

2018

Probabilistic Neural Network

Laporan Tugas 1.3 Machine Learning

Dosen Pengampu : ANDITYA ARIFianto, S.T., M.T. **(ADF)**

Oleh :

Fakhri Fauzan

1301154374

IF – 39 – 10

1. Deskripsi Kasus

Probabilistic Neural Network (PNN) merupakan bentuk Artificial Neural Network (Jaringan Saraf Tiruan) dengan penambahan probabilistic yang dalam hal ini digunakan fungsi gaussian untuk menentukan nilai dari Probability Density Function (PDF) nya. Fungsinya seperti berikut :

$$f(x) = \frac{1}{(2\pi)^{p/2} \sigma^p n} \sum_{k=1}^n e^{-\frac{\|x-x_k\|^2}{2\sigma^2}}$$

Dalam kasus ini diberikan 150 Data Training dan 30 Data Testing. Dimana 150 Data Training dibagi dalam 3 kelas (0,1,2) dan 3 parameter yang digunakan yaitu x,y,z atau bisa disebut sebagai persebaran 3 Dimensi. Untuk parameter n yang digunakan didapatkan dari jumlah n dalam sebuah class. Parameter sigma juga merupakan nilai dari smoothing nilainya pada setiap class dapat berbeda. Dengan PNN kita nantinya akan melakukan klasifikasi terhadap data set testing.

2. Rancangan Metode

Untuk menyelesaikan permasalahan diatas, diperlukan tahap-tahap sebagai berikut :

1. Dataset Training

Dalam Kasus ini dataset Training disediakan 150 data dalam file data_train_PNN.txt.

2. Dataset Testing

Dalam Kasus ini dataset Training disediakan 30 data dalam file data_test_PNN.txt.

3. Menghitung Smoothing (Sigma)

Setiap kelas pada data testing akan dihitung nilai Sigma nya, nilai Sigma nantinya akan digunakan dalam proses menentukan nilai PDFnya, lalu akan disimpan kedalam array.

4. Menentukan nilai F(x)

Nilai F(x) didapatkan dari penjumlahan seluruh hasil G(x) yang di dapatkan dari penghitungan antara data testing dibandingkan dengan keseluruhan data training, nantinya hasil G(x) akan dikelompokan berdasarkan kelas data training, lalu seluruhnya dijumlahkan untuk mendapatkan nilai F(x).

5. Menentukan Kelas data Testing

Setelah seluruh nilai $F(x)$ dari semua jenis kelas sudah didapatkan. Untuk menentukan kelas dari data training kita cukup mencari nilai dari $F(x)$ yang paling maksimum atau dengan kata lain dengan probability terbesar.

3. Observasi dan Simulasi Metode

Berikut adalah script program ProbabilisticNN menggunakan python.

```

===== PROBABILISTIC NEURAL NETWORK =====
# Machine Learning Task 1.3 (2018)
# Fakhri Fauzan
# IF - 39 - 10
# 1301154374
# Informatics Engineering, Telkom University
=====

import math
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

class ProNN :
    def __init__(self):
        self.training = []
        self.testing = []
        self.result = []
        self.n = [0, 0, 0]
        self.prob = [0.0, 0.0, 0.0]
        self.sigma = [0.0, 0.0, 0.0]

    def visualize(self, file):
        x = []
        y = []
        z = []
        count = 0

        with open(file, "r") as file:
            tLine = file.readlines()
            for t in tLine:
                if (count > 0):
                    row = t.split()
                    if (int(row[3]) == 0):
                        x.append([float(row[0]), float(row[1]), float(row[2])])
                    elif (int(row[3]) == 1):
                        y.append([float(row[0]), float(row[1]), float(row[2])])
                    else:
                        z.append([float(row[0]), float(row[1]), float(row[2])])
                    count = count + 1
        x = np.array(x)
        y = np.array(y)
        z = np.array(z)

        fig = plt.figure()
        data = fig.add_subplot(111, projection='3d')
        data.set_xlabel('Attr 1')
        data.set_ylabel('Attr 2')
        data.set_zlabel('Attr 3')
        data.scatter(x[:,0], x[:,1], x[:,2], c='r', marker='o')
        data.scatter(y[:,0], y[:,1], y[:,2], c='g', marker='+')
        data.scatter(z[:,0], z[:,1], z[:,2], c='b', marker='^')
        plt.show()

```

```

def loadTraining(self, file):
    train = []
    count = 0
    with open(file, "r") as file:
        tLine = file.readlines()
        for x in tLine:
            if (count > 0):
                row = x.split()
                train.append([float(row[0]), float(row[1]), float(row[2]),
float(row[3])])
                self.n[int(row[3]) - 1] = int(self.n[int(row[3]) - 1]) + 1
            count = count + 1
    self.training = sorted(train, key=lambda train:train[3], reverse=False)

def loadTesting(self, file):
    count = 0
    with open(file, "r") as file:
        tLine = file.readlines()
        for x in tLine:
            if (count > 0):
                row = x.split()
                self.testing.append([float(row[0]), float(row[1]),
float(row[2])])
            count = count + 1

def writeResultTesting(self, file):
    file = open(file, "w")
    for i in range(len(self.result)):
        file.write(str(self.result[i]) + "\n")
    file.close()

def smoothing(self, g=None):
    if g is None:
        g = 1.4
    dtot1 = []
    dtot2 = []
    dtot3 = []
    for i in range(len(self.training)):
        fmd = []
        for j in range(len(self.training)):
            a = 0
            if (self.training[i] != self.training[j] and self.training[i][3]
== self.training[j][3]) :
                for z in range(0, 3):
                    a = a + pow((float(self.training[j][z]) -
float(self.training[i][z])), 2)
                d = math.sqrt(a)
                fmd.append(d)
            if (self.training[i][3] == 1) :
                dtot1.append(min(fmd))
            elif (self.training[i][3] == 2) :
                dtot2.append(min(fmd))
            else :
                dtot3.append(min(fmd))
    self.sigma = [float((g * sum(dtot1)) / len(dtot1)), float((g *
sum(dtot2)) / len(dtot2)), float((g * sum(dtot3)) / len(dtot3))]

def main(self) :
    #fungsi g(x)
    for z in range(len(self.testing)):
        self.prob = [0.0, 0.0, 0.0]
        for i in range(len(self.training)):
            a = 0.0
            p = 3
            kelas = int(self.training[i][3])
            for j in range(0,3):
                a = a + float(pow((float(self.testing[z][j]) -

```

```

float(self.training[i][j]),2))
        b = 2*(pow(self.sigma[kelas],2))
        dbp = pow(2*math.pi,p/2)
        dbs = pow(self.sigma[kelas],p)
        dbn = self.n[kelas]
        # print("DBA = " + str(dbp) + " | DBS = " + str(dbs) + " | DBN = "
" + str(dbn))
        d = 1 / (dbp * dbs * dbn)
        form = math.exp(-(a/b))
        x = d*form
        # print("A = " + str(a) + " | B = " + str(b) + " | Form = " +
str(form) + " | X = " + str(x) + " | Kelas = " + str(kelas))
        self.prob[kelas] = self.prob[kelas] + x
        self.result.append(self.decision())

    def decision(self):
        index = self.prob.index(max(self.prob))
        return index

    def getAccuracy(self, file, g):
        train = []
        expected = []
        count = 0
        with open(file, "r") as file:
            tLine = file.readlines()
            for x in tLine:
                if (count > 0):
                    row = x.split()
                    if (count < 101) :
                        train.append([float(row[0]), float(row[1]),
float(row[2]), float(row[3])])
                        self.n[int(row[3]) - 1] = int(self.n[int(row[3]) - 1]) +
1
                    else :
                        expected.append(int(row[3]))
                        self.testing.append([float(row[0]), float(row[1]),
float(row[2])])
                count = count + 1
            self.training = sorted(train, key=lambda train: train[3],
reverse=False)
            self.smoothing(g)
            self.main()
            cSama = 0.0
            for x in range(len(self.result)):
                if (self.result[x] == expected[x]):
                    cSama += 1
            akurasi = (cSama / len(expected)) * 100
            return akurasi

# .:: MAIN PROGRAM ::.
#===== Load Data =====
fileT = "data_train_PNN.txt"
fileTs = "data_test_PNN.txt"
#=====

```

```

#===== PROSES OBSERVASI MENCARI NILAI G MAKSIMUM AKURASI =====
g = []
a = np.linspace(0.1, 2.0, num=20, endpoint=True)
fileOb = open("observasi.txt", "w")
for i in range(len(a)) :
    g.append(round(a[i],2))
# print ("Himpunan G : " + str(g))
fileOb.write("Himpunan G : " + str(g))
for k in range(len(g)) :
    pnn = ProNN()
    fileOb.write("G = " + str(g[k]) + " | Akurasi : " +
str(pnn.getAccuracy(fileT, g[k])) + "%" + "\n")
    fileOb.write("Smoothing Value[0,1,2] : " + str(pnn.sigma) + "\n")
fileOb.close()
#=====

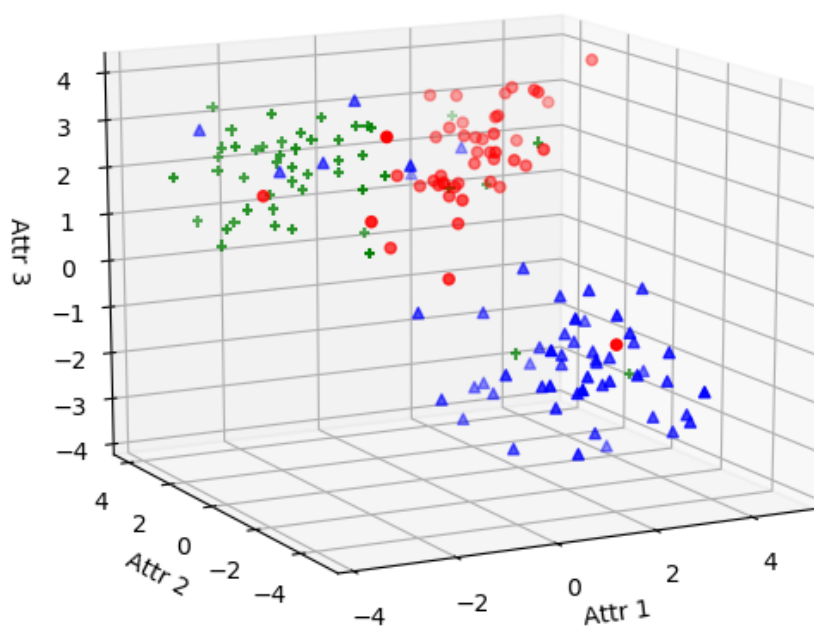
# ===== PROSES KLASIFIKASI DATA TESTING =====
pnn = ProNN()
pnn.loadTraining(fileT)
pnn.loadTesting(fileTs)
pnn.smoothing()
print("Smoothing Value : " + str(pnn.sigma))
pnn.main()
pnn.writeResultTesting("prediksi.txt")
print("Data Testing Berhasil di Klasifikasi!")
pnn.visualize(fileT)
# =====

```

Beberapa observasi yang dilakukan, meliputi :

3.1. Visualisasi Data Training

Dalam kasus ini digunakan library matplotlib untuk dapat memvisualisasikan data training. Data training di visualisasikan secara 3D dengan pembagian kelas 0 (merah), kelas 1 (hijau), kelas 2 (biru). Observasi dilakukan sebagai berikut :



3.2. Fungsi Utama untuk menentukan kelas dengan metode Probabilistic Neural Network

Dalam kode program diatas fungsi utama metode PNN dituangkan dalam beberapa method, pertama method smoothing, method main, dan method decision. Selain method tersebut digunakan untuk membantu load data dari txt dan visualisasi menggunakan scatterplot.

3.3. Parameter yang Digunakan

3.3.1. Nilai G

Untuk mendapatkan nilai G perlu mendapatkan nilai akurasi model, dimana akurasi data pada akurasi model digunakan 150 data training yang di split menjadi 1 : 3 atau 100 data training dan 50 data validasi. Untuk menentukan nilai G yang optimal digunakan metode brute force dengan rentang nilai[0. – 2.0] sebanyak 20 kali. Didapatkan hasil akurasi model sebagai berikut (hasil dapat dilihat di **observasi.txt**):

Himpunan G : [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0]

G	Smoothing Values Data Validasi [Attr1, Attr2, Attr3]	Akurasi Model (3 : 1) 100 Train 50 Validasi
0.1	[0.08107464939515667, 0.09656540359858616, 0.06633749194434972]	76.0%
0.2	[0.16214929879031334, 0.19313080719717232, 0.13267498388869944]	78.0%
0.3	[0.24322394818547, 0.28969621079575847, 0.19901247583304912]	76.0%
0.4	[0.3242985975806267, 0.38626161439434464, 0.2653499677773989]	76.0%
0.5	[0.4053732469757833, 0.48282701799293076, 0.33168745972174857]	80.0%

0.6	[0.48644789637094, 0.5793924215915169, 0.39802495166609825]	82.0%
0.7	[0.5675225457660966, 0.6759578251901031, 0.464362443610448]	86.0%
0.8	[0.6485971951612534, 0.7725232287886893, 0.5306999355547978]	86.0%
0.9	[0.72967184455641, 0.8690886323872754, 0.5970374274991475]	88.0%
1.0	[0.8107464939515666, 0.9656540359858615, 0.6633749194434971]	88.0%
1.1	[0.8918211433467235, 1.0622194395844478, 0.7297124113878469]	88.0%
1.2	[0.97289579274188, 1.1587848431830339, 0.7960499033321965]	88.0%
1.3	[1.0539704421370366, 1.25535024678162, 0.8623873952765463]	88.0%
1.4	[1.1350450915321932, 1.3519156503802061, 0.928724887220896]	90.0%
1.5	[1.2161197409273499, 1.4484810539787922, 0.9950623791652458]	90.0%
1.6	[1.2971943903225067, 1.5450464575773786, 1.0613998711095955]	90.0%
1.7	[1.3782690397176633, 1.6416118611759647, 1.1277373630539451]	90.0%
1.8	[1.45934368911282, 1.7381772647745508, 1.194074854998295]	90.0%
1.9	[1.5404183385079766, 1.834742668373137, 1.2604123469426445]	90.0%
2.0	[1.6214929879031332, 1.931308071971723, 1.3267498388869943]	90.0%

3.3.2. Smoothing Value (Sigma)

Smoothing Value didapatkan berdasarkan observasi sebelumnya, didapatkan nilai akurasi tertinggi (diberi label orange) dengan nilai $G = 1.4$ sehingga didapatkan nilai smoothing untuk masing – masing kelas seperti berikut :

Kelas	Smoothing Value Data Testing
0	0.9063559400706809
1	1.0891092430911102
2	0.843145723468938

3.4. Simulasi Metode

Simulasi pengujian dilakukan dengan tahap sebagai berikut :

3.4.1. Load data latih dan data uji.

Pada tahap ini dilakukan dalam method loadTraining dan loadTesting lalu dimasukan kedalam list training dan testing.

3.4.2. Proses klasifikasi terhadap data uji.

Pada tahap ini dilakukan dengan cara pemanggilan beberapa method secara berurutan mulai dari loadtraining, loadTesting, Smoothing, dan Main.

3.4.3. Output hasil klasifikasi

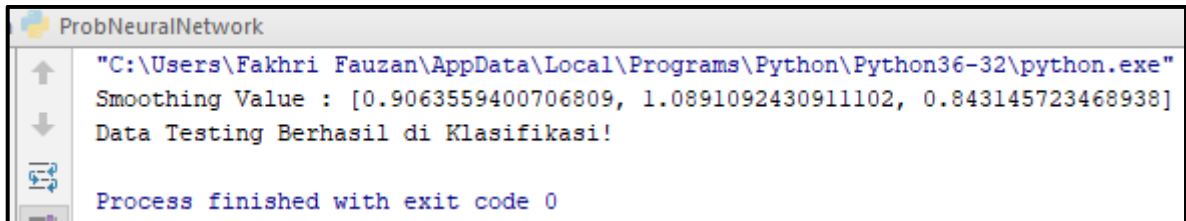
Setelah dilakukan proses klasifikasi, ditulis dalam bentuk file “prediksi.txt”

```
0
0
0
0
0
0
0
1
0
0
1
1
1
1
1
1
2
1
1
1
1
2
```

```
2
2
0
0
2
2
2
2
0
```

4. Screenshot Output Program

Output dari program tersebut. Data hasil klasifikasi akan ditulis dalam file txt 'prediksi.txt'



```
ProbNeuralNetwork
"C:\Users\Fakhri Fauzan\AppData\Local\Programs\Python\Python36-32\python.exe"
Smoothing Value : [0.9063559400706809, 1.0891092430911102, 0.843145723468938]
Data Testing Berhasil di Klasifikasi!
Process finished with exit code 0
```