Dari 150 data latih tersebut, kita diminta untuk menentukan *smoothing parameter* yang berpengaruh terhadap melakukan klasifikasi dengan rumus PDF.

II. Landasan Teori

Probabilistic Neural Network (PNN) adalah salah satu metode yang dapat digunakan untuk melakukan klasifikasi. PNN adalah bentuk *feed forward* dari Neural Network namun digabungkan dengan probabilitas yang dilihat dari Bayesian Network. PNN memiliki 4 layer utama, yaitu layer input, *hidden* layer, *sum* layer, dan layer output. Keempat layer ini menggambarkan proses/algoritma menggunakan PNN.

Algoritma PNN.

1. Data input masuk ke layer input
2. Layer input membawa nilai setiap atribut ke setiap neuron yang ada di hidden layer (fully-connected)
3. Pada setiap neuron akan dilakukan penghitungan fungsi aktivasi
4. Data yang memiliki data dengan kelas yang sama akan dijumlahkan hasil penghitungan fungsi aktivasinya
5. Nilai penjumlahan yang lebih besar akan menentukan kelas dari data yang masuk tersebut

Neuron yang ada pada setiap layer memiliki jumlah yang sesuai dengan data. Biasanya, jumlah layer input akan sesuai dengan jumlah atribut setiap data. Hidden layer akan berjumlah sama dengan jumlah data latih. Sum layer akan berjumlah sebanyak kelas

yang ada dan layer output memiliki 1 neuron, karena hasil akhir hanya boleh memiliki 1 nilai.

Fungsi aktivasi yang digunakan adalah PDF dari Gaussian, yaitu:

$$f(x) = \frac{1}{n\sigma}\sum_{k=1}^{n} e^{-\frac{(x-x')^2}{\sigma^2}}$$

dimana:

σ adalah *smoothing effect*

n adalah jumlah data

x adalah data input

x' adalah data yang ingin diklasifikasikan

III. *Code* Program

Dalam tugas program-1 ini, saya menggunakan bahasa pemrograman *Python* dengan IDE *PyCharm CE*. Dalam program ini, saya menentukan parameter *smoothing effect* (SE) dengan metode *brute force*. Pertama, dicoba memberi nilai SE sebesar 1 hingga 10, yang didapatkan terbesar pada nilai 2. Kemudian dilakukan pemilihan kembali antara 2 hingga 3 dengan jarak perubahan setiap 0.1, dan didapat nilai 2.1. Namun, saya mencoba untuk melakukan *brute force* dari nilai 0 sampai 1, dengan jarak perubahan setiap 0.1. Berikut adalah hasil *brute force* dari nilai 0 sampai 1.

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/l
P =  0.1 Akurasi =  76.0
P =  0.2 Akurasi =  76.0
P =  0.30000000000000004 Akurasi =  76.0
P =  0.4 Akurasi =  80.0
P =  0.5 Akurasi =  80.0
P =  0.6 Akurasi =  82.0
P =  0.7 Akurasi =  84.0
P =  0.7999999999999999 Akurasi =  86.0
P =  0.8999999999999999 Akurasi =  86.0
P =  0.9999999999999999 Akurasi =  86.0

Process finished with exit code 0
```

Gambar 1 Hasil *brute force* nilai *smooth* 0 – 1

Dari hasil yang dicoba, dapat dilihat nilai *smooth* yang memiliki akurasi paling baik adalah 0.79999 – 0.999999. Maka, saya memilih nilai tengah untuk digunakan, yaitu 0.899999 yang dibulatkan menjadi 0.9 dengan akurasi pada data latih (data ke-101 sampai 150 menjadi data validasi dan data ke-1 sampai 100 menjadi data training) mendapat 86%.

Berikut ditampilkan kode program untuk data validasi.

```
from numpy import genfromtxt, math
import numpy as np
import matplotlib.pyplot as plot
from mpl_toolkits.mplot3d import Axes3D
fig = plot.figure()
ax = fig.add_subplot(111, projection='3d')


dataTrain = genfromtxt('Workbook2.csv',delimiter=',') #workbook2.csv adalah file data
latih

att1 = dataTrain[:,0]
att2 = dataTrain[:,1]
att3 = dataTrain[:,2]
kelas = dataTrain[:,3]

#------Code untuk visualisasi data latih dengan scatter plot------
ax.scatter(xs=att1, ys=att2, zs=att3,c=kelas)
plot.show()


#------Code Fungsi-----
smooth = 0.9
T = 0

def pangkat(x,y,z,xtrain,ytrain,ztrain):
    pangkatexp = -1*((((x-xtrain)**2) + ((y-ytrain)**2) + ((z-ztrain)**2)) /
((smooth)**2))
    return pangkatexp

def exponen(x):
    return math.exp(x)

def classification0(y):
    x = 1/(32*smooth) #32 adalah jumlah data latih dengan kelas 0
    return x*y

def classification1(y):
    x = 1 / (31 * smooth) #31 adalah jumlah data latih dengan kelas 0
    return x * y

def classification2(y):
    x = 1 / (37 * smooth) #37 adalah jumlah data latih dengan kelas 0
    return x * y

#-----Code Proses-----

label0 = 0
label1 = 0
label2 = 0

o = 0
hasil = [0]*50


#melakukan penghitungan dengan PDF pada fungsi pangkat dan eksponen
```

```
for i in range(100,150):
    for j in range(0,100):
        x =
pangkat(dataTrain[j,0],dataTrain[j,1],dataTrain[j,2],dataTrain[i,0],dataTrain[i,1],dat
aTrain[i,2])
        y = exponen(x)

        #hasil perhitungan data dengan PDF yang kelasnya sama, dijumlahkan (sum
function)
        if (dataTrain[j,3] == 0.0):
            label0 = label0 + y
        elif (dataTrain[j,3] == 1.0):
            label1 = label1 + y
        elif (dataTrain[j,3] == 2.0):
            label2 = label2 + y

    #dilakukan penghitungan klasifikasi dengan rumus (1/n*sigma)*(jumlah perhitungan
PDF) dimana n adalah jumlah data dengan kelas data tersebut
    label0 = classification0(label0)
    label1 = classification1(label1)
    label2 = classification2(label2)


    #akan ditentukan nilai kelas paling besar yang menjadi hasil prediksi kelas dari
data tersebut
    if (label0 > label1 and label0 > label2):
        hasil[o] = 0
        o = o + 1
    elif (label1 > label0 and label1 > label2):
        hasil[o] = 1
        o = o + 1
    elif (label2 > label0 and label2 > label1):
        hasil[o] = 2
        o = o + 1

    label0 = 0
    label1 = 0
    label2 = 0
    print(hasil[o-1])

#melakukan perhitungan akurasi
j = 100
for i in range(0,50):
    if (int(hasil[i]) == (int(dataTrain[j,3]))):
        T = T+1
    j = j+1

print((T/50)*100)
```

Kode program untuk melakukan prediksi kelas pada data testing.

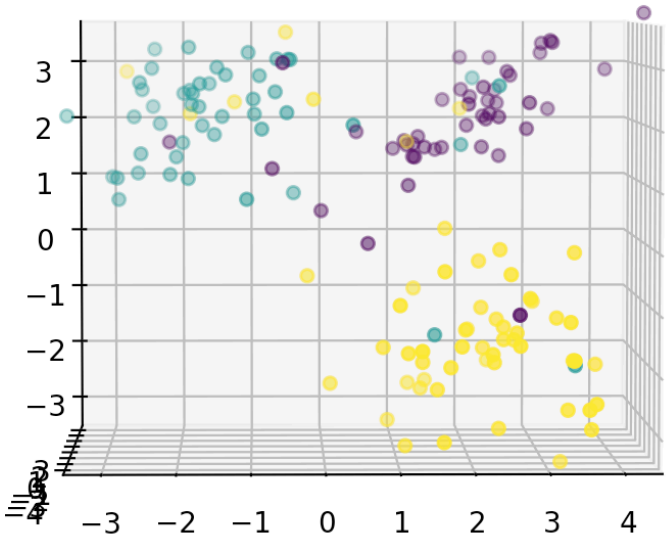
```
from numpy import genfromtxt, math
import numpy as np
import matplotlib.pyplot as plot
from mpl_toolkits.mplot3d import Axes3D
fig = plot.figure()
ax = fig.add_subplot(111, projection='3d')


dataTrain = genfromtxt('Workbook2.csv',delimiter=',')
dataTest = genfromtxt('datatestPNN.csv',delimiter=',')

att1 = dataTrain[:,0]
att2 = dataTrain[:,1]
att3 = dataTrain[:,2]
kelas = dataTrain[:,3]

# ------Code untuk visualisasi data latih dengan scatter plot------
# ax.scatter(xs=att1, ys=att2, zs=att3,c=kelas)
# plot.show()
```

```
#------Code Fungsi-----

smooth = 0.9
T = 0

def pangkat(x,y,z,xtrain,ytrain,ztrain):
    pangkatexp = -1*((((x-xtrain)**2) + ((y-ytrain)**2) + ((z-ztrain)**2)) /
((smooth)**2))
    return pangkatexp

def exponen(x):
    return math.exp(x)

def classification0(y):
    x = 1/(46*smooth) #46 adalah jumlah data latih dengan kelas 0
    return x*y

def classification1(y):
    x = 1 / (48 * smooth) #48 adalah jumlah data latih dengan kelas 0
    return x * y

def classification2(y):
    x = 1 / (56 * smooth) #56 adalah jumlah data latih dengan kelas 0
    return x * y

#-----Code Proses-----

label0 = 0
label1 = 0
label2 = 0

o = 0
hasil = [0]*30

#melakukan penghitungan dengan PDF pada fungsi pangkat dan eksponen
for i in range(0, 30):
    for j in range(0, 150):
# for i in range(100,150):
#     for j in range(0,100):
#         x =
pangkat(dataTrain[j,0],dataTrain[j,1],dataTrain[j,2],dataTrain[i,0],dataTrain[i,1],dataTr
ain[i,2])
        x = pangkat(dataTrain[j, 0], dataTrain[j, 1], dataTrain[j, 2], dataTest[i, 0],
dataTest[i, 1], dataTest[i, 2])
        y = exponen(x)

        #hasil perhitungan data dengan PDF yang kelasnya sama, dijumlahkan (sum function)
        if (dataTrain[j,3] == 0.0):
            label0 = label0 + y
        elif (dataTrain[j,3] == 1.0):
            label1 = label1 + y
        elif (dataTrain[j,3] == 2.0):
            label2 = label2 + y

    # dilakukan penghitungan klasifikasi dengan rumus (1/n*sigma)*(jumlah perhitungan
PDF) dimana n adalah jumlah data dengan kelas data tersebut
    label0 = classification0(label0)
    label1 = classification1(label1)
    label2 = classification2(label2)

    # akan ditentukan nilai kelas paling besar yang menjadi hasil prediksi kelas dari
data tersebut
    if (label0 > label1 and label0 > label2):
        hasil[o] = 0
        # print(hasil[o])
        o = o + 1
    elif (label1 > label0 and label1 > label2):
        hasil[o] = 1
        # print(hasil[o])
        o = o + 1
    elif (label2 > label0 and label2 > label1):
        hasil[o] = 2
        # print(hasil[o])
```

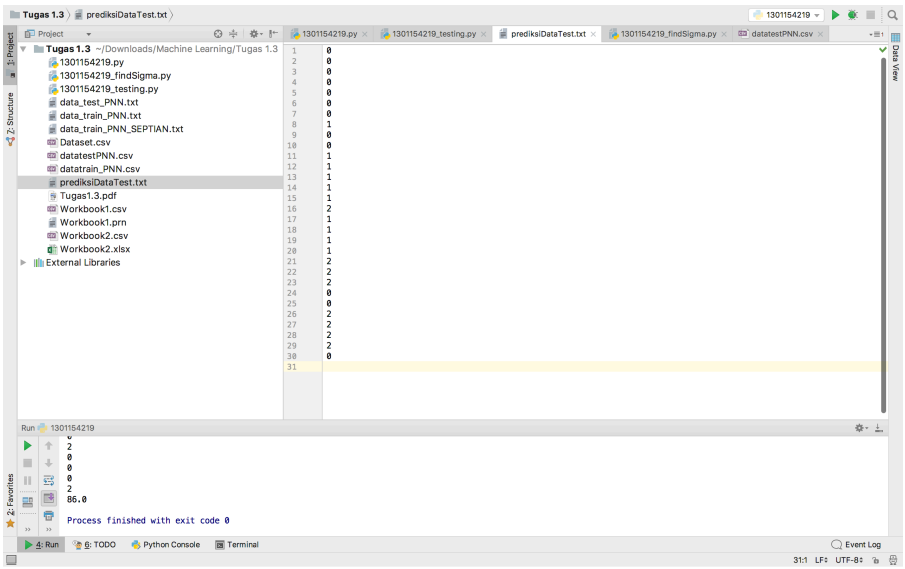
```
        o = o + 1

    label0 = 0
    label1 = 0
    label2 = 0

#hasil prediksi di simpan pada file prediksiDataTest.txt
file = open('prediksiDataTest.txt', 'w')
for i in range(0,30):
    file.write(str(int(hasil[i])))
    file.write('\n')
file.close()

#menampilkan hasil visualisasi data test
ax.scatter(xs=dataTest[:,0], ys=dataTest[:,1], zs=dataTest[:,2],c=hasil)
plot.show()
```

IV. Hasil Keluaran Program dan Nilai Minimal

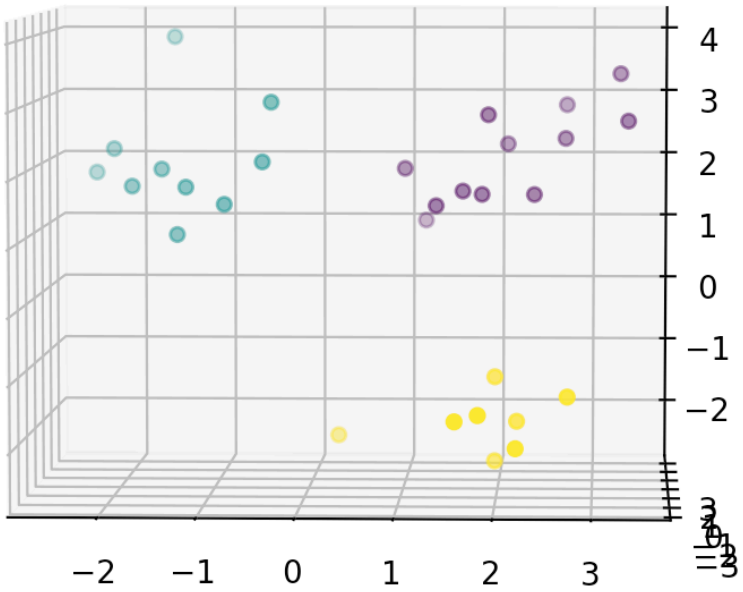
Berikut adalah hasil keluaran dari program yang dijalankan dan nilai minimal yang pernah di dapat.



Gambar 2 Hasil visualisasi data latih



Gambar 3 Hasil prediksi data test



Gambar 4 Hasil visualisasi prediksi data test

V. Kesimpulan

Dari hasil yang didapat, saya simpulkan bahwa *brute force* dapat digunakan sebagai salah satu cara untuk menentukan nilai *smooting*. Namun, menurut saya jika jumlah data yang ingin diolah tersebut banyak (lebih dari 500 data), lebih baik mencari dengan menurunkan interval data sehingga data lebih mudah di proses.

Daftar Pustaka

Suyanto. (2014). *Artificial Intelligence* (Vol. 2). Bandung: Informatika Bandung.