Laporan Observasi
Nilai Sigma pada *Probability Neural Network* (PNN)
Studi Kasus: Tugas 1.3 Mata Kuliah Pembelajaran Mesin pada Univeritas Telkom

Compiled by: Aditya Alif Nugraha (1301154183)

Visualisasi Data Train

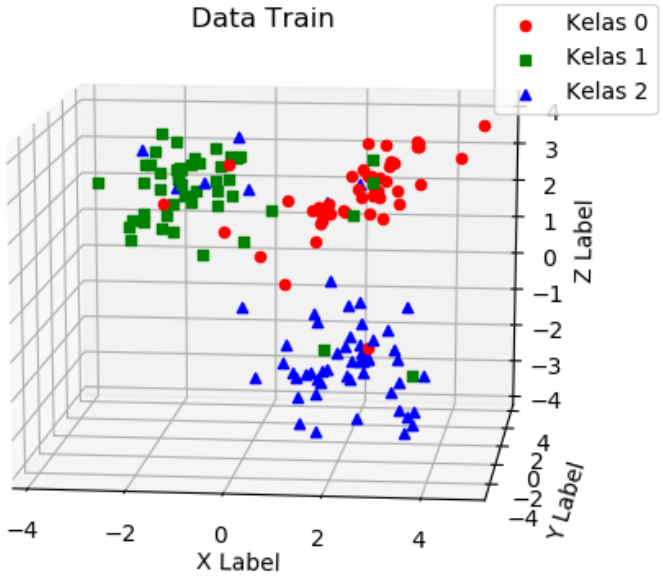


Figure 1 Visualisasi Data Train

Observasi Sigma (Smoothing Parameter) σ

Sigma (*smoothing parameter*) adalah parameter yang penting dalam PNN. PNN memerlukan nilai sigma yang tepat agar dapat mengklasifikasikan data dengan baik. Nilai sigma yang terlalu kecil akan mengakibatkan seolah-olah nilai tiap titik/variabel menjadi berbeda/tidak berhubungan. Sedangkan, nilai sigma yang terlalu besar akan membuat nilai PDF seperti distribusi Gaussian.

Ada beberapa metode untuk mencari nilai sigma yang optimal. Salah satunya yaitu Jackknifing. Penerapan metode Jackknifing yaitu sebagai berikut:

1. Mencoba nilai sigma dalam interval yang ditentukan
2. Menentukan interval baru dan mempersempit *range* pencarian dari hasil percobaan yang sebelumnya
3. Ulangi langkah 1 dan 2 hingga menemukan sigma yang dianggap optimal

Sebelum melakukan Jackknifing, dilakukan terlebih dahulu *train test split*. Dimana dataset pada data *train* dibagi menjadi 2 bagian, *train set* dan *validation set*. Hal tersebut dilakukan agar dapat dihitung akurasi dari prediksi kelas pada *validation set.*

Dari hasil pencarian sigma dengan metode Jacknifing yang diterapkan pada kasus Tugas 1.3 Mata Kuliah Pembelajaran Mesin pada Univeritas Telkom, maka dihasilkan visualisasi pencarian seperti pada *Figure 2*.

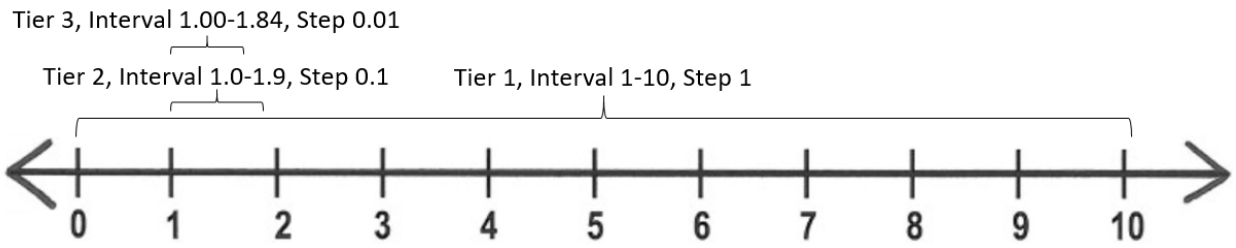
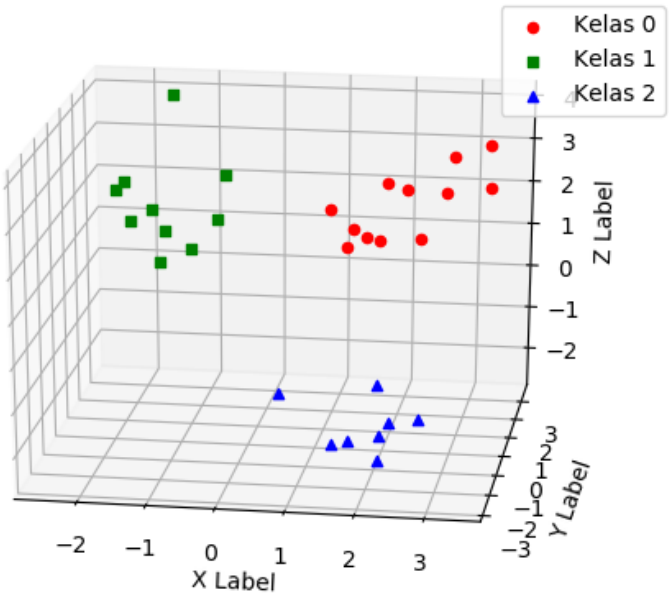


Figure 2 Pencarian Sigma

Pada *tier 1*, ditentukan interval awal dari 0 hingga 10 dengan *range* pencarian 1. Kemudian didapatkan akurasi terbaik pada nilai sigma 1 hingga 1,9. Maka akan dibuat interval pencarian sigma baru dari 1,0 hingga 1,9. Interval tersebut akan digunakan pada *tier* 2 dan dihasilkan akurasi terbaik pada interval 1,00 hingga 1,84. Maka sigma yang digunakan yaitu nilai tengah antara 1,00 dan 1,84. Sehingga didapat nilai sigma yang optimal sebesar 1,42.

Setelah didapatkan sigma dari metode Jackknifing, sigma tersebut digunakan untuk memprediksi data test. Saat diterapkan, model tersebut dapat mengklasifikasikan data dengan baik. Terlihat pada visualisasi pada *Figure 3*.

Figure 3 Hasil Klasifikasi pada Data Test



Dari gambar diatas, dapat disimpulkan bahwa kelas pada data test yang diprediksi sudah berhasil terpisah sesuai dengan kelasnya. Hal tersebut menandakan bahwa nilai sigma sebesar 1,42 dapat mengklasifikasikan data dengan baik.

Implementasi Fungsi Probability Neural Network (PNN)

Fungsi Membuka File

```
def open_csv_file(filename, set_type=""):
    # Soal A nomer 1
    """
    Open csv file return X and/or y based on choosen set type.
    Set type must: "train" or "test".
    """
    df = pd.read_csv(filename, header=None)

    if set_type == "train":
        # rand_val = random.randint(0, 100)
        # print(rand_val)
        df = df.sample(frac=1, random_state=77).reset_index(
            drop=True)  # Mengacak data train
        X = df.iloc[:, 0:-1].values
        y = df.iloc[:, -1].values
        return X, y
    elif set_type == "test":
        X = df.iloc[:, :].values
        return X
    else:
        raise TypeError("Err: set_type not available")
```

Fungsi Membagi Data Train Menjadi *Train Set* dan *Test Set*

```
def train_validation_split(X, y, train_size):
    """
    Split the dataset to train and validation set.
    This function return X_train, y_train, X_validation, y_validation.
    """
    size_limit = int(len(X) * train_size)
    X_train = X[0:size_limit, :]
    y_train = y[0:size_limit]
    X_validation = X[size_limit:len(X) + 1, :]
    y_validation = y[size_limit:len(X) + 1]

    return X_train, y_train, X_validation, y_validation
```

Fungsi Visualisasi

```
def scatter3d_visualize(X, y, title=""):
    # Soal A nomer 1
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    for i in range(len(X)):
        xs = X[i][0]
        ys = X[i][1]
        zs = X[i][2]

        if y[i] == 0:
            bundar = ax.scatter(xs, ys, zs, c="r", marker="o", label="Kelas 0")
        elif y[i] == 1:
            kotak = ax.scatter(xs, ys, zs, c="g", marker="s", label="Kelas 1")
        elif y[i] == 2:
            segitiga = ax.scatter(
                xs, ys, zs, c="b", marker="^", label="Kelas 2")

    ax.set_xlabel('X Label')
    ax.set_ylabel('Y Label')
    ax.set_zlabel('Z Label')
    plt.legend(handles=[bundar, kotak, segitiga])
    plt.title(title)
    plt.show()
```

Fungsi Menghitung Nilai PDF

```
def count_pdf(X, y, X_input, sigma):
    """Fungsi untuk menghitung PDF pada PNN dengan input X, y, X input, dan sigma.
    Fungsi ini mengembalikan nilai PDF untuk semua kelas terurut sesuai index."""
    n = len(X)
    # print(X_input)
    sum_gx = [0 for i in range(max(y) + 1)]
    # print(sum_gx)
    count_0 = 0
    count_1 = 0
    count_2 = 0
    # fx = (1/len(X)) * sum(math.exp(-1*((sum(abs(x_input-X)) ** 2))/2*(sigma**2)))
    for i in range(n):
        sum_gx[y[i]] += math.exp(-1 * (((X_input[0] - X[i][0])**2 + (X_input[1] - X[i][1])
                                       ** 2 + (X_input[2] - X[i][2])**2) / (2 * (sigma**2))))  # menghitung gx
        if y[i] == 0:
            count_0 += 1
        if y[i] == 0:
            count_1 += 1
        else:
            count_2 += 1

    # mengembalikan nilai PDF
    return [sum_gx[0] / count_0, sum_gx[1] / count_1, sum_gx[2] / count_2]
```

Fungsi untuk Klasifikasi

```
def classification(X_train, y_train, X_input, sigma):
    """Fungsi ini untuk memprediksi kelas dari input X_train, y_train, X_input, dan sigma."""
    y_pred = []
    for X in X_input:
        # print(sigma)
        pdf = count_pdf(X_train, y_train, X, sigma)
        y_pred.append(pdf.index(max(pdf)))
    return y_pred
```

Fungsi Menghitung Akurasi

```
def countAccuracy(y_real, y_pred):
    acc = 0
    n = len(y_real)
    for i in range(n):
        if y_real[i] == y_pred[i]:
            acc += 1
    return acc / n
```

Fungsi Membuat File Output

```
def create_output_file(filename, y_answer):
    f = open(filename, "w+")
    f.truncate()
    f.write(str(y_answer))
    f.close()
```

Fungsi Observasi Sigma dengan Metode Jackknifing

```
def observe_sigma(start, end, X_train, y_train, X_validation, y_validation):
    """Fungsi observasi Sigma secara otomatis.
    Mencari nilai Sigma dengan interval dari nilai start-end yang diinputkan."""

    # Tier 1 - Mencari Sigma pada range start - end dengan step 1
    sig = []
    for i in range(start, end):
        if i == 0:
            i = 0.1
        # print(i)
        sig.append(countAccuracy(y_validation, classification(
            X_train, y_train, X_validation, i)))

    # Tier 2 - Mencari Sigma pada range yang memiliki accuracy terbaik dengan step 0,1
    start_tier2 = sig.index(max(sig))
    i = start_tier2
    end_tier2 = start_tier2 + sig.count(max(sig))
    # print(start_tier2, end_tier2)
    sig = []
    while i <= end_tier2:
        if i == 0:
            i = 0.1
        sig.append(countAccuracy(y_validation, classification(
            X_train, y_train, X_validation, i)))
        i += 0.1
    # print(sig)

    # Tier 3 - Mencari Sigma pada range yang memiliki accuracy terbaik dengan step 0,01
    start_tier3 = start_tier2 + sig.index(max(sig)) / 10
    i = start_tier3
    end_tier3 = start_tier3 + sig.count(max(sig)) / 10
    sig = []
    while i <= end_tier3:
        if i == 0:
            i = 0.1
        sig.append(countAccuracy(y_validation, classification(
            X_train, y_train, X_validation, i)))
        i += 0.01

    start_best_sigma = start_tier3 + (sig.index(max(sig)) / 100)
    end_best_sigma = start_best_sigma + (sig.count(max(sig)) / 100)
    # print(start_best_sigma, end_best_sigma)
    # Mengembalikan nilai sigma terbaik
    return (start_best_sigma + end_best_sigma) / 2
```