# LAPORAN TUGAS 1.3

# MACHINE LEARNING

# Observasi Nilai Sigma pada *Probabilistic Neural Network* (PNN)


Disusun Oleh :

Nadine Azhalia P. (1301154519)

Kelas : IF 39-01

PRODI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

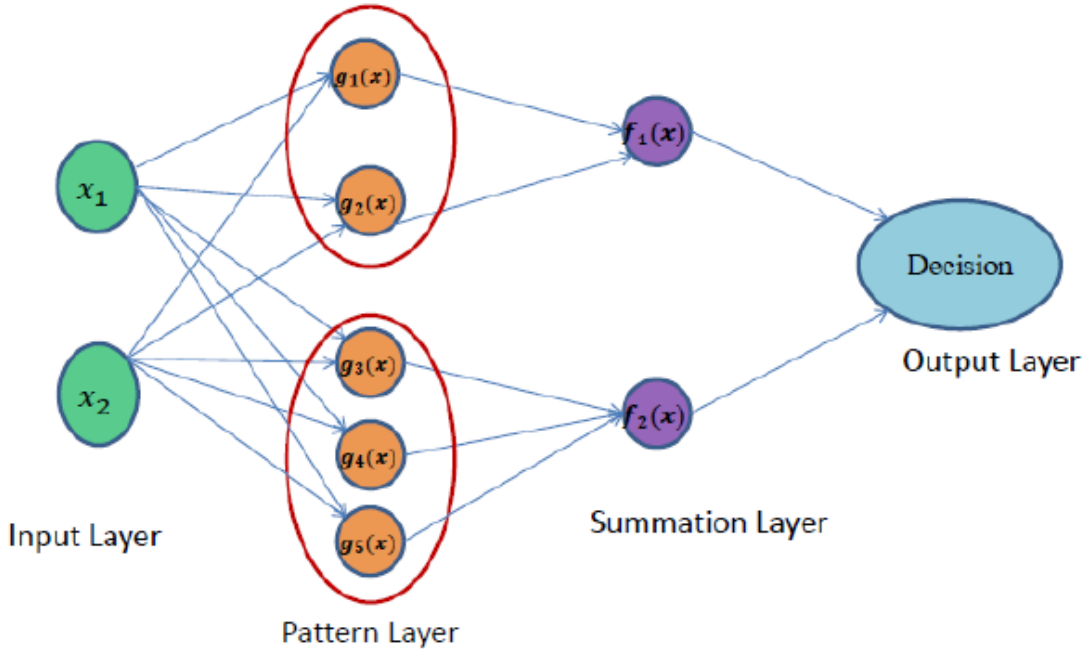
BANDUNG

2017

# 1. Probabilistic Neural Network (PNN)

Motivasi dari algoritma *Probabilistic Neural Network* (PNN) adalah mencari nilai statistik untuk mendapatkan nilai *decision boundary* yang optimal. *Decision boundary* merupakan pembatas antar kelas yang ditentukan berdasarkan likelihood atau nilai *Probability Density Function* (PDF).

Arsitektur dari algoritma PNN:




Proses *training* pada PNN dilakukan dengan mencari parameter $\boldsymbol{\sigma}$ (sigma) paling optimal yang didapatkan dari ditribusi normal. Proses pencarian nilai distribusi normal ada pada pattern layer.

Proses *testing* pada PNN akan menghitung nilai PDF dari setiap kelas dengan mengukur data baru dengan data *train* yang tersedia dari setiap kelas. Proses ini ada pada summation layer. Pada output layer akan menghasilkan kelas yang memiliki jumlah probabilitas tertinggi.

# 2. Smoothing Parameter $\sigma$

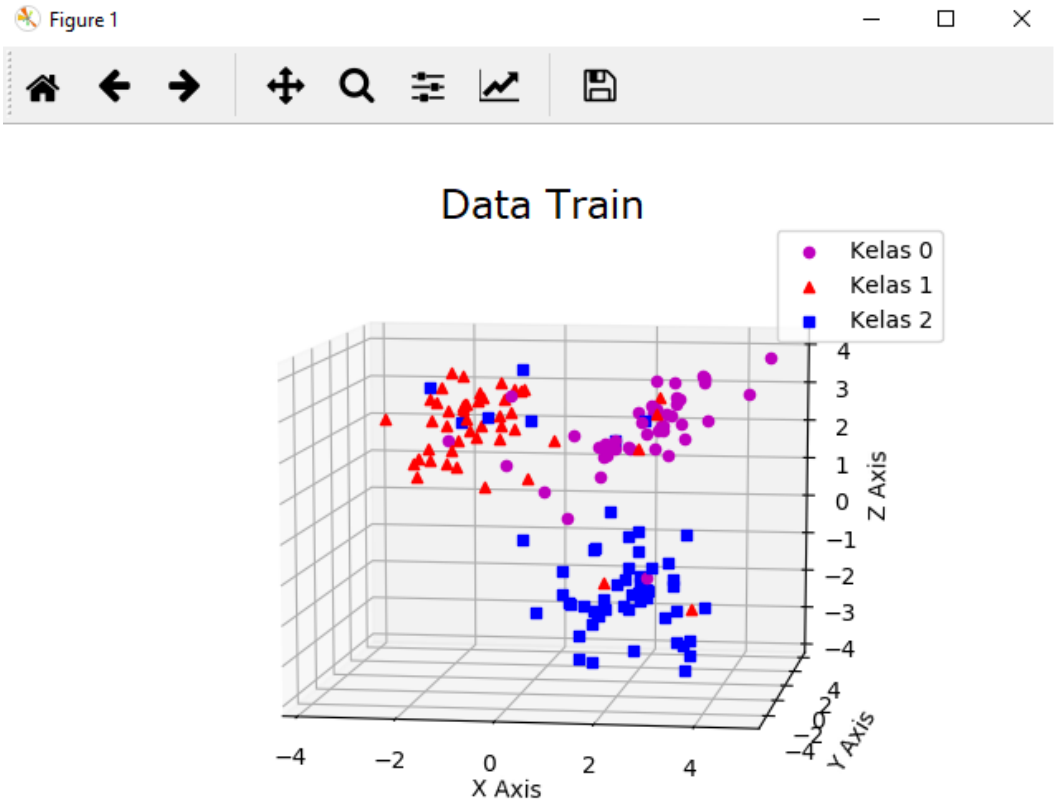
Smoothing parameter adalah cara untuk mencari nilai sigma yang optimum. Dalam kasus ini smooting parameter yang digunakan adalah cara pertama, yaitu menggunakan nilai $\sigma$ (sigma) yang sama untuk setiap kelasnya. Nilai $\sigma$ didapat dengan cara mengobservasi nilai tersebut.

Untuk mencari nilai sigma yang optimal dilakukan dengan cara melihat hasil visualisasi yang terbentuk saat nilai sigma dimasukkan. Metode *Brute Force* digunakan dalam proses pencarian nilai sigma yang optimal, karena *Brute Force* memiliki 3 tahapan dalam pemecahan masalah yang cocok diterapkan pada proses pencarian sigma. Tiga tahap tersebut diantaranya:

1. Membangkitkan semua solusi yang memungkinkan.
2. Membandingkan semua solusi tersebut.
3. Mengambil solusi yang paling optimal setelah dibandingkan.

## Proses Pembangunan Model

1. Visualisasi Data Train menggunakan *Scatter Plot*




2. Fungsi utama untuk mengklasifikasikan sebuah data menggunakan metode Jaringan Saraf Probabilistik.

- Fungsi visualisasi data

```python
# fungsi untuk memvisualkan data
def visualisasi(data):

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    for row in data:
        xs = float(row[0])
        ys = float(row[1])
        zs = float(row[2])
        if row[3] == '0':
            c = 'm'
            marker = 'o'
            scatter1 = ax.scatter(xs, ys, zs, c=c, marker=marker)
        elif row[3] == '1':
            c = 'r'
            marker = '^'
            scatter2 = ax.scatter(xs, ys, zs, c=c, marker=marker)
        else:
            c = 'b'
            marker = 's'
            scatter3 = ax.scatter(xs, ys, zs, c=c, marker=marker)
        #print (row)

    ax.set_xlabel('X Axis')
    ax.set_ylabel('Y Axis')
    ax.set_zlabel('Z Axis')
    ax.legend([scatter1, scatter2, scatter3], ['Kelas 0', 'Kelas 1', 'Kelas 2'])

    plt.legend()
    plt.show()
```

- Fungsi untuk menentukan nilai $g(x)$

```python
# fungsi untuk menghitung nilai g(x) pdf (step pattern layer)
def pdf_count(x1, x2, x3, sigma, x1_train, x2_train, x3_train):
    g = math.exp(-1*((float(x1) - float(x1_train)) ** 2 + (float(x2) - float(x2_train)) **
                     2 + (float(x3) - float(x3_train)) ** 2 / (2 * (sigma ** 2))))
    return g
```

- Fungsi untuk menentukan nilai $f(x)$

```python
# fungsi untuk menjumlahkan nilai pdf per-kelas f(x) (step summation layer)
def sum_pdf(data_train, sigma, x_test):
    sum_nol = 0
    sum_satu = 0
    sum_dua = 0
    count_nol = 0
    count_satu = 0
    count_dua = 0
    for row in data_train:
        if row[3] == '0':
            sum_nol += pdf_count(row[0], row[1], row[2],
                                 sigma, x_test[0], x_test[1], x_test[2])
            count_nol += 1

        elif row[3] == '1':
            sum_satu += pdf_count(row[0], row[1], row[2],
                                  sigma, x_test[0], x_test[1], x_test[2])
            count_satu += 1

        else:
            sum_dua += pdf_count(row[0], row[1], row[2],
                                 sigma, x_test[0], x_test[1], x_test[2])
            count_dua += 1


    sum_nol = sum_nol / count_nol
    sum_satu = sum_satu / count_satu
    sum_dua = sum_dua / count_dua
    return [sum_nol, sum_satu, sum_dua]
```

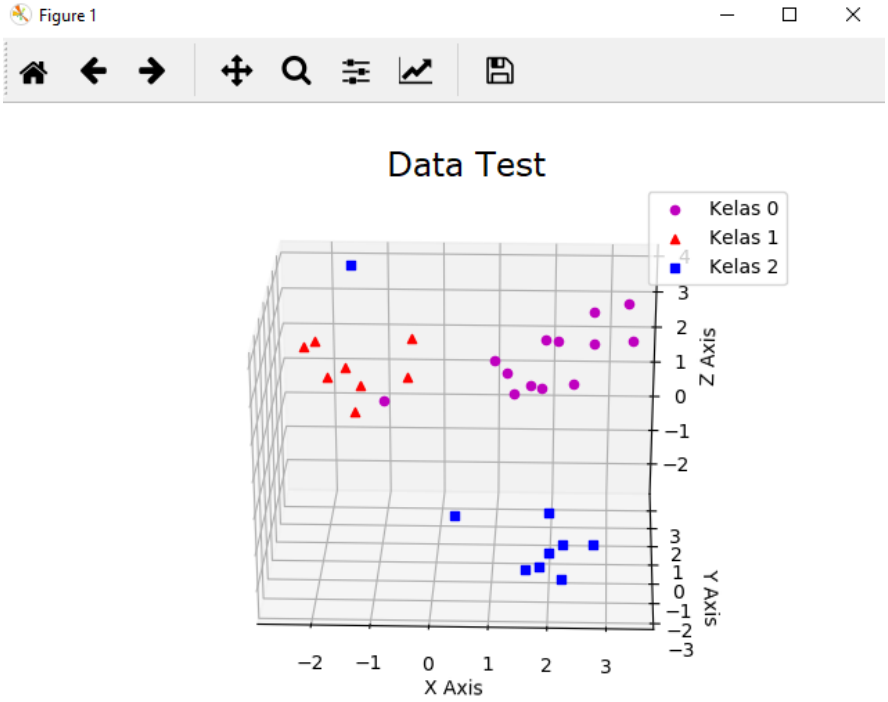
- Fungsi untuk mengklasifikasikan data sesuai dengan kelasnya

```python
# fungsi untuk mengklasifikasi hasil dari sum_pdf agar dapat menentukan kelas (output layer)
def klasifikasi(data_train, sigma, data_test):
    hasil_y = []
    for row in data_test:
        hasil_pdf = sum_pdf(data_train, sigma, row)
        print(hasil_pdf)
        if hasil_pdf[0] > hasil_pdf[1] and hasil_pdf[0] > hasil_pdf[2]:
            hasil_y.append(0)
        elif hasil_pdf[1] > hasil_pdf[0] and hasil_pdf[1] > hasil_pdf[2]:
            hasil_y.append(1)
        else:
            hasil_y.append(2)
    return hasil_y
```

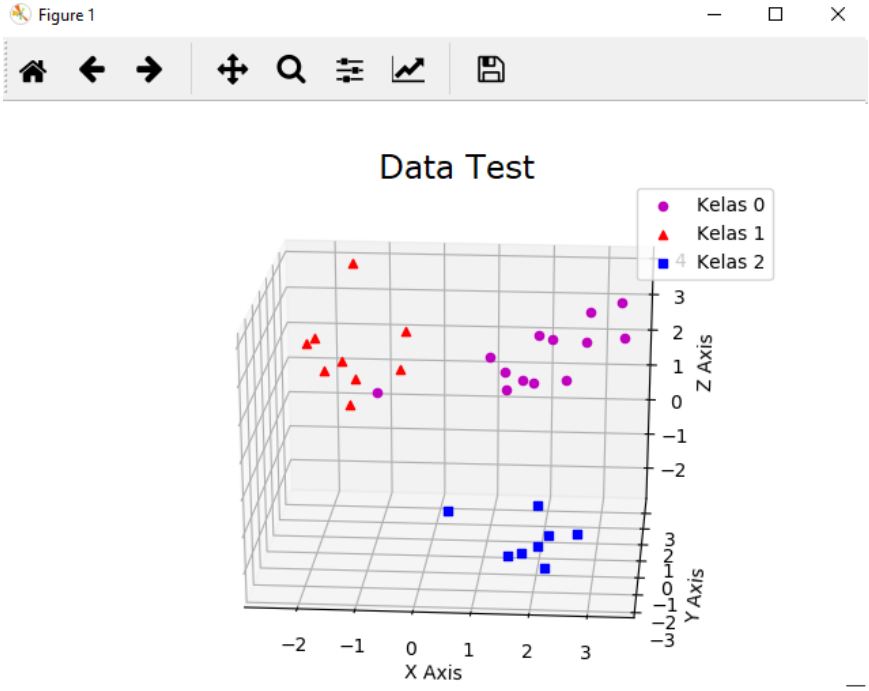
3. Observasi untuk menentukan parameter-parameter terbaik yang akan digunakan di proses pengujian

Untuk mendapatkan nilai sigma yang optimal, dilakukan dengan cara *Brute Force*. Mencoba nilai sigma secara random dan dilihat visualisasinya sampai sesuai dengan kelasnya.

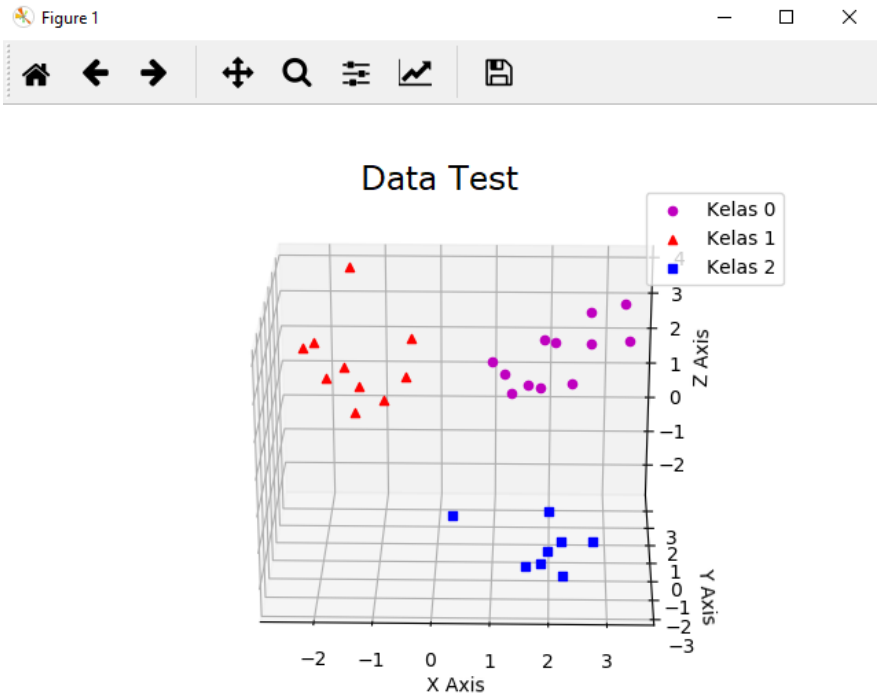
**Sigma = 0.1**



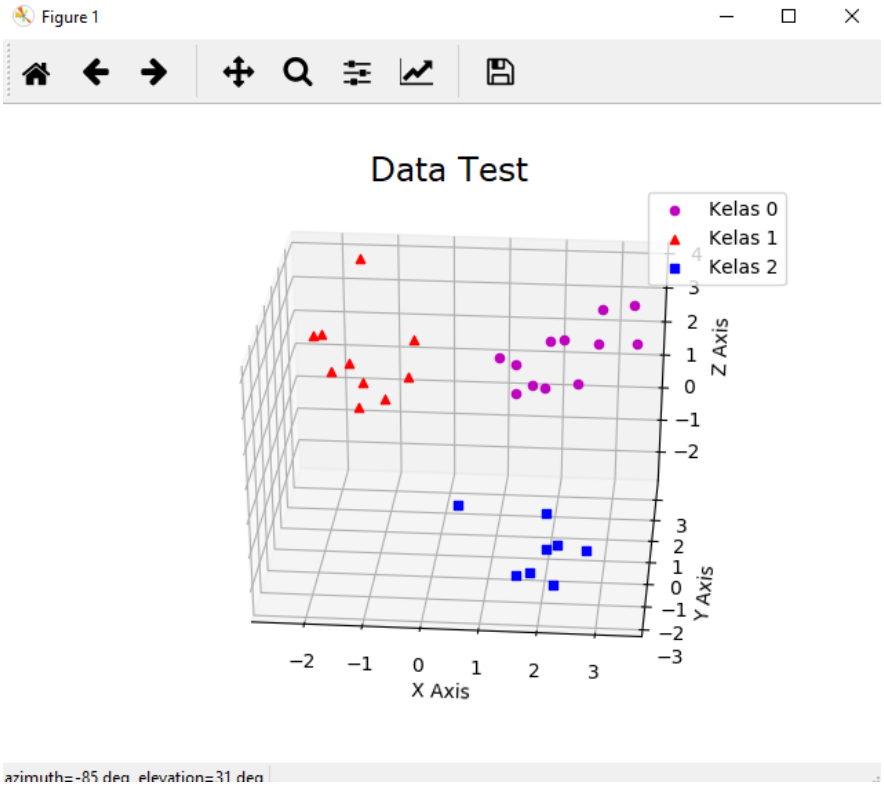

**Sigma = 0.2**



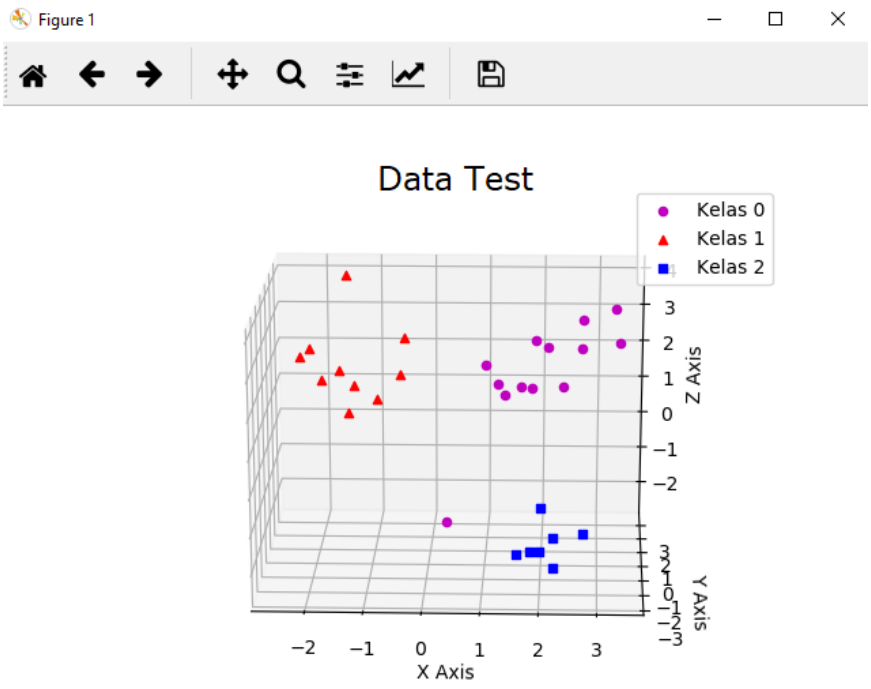
**Sigma = 0.4**




**Sigma = 0.8**




**Sigma = 1.8**



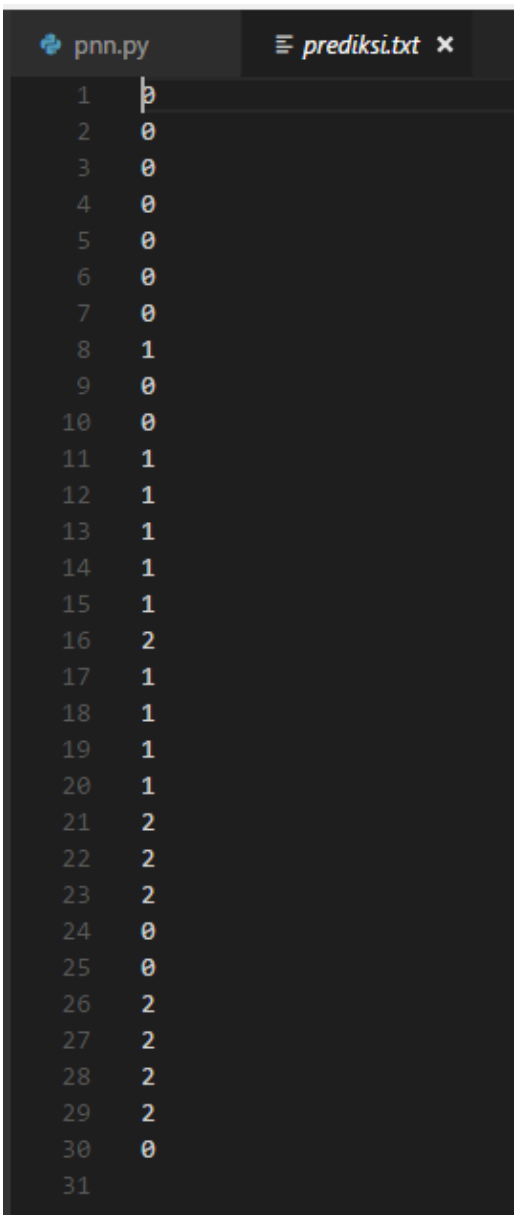

Maka, nilai sigma yang optimal untuk kasus ini sebesar $\boldsymbol{\sigma}$ **= 0.8**. karena dapat dilihat pada hasil visualisasinya jika kita menggunakan sigma sebesar 0.8 data akan terkumpul sesuai kelasnya.

## Proses Pengujian

Setelah mendapat nilai sigma yang optimal yaitu $\boldsymbol{\sigma}$ = 0.8, maka nilai sigma tersebut dimasukkan ke fungsi klasifkasi. Tujuannya adalah untuk mendapatkan hasil klasifikasi untuk data test. Atau nilai y yang ada pada data test.

```
98   # mengobservasi nilai sigma agar mendapat hasil yg optimal
99   y_hasil = klasifikasi(data_train, 0.5, data_test)
100  print (y_hasil)
101
102  # memasukkan nilai y_hasil ke list data_test
103  for i in range(len(data_test)):
104      data_test[i].append(str(y_hasil[i]))
105  print (data_test)
106
```

Hasil klasifikasi (prediksi.txt):


pnn.py
prediksi.txt
0
0
0
0
0
0
0
1
0
0
1
1
1
1
1
2
1
1
1
1
2
2
2
0
0
2
2
2
2
0