2018

Tugas Program Probabilistic Neural Network

Dosen Pengampu: Dr. Suyanto, S.T, M.Sc

Nama: Aziza Hayupratiwi

NIM: 1301150440

Kelas : IF 39-06

2/2/2018

1. Deskripsi Kasus

Probabilistic Neural Network atau Jaringan Saraf Tiruan Probabilistik adalah model yang dibentuk berdasarkan penaksir fungsi padat yang berguna untuk memberikan pengklasifikasian yang sangat baik dan cepat dalam pelatihan karena dilakukan hanya dalam satu tahap pelatihan.

Data yang digunakan untuk data training ada sebanyak 150 data dengan tiga atribut dan tiga klasifikasi yaitu 0,1, dan 2. Data yang digunakan pada program kali ini bernilai kontinyu dan klasifikasinya bernilai diskrit. Data yang digunakan sebagai training disimpan dalam file data_train_PNN.txt dan data yang digunakan sebagai data test disimpan dalam file data_test_PNN.txt.

2. Rancangan Metode

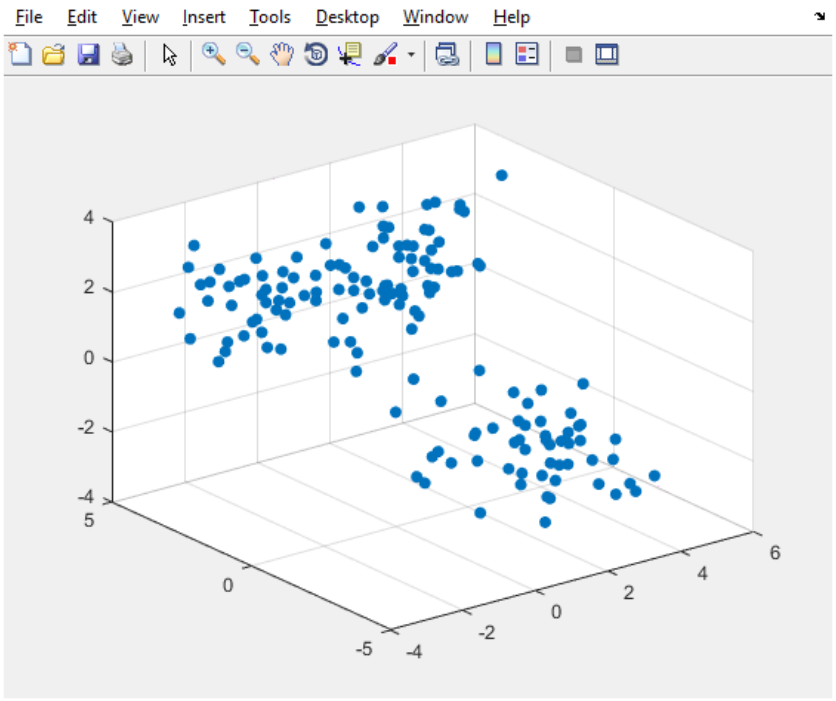
Untuk menyelesaikan masalah ini digunakan bahasa pemrograman matlab dengan metode PNN terhadap data training untuk dijadikan sebagai data testing. Dalam hal ini digunakan 1 buah data train yang nantinya akan diuji untuk mengetahui klasifikasi, apakah sesuai dengan data atau tidak.

3. Simulasi Metode

Tahap pertama yaitu membuka file data dalam bentuk txt dan menampilkan scatter plot persebaran data di program bernama **trainPNN.m.** Klasifikasi dataTrain akan disimpan dalam file dataTrain.mat. Karena data dibaca dalam bentuk table, maka harus dikonversi ke array agar dapat dilakukan perhitungan probabilitas.

```
%membaca file data train PNN
data_train = readtable('data_train_PNN.txt');
data_train = sortrows(data_train,'label','ascend');
dTrain = table2array(data_train); %convert table to array
x = dTrain(:,1);
y = dTrain(:,2);
z = dTrain(:,3);
dLabelTrain = dTrain(:,4);
save dataTrain.mat dLabelTrain
```

Gambar 1 Open File



Gambar 2 Scatter Plot

Pada tahap selanjutnya, masing-masing klasifikasi data akan ditampung dalam array yang nantinya akan digunakan untuk proses pencarian nilai-nilai yang akan dibangun dalam PNN.

```
%label 0,1,2 di x,y,z
label0=[x(1: 46,1) y(1: 46,1) z(1: 46,1)];      %klasifikasi 0
label1=[x(47: 94,1) y(47: 94,1) z(47: 94,1)];   %klasifikasi 1
label2=[x(95: 150,1) y(95: 150,1) z(95: 150,1)];%klasifikasi 2
```

Gambar 3 Array Klasifikasi

Kemudian akan dicari jarak atau *distance* dari setiap data di setiap kelas menggunakan Euclidean Distance.

$$E(x,y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$$

Gambar 4 Eucludean Distance

```
%mencari jarak di klasifikasi/label 0
lbl0 = 0;
for i=1:length(label0),
    for j=1:length(label0),
        if i~=j,
            jarak0 = [jarak0 norm(label0(i,:)-label0(j,:))];
            tmpMin = [sort(jarak0)];
        end
    end
    lbl0 = lbl0+min(jarak0);
end
avgJarak0 = lbl0*(1/length(label0));
```

Gambar 5 Jarak tiap data di klasifikasi 0

```
%mencari jarak di klasifikasi/label 1
lbl1 = 0;
for i=1:length(label1),
    for j=1:length(label1),
        if i~=j,
            jarak1 = [jarak1 norm(label1(i,:)-label1(j,:))];
        end
    end
    lbl1 = lbl1+min(jarak1);
end
avgJarak1 = lbl1*(1/length(label1));
```

Gambar 6 Jarak tiap data di klasifikasi 1

```
%mencari jarak di klasifikasi/label 2
lbl2 = 0;
for i=1:length(label2),
    for j=1:length(label2),
        if i~=j,
            jarak2 = [jarak2 norm(label2(i,:)-label2(j,:))];
        end
    end
    lbl2 = lbl2 + min(jarak2);
end
avgJarak2 = lbl2*(1/length(label2));
```

Gambar 7 Jarak tiap data di klasifikasi 2

Setelah ditemukan, akan diambil nilai minimum dari setiap data dan dijumlahkan. Lalu, dicari nilai rata-rata dari jarak disetiap kelas untuk digunakan pada perhitungan nilai *smoothing* atau penghalus. Dengan $d_{avg}[k]$ = rata-rata jarak minimum, $|C_k|$ = banyaknya data di klasifikasi, σ_k = *smoothing*, dan g=konstanta akurasi pengklasifikasian. Pada program kali ini digunakan nilai g=0.1 yaitu sebagai standar default nilai error.

$$d_{avg}[k] = \frac{1}{|C_k|} \sum_{\rho_i \in C_k} d_i$$

Gambar 8 Rata-Rata Nilai Minimum Jarak

$$\sigma_k = g \cdot d_{avg}[k]$$

Gambar 9 Smoothing

```
g = 0.1; %default nilai error
%mencari nilai tho (smoothing)
tho0 = g*avgJarak0;
tho1 = g*avgJarak1;
tho2 = g*avgJarak2;
```

Gambar 10 Smoothing tiap klasifikasi

Selanjutnya buka file **testPNNsatudata.m** untuk melakukan test terhadap data train, tujuannya yaitu untuk mencek apakah program yang terlah dibuat sudah benar. Pertama, membuka dataTrain.mat yang berisi klasifikasi dari data training PNN. Lalu membuka data_test_PNN. Karena data dibaca dalam bentuk table, maka harus dikonversi ke array agar dapat dilakukan perhitungan probabilitas.

```
load dataTrain.mat
data_test = readtable('data_test_PNN.txt');
dTest = table2array(data_test); %convert table to array
x = dTest(:,1);
y = dTest(:,2);
z = dTrain(:,3);
```

Gambar 11 load Data Test

Akan digunakan data berikut untuk menguji program berjalan dengan baik atau tidak.

```
PTest = [[-1.23906680200000 1.68879599000000 2.33046295600000]]; %klasifikasi 2, didapat dari data train
```

Gambar 12 Data yang akan diuji

Selanjutnya, tentukan variabel m, yaitu banyaknya klasifikasi data dan nilai g=0.1 (standar default error).

```
m = 3; %banyaknya klasifikasi/label
g = 0.1; %default nilai error
```

Gambar 13 Nilai m dan g

Hitung nilai probabilitas dari setiap klasifikasi, lalu jumlahkan nilai minimum setiap probabilitas. Untuk nilai *smoothing,* telah dilakukan perhitungan pada file **trainPNN.m**.

$$p(x|C_k) \approx \frac{1}{(2\pi)^{m/2}\,\sigma_k^m\,|C_k|} \sum_{\rho_i \in C_k} \exp\left[-\|x - w_i\|^2 / \left(2\sigma_k^2\right)\right]$$

Gambar 14 Rumus PNN

```
%probabilitas klasifikasi 0
prob = [];
tmp=0;
for i=1:length(label0),
    sumE = exp(-(norm(PTest-label0(i,:)) / (2*(tho0^2))));
    tmp = tmp+sumE;
end
prob = [prob tmp/((2*pi)^(m/1))*(tho0^m)*length(label0)];
```

Gambar 15 Probabilitas klasifikasi 0

```
%probabilitas klasifikasi 1
tmp=0;
for i=1:length(label1),
    sumE = exp(-(norm(PTest-label1(i,:)) / (2*(tho1^2))));
    tmp = tmp+sumE;
end
prob = [prob tmp/((2*pi)^(m/1))*(tho1^m)*length(label1)];
```

Gambar 16 Probabilitas klasifikasi 1

```
%probabilitas klasifikasi 2
tmp=0;
for i=1:length(label2),
    sumE = exp(-(norm(PTest-label2(i,:)) / (2*(tho2^2))));
    tmp = tmp+sumE;
end
prob = [prob tmp/((2*pi)^(m/1))*(tho2^m)*length(label2)];
```

Gambar 17 Probabilitas klasifikasi 2

Dibawah ini pengklasifikasian apabila suatu data masuk ke klasifikasi 0, 1, atau 2 dengan mencari nilai probabilitas maksimum.

```
probMax = max(prob)
if prob(1,1)>0,
    Hasil = 'Klasifikasi 0'
end
if prob(1,2)>0,
    Hasil = 'Klasifikasi 1'
end
if prob(1,3)>0,
    Hasil = 'Klasifikasi 2'
end
```

Untuk data test keseluruhan, dapat dijalankan di file **testALL_PNN.m**. Secara koding hampir keseluruhan sama dengan **testPNN.m**, yang berbeda hanya pada testALL_PNN.m akan menghasilkan klasifikasi untuk data test.