UNIVERSITAS INDONESIA

SISTEM BERBASIS PENGETAHUAN
LAPORAN PROYEK JARINGAN SYARAF TIRUAN MENGGUNAKAN ALGORITMA PROPAGASI BELAKANG

Hansel Matthew (1806194914)

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO

DEPOK
APRIL 2021

DAFTAR ISI

BAB I
PENDAHULUAN

2.1 Latar Belakang

Sistem kendali adalah cabang ilmu yang fokus mempelajari pengendalian respon dari sebuah sistem. Sistem tersebut dapat datang dari banyak ruang lingkup seperti pada sistem proses kimia, produksi tenaga listrik, bidang manufaktur, produksi minyak dan gas, dan masih banyak lagi. Sistem kendali pertama yaitu sistem kendali klasik PID dikembangkan pada tahun 1940 untuk pengendali dalam proses industri. 90% dari lup pengendalian menggunakan pengendali PID (Proporsional – Integral – Diferensial). Sistem kendali klasik ini Sebagian besar masih bekerja secara manual dengan parameter ditentukan secara trial and error. Penalaan ulang dari parameter ini menghabiskan waktu yang lama serta memakan biaya yang tinggi. Sering sekali industri menggunakan pengendali dengan nilai parameter *default* sehingga memberikan kinejar yang tidak sama untuk perubahan dari titik operasi.

Seiring berkembangnya zaman, sistem kendali menjadi semakin dibutuhkan, hal itu dikarenakan kemampuan manusia tidak dapat lagi mengalahkan kemampuan mesin dalam hal mengendalikan dengan akurat dan presisi. Sistem kendali pun mengalamni beberapa perkembangan menjadi berbagai jenis seperti optimal control, robust control, non linear control, dan lain lain. Salah satu perkembangannya adalah intelligent control, yaitu suatu pengendali adaptif yang berbasis pengetahuan. Terciptalah suatu sistem yang memiliki cara kerja mirip dengan sistem kerja jaringan saraf manusia yang dinamakan *Artificial Neural Network. Artificial Neural Network*. *Artificial Neural Network* (ANN) atau Jaringan Saraf Tiruan adalah suatu sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Sistem ini tidak menggunakan sebuah pemodelan matematis karena menggunakan pemodelan *blackbox*. Pemodelan ini dilakukan dengan cara menganalisa perilaku sistem jika diberikan variasi pasangan input dan output. Dari perilaku inilah maka jaringan saraf tiruan akan berusaha untuk menghasilkan output yang diinginkan.

2.2 Rumusan Masalah

a. Bagaimana konsep dasar jaringan saraf tiruan?
b. Bagaimana cara kerja metode pembelajaran jaringan saraf tiruan dengan menggunakan algoritma *backpropagation*?
c. Bagaimana suatu jaringan saraf tiruan dapat mengklasifikasikan suatu dataset?

2.3Tujuan Penelitian

Berdasarkan permasalahan yang telah dipaparkan, maka tujuan dari laporan ini adalah untuk menganalisis dan mempelajari cara kerja dari jaringan saraf tiruan pada dataset yang diberikan.

BAB II
DASAR TEORI

2.1Artificial Neural Network

Artificial Neural Network (ANN) atau Jaringan Saraf Tiruan lahir dari usaha manusia yang ingin memodelkan otak manusia untuk membuat suatu sistem yang paling sempurna. Model ANN pertama kali dikenalkan oleh Mc. Culloh dan Pitts sebagai komputasi dari aktivitas syarat. Hasil penelitian mereka menjadi dasar bagi penelitian di bidang ANN pada masa berikutnya.

Pada tahun 1958, Rosenblat, Widrow dan Hoff pertama kali menemukan aturan pembelajaran pada perceptrons. Minski dan Papert (1969) meyakini bahwa penggunaan perceptrons sangat terbatas yaitu hanya sebagai metode perhitungan dalam kehidupan nyata. Bernard Widrow menemukan neuron sederhana yang mirip dengan perceptron yang disebut ADALINE (neuron linier adaptif), dan jaringan multilayernya yang disebut MADALINE (multiple adaline). Selanjutnya, Widrow juga mengembangkan program pembelajaran terbimbing yang disebut dengan metode pembelajaran Least Mean Square (LMS) atau Widrow Hoff. Di era berikutnya, jaringan syaraf tiruan berkembang sedemikian rupa sehingga menemukan berbagai metode pembelajaran dan aturan pembelajaran.

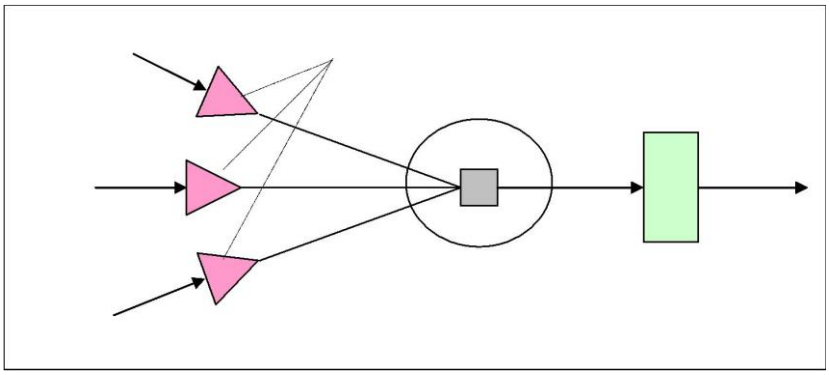
ANN merupakan jaringan yang dibuat dengan meniru jaringan syaraf manusia dengan diilhami oleh struktur dan cara kerja otak dan sel syaraf manusia. Sebuah neuron memiliki sebuah badan sel, pengirim sinyal dan penerima sinyal. Neuron biologis pada manusia dapat dilihat pada gambar 2.1

Gambar 2.1. Ilustrasi Neuron Biologis Manusia

Cara kerja dari sebuah neuron adalah akan bereaksi apabila potensial listrik mencapai suatu batasan tertentu. Neuron akan menjumlahkan sinyal yang masuk melalui dendrite yang dikalikan dengan pembobot sinapsis[1]. Proses pembelajaran terjadi dengan perubahann yang

terjadi pada sinasis. Sinyal yang masuk akan dijumlahkan dan dikonversi dengan suatu fungsi aktivitas yang kemudian akan mengeluarkan suatu sinyal pemicu yang dialirkan ke neuron lain.

Prinsip kerja dari neuron ini kemudian dimodelkan secara matematis. Model matematik orde pertama dari neuron dapat dilihat pada Gambar 2.2



Gambar 2.2. Model Matematik dari Neuron

Pemodelan matematika inilah yang menjadi dasar dari ANN. Elemen dasar dari sebuah ANN adalah sebuah neuron. Neuron ini akan mengubah sinyal msukan menjadi sebuah keluaran. Setiap neuron mempunyai suatu inputan yang memiliki bobotnya masing – masing. Sinyal kemudian akan dikonversi menggunakan sebuah fungsi aktivasi. Fungsi aktivasi yang digunakan dapat beragam seperti *Relu, sigmoid, tanh* dan lain lain

2.2 *Backpropagation*

Backpropagation adalah suatu algoritma ANN yang memiliki kekuatan utama pada klasifikasi suatu pola atau disebut sebagai *pattern recognition*. Jaringan syaraf *backpropagation* dapat digunakan untuk memprediksi luaran dari sebuah sistem. Pada mulanya data akan diberikan pada sistem. Jika sistem menghasilkan luaran yang tidak sesuai dengan yang diharapkan, maka *backpropagation* akan memodifikasi bobot pada hubungan neuron.[2]

Backpropagation menggunakan gradient descent untuk mengoptimasi nilai dari bobot tersebut. Gradient descent bertujuan adalah untuk terus mengambil sampel gradien parameter model ke arah yang berlawanan berdasarkan bobot w serta memperbarui secara konsisten hingga ANN dapat mencapai fungsi minimum global J (w). Gradient descent memiliki suatu parameter yaitu Learning Rate (α) yang mengatur seberapa besar perubahan bobot pada setiap epoch.

Pada gradient descent dikenal juga istilah momentum. Momentum adalah suatu metode agar mempertahankan arah dari gradient descent. Hal ini diperoleh dengan mengkombinasikan

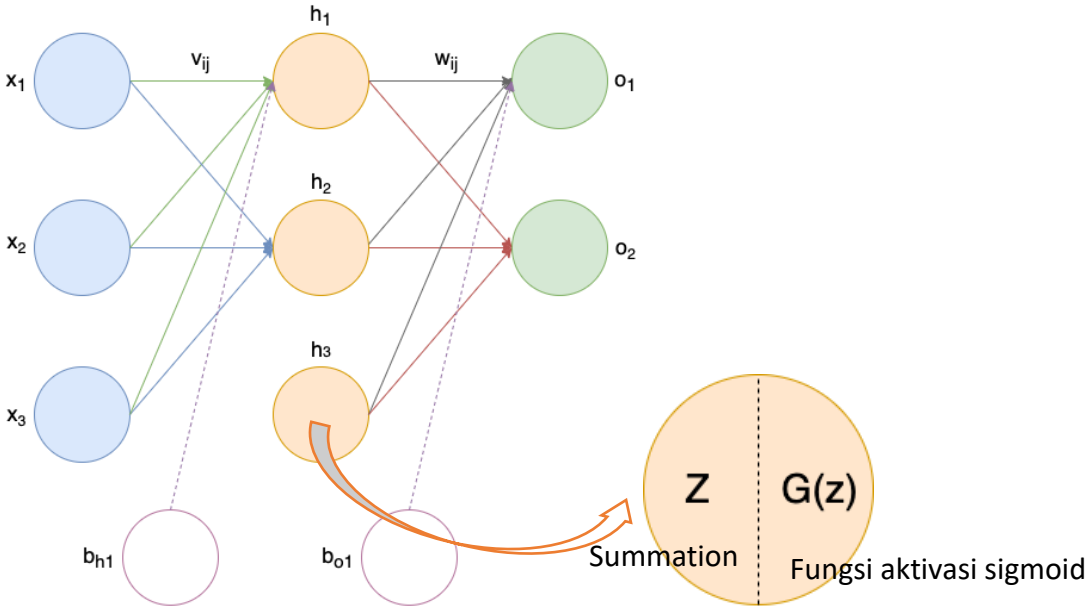
arah yang dikomputasi pada iterasi sekarang dengan iterasi sebelumnya. Besar pengaruh arah iterasi sebelumnya dapat diatur dengan mengatur koefisien momentum (μ**).**

ANN terdiri dari 3 layer, yaitu *input layer* yang memiliki neuron sebanyak I buah, *hidden layer* yang memiliki neuron sebanyak J buah, dan *output layer* yang memiliki neuron sebanyak K buah. Antara layer dihubungkan dengan bobot. Antara *input layer* dengan *hidden layer* dihubungkan dengan bobot **w**. Antara layer *hidden layer* dengan *output layer* dihubungkan dengan bobot **v**. Pada layer input dan hidden ditambah sebuah neuron bobot bias, yang berguna sebagai konstanta.[3]

Algoritma *backpropagation* dapat dibagi menjadi dua buat proses yaitu training dan testing. Dimana kedua proses tersebut dapat dibagi lagi menjadi:

1. Training
   a. Proses pengolahan data input (*feedforward*).
   b. Perhitungan error (*backpropagation*).
   c. Pembaruan bobot.
2. Testing
   a. Pengelompokkan nilai output (*Quantizing*).
   b. Perhitungan *Recognition Rate*.

Diagram sederhana pemodelan jaringan saraf tiruan yang digunakan dapat dilihat pada Gambar 2.3.



Gambar 2.3. Struktur Neural Networks Sederhana

Keterangan:

∂ x → Input layer ∂ h → Hidden layer

∂ o → Output layer

∂ i → Ukuran input layer

∂ j → Ukuran hidden layer

∂ v → Bobot hidden layer

∂ w → Bobot output layer

Pada laporan ini, digunakan data Teaching Assistant Evaluation pada *https://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation*[4] dengan jumlah *instance* sebanyak 151 data. Ada 5 buah objek yang diamati, yaitu

1. Apakah asisten pengajar merupakan penutur asli bahasa Inggris (biner); 1 = penutur bahasa Inggris, 2 = penutur non-bahasa Inggris
2. Instruktur kursus (kategorikal, 25 kategori)
3. Kursus (kategorikal, 26 kategori)
4. Semester regular atau musim panas (biner) 1 = Musim Panas, 2 = Regular
5. Ukuran kelas (numerik)

Kombinasi ini membuat matrix input memiliki ukuran 151x50. Dari 151 data, data tersebut dipecah dengan perbandingan 70:30 dimana 105 data akan digunakan untuk pembelajaran dan 46 data sisanya akan digunakan untuk proses *testing*.

Berikut merupakan algoritma pemrograman *backpropagation*:

A. Inisialisasi bobot

Inisialisasi awal dari tiap bobot hubungan antar neuron dapat dilakukan menggunakan dua buah metode yaitu random dan nguyen widrow.

1. Random

Metode inisialisasi random adalah metode dimana bobot awal ditentukan secara acak. Hal ini dapat kita lakukan menggunakan fungsi random pada MATLAB. Inisialisasi awal yang acak dibutuhkan untuk menghasilkan symmetry breaking. Suatu strategi yang efektif untuk diterapkan pada inisialisasi random adalah menentukan batas atas dan batas bawah dari nilai acak tersebut. Hal ini dilakukan untuk memastikan bahwa bobot neuron tidak terlalu besar sehingga proses pembelajaran lebih efisien.

Berikut merupakan kode MATLAB untuk inisialisasi random:

```
%Random Init
epsilon_init = 0.5;
weight_xz = rand(x_size, z_size) * 2 * epsilon_init - epsilon_init;
weight_zy = rand(z_size, y_size) * 2 * epsilon_init - epsilon_init;
bias_xz = rand(1,z_size) * 2 * epsilon_init - epsilon_init;
bias_zy = rand(1,y_size) * 2 * epsilon_init - epsilon_init;
```

2. Nguyen Widrow

Metode nguyen widrow adalah sebuah algoritma modifikasi sederhana dari bobot yang mampu meningkatkan kecepatan proses pembelajaran. Algoritma nguyen-widrow adalah sebagai berikut:

- Menentukan besar faktor skala (β)

  $\beta = 0{,}7 * J^{\frac{1}{I}}$

  Dengan: J = Ukuran *hidden layer*; I = Ukuran *input layer*.
- Inisialisasi bobot secara random pada rentang -0.5 sampai 0.5
- Menghitung norm dari vektor bobot

  $\|v_j\| = \sqrt{\sum_{i=1}^{I} v_{ij}^2}$ dan $\|w_k\| = \sqrt{\sum_{j=1}^{J} w_{jk}^2}$

- Memperbaharui bobot

  $v_{ij} = \frac{\beta v_{ij}}{\|v_j\|}$ $w_{jk} = \frac{\beta w_{jk}}{\|w_k\|}$
- Mengatur bias sebagai bilangan acak antara β sampai - β

  Berikut merupakan kode MATLAB untuk inisialisasi nguyen-widrow:

```
%Nguyen Widrow
beta = 0.7 * x_size^(1/z_size);
%Input->Hidden
for i = 1:z_size
    norm(i) = sqrt(sum(weight_xz(:,i).^2));
    weight_xz(:,i) = beta*((weight_xz(:,i))/norm(i));
end

bias_xz = rand(1, z_size) * 2 * beta - beta;
bias_zy = rand(1, y_size) * 2 * beta - beta;
```

B. Data Preprocessing

Preprocessing pertama adalah normalisasi data yaitu proses mengubah data pada suatu feature agar mempunyai rentang yang sama yaitu diantara 0-1. Hal ini dilakukan agar seluruh feature mempunyai rentang yang sama sehingga tidak ada feature yang mendominasi satu dengan yang lainnya

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Berikut merupakan kode MATLAB untuk normalisasi data:

```
%% Normalisasi
for m = 1 : feature_row
    for n = 1 : feature_col
       feature_norm(m,n) = ((feature(m,n) -
min(feature(:,n)))/(max(feature(:,n)) - min(feature(:,n))));
    end
end
```

C. Algoritma Pembelajaran

Algoritma untuk proses pembelajaran adalah sebagai berikut:

Selama kondisi stopping FALSE, maka untuk setiap pasangan pelatihan:

Proses *forward pass*

1. Komputasi *Input Layer*. Setiap x_i, i = 1,2,...,I
   - Menerima input x_i
   - Mengirimkannya ke semua *Hidden Layer* diatasnya.
2. Komputasi *Hidden Layer*. Setiap h_j, j = 1,2,...,J
   - $z\_in_j = v_{0j} + \Sigma x_{pi} v_{ij}$
   - $z_j = \frac{1}{1+e^{-z\_in_j}}$
3. Komputasi *Output Layer*. Setiap o_k, k = 1,2,...,K
   - $o\_in_k = w_{0k} + \Sigma z_j w_{jk}$
   - $o = \frac{1}{1+e^{-y\_in_k}}$
   - Kuantisasi output
     - 0 <= O_k < 0.3 → O_k = 0
     - 0.7 < O_k <= 1 → O_k = 1
     - Else → O_k = O_k.

Pada layer output diterapkan suatu threshold untuk membantu proses pembelajaran. Hal ini dilakukan karena sebuah output yang bernilai 1 atau 0 pada neuron output nyaris mustahil untuk tercapai tanpa adanya fungsi matematika tambahan. Threshold yang dipilih adalah 0.7 untuk batas atas dan 0.3 untuk batas bawah. Sehingga output yang bernilai lebih besar dari 0.7 akan dikonversi menjadi nilai biner 1 dan output yang bernilai lebih kecil dari 0.3 akan dikonversi menjadi nilai biner 0

Proses *backward pass*

1. Komputasi di *Output Layer*. Setiap o_k, k = 1,2,...,K
   - Menghitung informasi error:

$$\delta_k = (t_{pk} - y_{pk}).f'(y\_in_k)$$

- Menghitung besarnya koreksi bobot *output layer*:

  $$\Delta w_{jk} = \alpha \delta_k z_j$$

- Menghitung besarnya koreksi bias output:

  $$\Delta w_{0k} = \alpha \delta_k$$

4. Komputasi di *Hidden Layer*. Setiap h_j, j = 1,2,...,J

   - Menghitung semua koreksi error:

     $$\delta\_in_j = \Sigma \delta_k w_{jk}$$

   - Menghitung nilai aktivasi koreksi error :

     $$\delta_j = \delta_{in_j} f'(z\_in_j)$$

   - Menghitung koreksi bobot unit hidden :

     $$\Delta v_{ij} = \alpha \delta_j x_i$$

   - Menghitung koreksi error bias unit hidden :

     $$\Delta v_{0j} = \alpha \delta_j$$

Proses memperbaharui bobot

1. Bobot ke *hidden layer*:

   $$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$$

   $$v_{0j}(baru) = v_{0j}(lama) + \Delta v_{0j}$$

2. Bobot ke *output layer*:

   $$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$$

   $$w_{0k}(baru) = w_{0k}(lama) + \Delta w_{0k}$$

Proses perhitungan error

Error dihitung dengan menggunakan fungsi kesalahan fungsi error kuadratis:

$$E = \frac{1}{2}\Sigma(t_{pk} - y_{pk})^2$$

Berikut merupakan kode MATLAB untuk threshold data output:

```
%Threshold
for s=1:3
    if y(n,s) >= 0.7
        y(n,s) = 1;
    end
    if y(n,s) <= 0.3
        y(n,s) = 0;
    end
end
```

Program MATLAB proses pembelajaran

```
clear;
close all;
%% Data Input
load('dataset.mat')
feature = dataset(1:105,1:5);
feature = feature{:,:};
class = dataset(1:105,6:8);
class = class{:,:};

[feature_row,feature_col] = size(feature);
[class_row,class_col] = size(class);
%% Normalisasi
feature_norm = zeros(size(feature));
for m = 1 : feature_row
    for n = 1 : feature_col
       feature_norm(m,n) = ((feature(m,n) -
min(feature(:,n)))/(max(feature(:,n)) - min(feature(:,n))));
    end
end
%% Hyperparameter & Declare Variable

%Hyperparameter
x_size  = feature_col; %Input layer
z_size = 7;  %Hidden layer
y_size = class_col; %Output layer

alpha = 0.1;
epoch = 2000;
miu = 0.1;

%Variable
stop = 0;
error_target = 0.1;
data_count = feature_row;
epoch_count = 1;

delta_zy_old = 0;
delta_xz_old = 0;
delta_zy_bias_old = 0;
delta_xz_bias_old = 0;
%% Initialisasi

%Random Init
rng(2) %Seed
epsilon_init = 0.5; %Range random number

%Randomize weight and bias
weight_xz = rand(x_size, z_size) * 2 * epsilon_init - epsilon_init;
weight_zy = rand(z_size, y_size) * 2 * epsilon_init - epsilon_init;
bias_xz = rand(1,z_size) * 2 * epsilon_init - epsilon_init;
bias_zy = rand(1,y_size) * 2 * epsilon_init - epsilon_init;

%Nguyen Widrow
beta = 0.7 * z_size^(1/x_size);

for i = 1:z_size
    norm(i) = sqrt(sum(weight_xz(:,i).^2));
    weight_xz(:,i) = beta*((weight_xz(:,i))/norm(i));
end
```

```
bias_xz = rand(1, z_size) * 2 * beta - beta;
bias_zy = rand(1, y_size) * 2 * beta - beta;
%% Backpropagation
while stop == 0 && epoch_count <= epoch
    for n = 1:data_count
        %Forward Pass
        %Input -> Hidden
        x = feature(n,:);
        t = class(n,:);

        z_in = bias_xz + x*weight_xz;

        for m=1:z_size
            z(1,m) = 1/(1+exp(-z_in(1,m)));
        end

        %Hidden -> Output
        y_in = bias_zy + z*weight_zy;

        for l=1:y_size
            y(n,l) = 1/(1+exp(-y_in(1,l)));
        end

        %Threshold
        for s=1:3
            if y(n,s) >= 0.7
                y(n,s) = 1;
            end
            if y(n,s) <= 0.3
                y(n,s) = 0;
            end
        end

        %Backward Pass
        %Output->Hidden
        for l=1:y_size
            do_k(1,l) = (y(n,l) - t(1,l)) * (y(n,l)*(1-y(n,l)));
        end

        delta_zy = (alpha .* z' * do_k);
        delta_zy_bias = alpha .* do_k;

        %Hidden->Input
        sigma_j = do_k * weight_zy';
        for m=1:z_size
            do_j(1,m) = (sigma_j(1,m)) .* (z(1,m)*(1-z(1,m)));
        end

        delta_xz = (alpha .* x' * do_j);
        delta_xz_bias = alpha .* do_j;

        %Momentum calculation
        momentum_zy = miu*delta_zy_old;
        momentum_xz = miu*delta_xz_old;
        momentum_bias_zy = miu*delta_zy_bias_old;
        momentum_bias_xz = miu*delta_xz_bias_old;

        %Weight Update
        weight_zy = weight_zy - delta_zy - momentum_zy;
        weight_xz = weight_xz - delta_xz - momentum_xz;
```

```
        bias_zy = bias_zy - delta_zy_bias - momentum_bias_zy;
        bias_xz = bias_xz - delta_xz_bias - momentum_bias_xz;

        %Cost function
        error(1,n) = 0.5.*sum((t-y(n,:)).^2);

        %Momentum update
        delta_zy_old = delta_zy;
        delta_xz_old = delta_xz;
        delta_zy_bias_old = delta_zy_bias;
        delta_xz_bias_old = delta_xz_bias;
    end
    error_per_epoch(1,epoch_count) = sum(error)/feature_row;

    disp(epoch_count)
    disp(sum(error)/feature_row)

    if error_per_epoch(1,epoch_count) < error_target
            stop = 1;
    end

    epoch_count = epoch_count+1;

end

plot(error_per_epoch)
Backprop_testing
```

D. Algoritma Pengujian

Setiap p = 1 sampai p = Pu :

Proses *feedforward*

3.1. Komputasi *Input Layer*. Setiap x_uji_{pi}, i = 1,2,...,I

- Menerima input x_uji_{pi}
- Mengirimkannya ke semua *Hidden Layer*.

3.2. Komputasi *Hidden Layer*. Setiap Z_j, j = 1,2,...,J

- $z\_in\_uji_j = v_{0j} + \Sigma x\_uji_{pi} v_{ij}$
- $z\_uji_j = \frac{1}{1+e^{-z\_in\_uji_j}}$

3.3. Komputasi *Output Layer*. Setiap y_k, k = 1,2,...,K

- $y\_in\_uji_k = w_{0k} + \Sigma z\_uji_j w_{jk}$
- $y\_uji_{pk} = \frac{1}{1+e^{-y\_in\_uji_k}}$
- Kuantisasi output
    - $0 <= y\_uji_{pk} < 0.3 \rightarrow y\_rec_{pk} = 0$
    - $0.7 < y\_uji_{pk} <= 1 \rightarrow y\_rec_{pk} = 1$
    - Else → $y\_rec_{pk} = y\_uji_{pk}$.

3.4. Menghitung *recognition rate*:

$$RR = \frac{banyaknya\ data\ output\ yang\ benar}{banyaknya\ data\ pengujian} \times 100\%$$

Output dikatakan benar jika nilai biner nya sama dengan binernya target.

Program MATLAB Proses Pengujian

```
%% Data Input
load('dataset.mat')
feature = dataset(106:151,1:5);
feature = feature{:,:};
class = dataset(106:151,6:8);
class = class{:,:};

[feature_row,feature_col] = size(feature);
[class_row,class_col] = size(class);

%% Normalisasi
feature_norm = zeros(size(feature));
for m = 1 : feature_row
    for n = 1 : feature_col
       feature_norm(m,n) = ((feature(m,n) -
min(feature(:,n)))/(max(feature(:,n)) - min(feature(:,n))));
    end
end

%% Hyperparameter & Declare Variable

%Variable
data_count = feature_row;
true_count = 0;

for n = 1:data_count
    %Forward Pass
    %Input -> Hidden
    x = feature(n,:);
    t = class(n,:);

    z_in = bias_xz + x*weight_xz;

    for m=1:z_size
        z(1,m) = 1/(1+exp(-z_in(1,m)));
    end

    %Hidden -> Output
    y_in = bias_zy + z*weight_zy;

    for l=1:y_size
        y(n,l) = 1/(1+exp(-y_in(1,l)));
    end


    %Threshold
    for s=1:3
        if y(n,s) >= 0.7
            y(n,s) = 1;
        end
        if y(n,s) <= 0.3
            y(n,s) = 0;
        end
    end
```

```
    %Recognition Rate
    [val, idx] = max(y(n,:));
    y(n,:) = zeros(size(y(n,:)));
    y(n,idx) = 1;

    if y(n,:) == t
       true_count = true_count + 1;
    end
end

recog_rate = true_count/data_count*100;
disp(recog_rate);
```

BAB III
ANALISIS HASIL PERCOBAAN

3.1 Hasil Percobaan

Parameter dari jaringan akan divariasikan yaitu metode inisialisasi , nilai learning rate (α), jumlah neuron pada hidden layer (J), dan menggunakan momentum atau tidak. Variasi dari parameter ini diharapkan dapat memberikan hasil yang dapat menggambarkan pengaruh parameter terhadap kondisi dan hasil dari ANN.

3.2.1 Hasil Variasi Metode Inisialisasi

Pada percobaan ini akan mencoba membandingkan dari hasil menggunakan metode inisialisasi random dan nguyen-widrow. Variabel tetap yang digunakan adalah :

- Learning Rate = 0.1
- Ukuran hidden layer = 7 node
- Target error maksimal = 0.1
- Maximum epoch = 2000 epoch
- Momentum = Tidak Digunakan

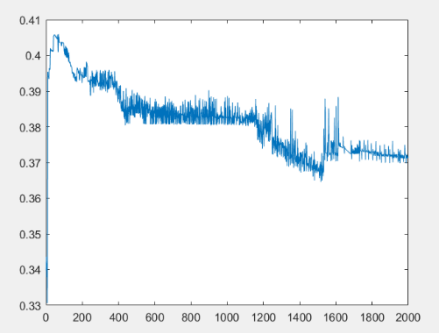
Hasil yang diperoleh adalah sebagai berikut:

a) Random Initialization

Final Error = 0. 3715

Epoch yang digunakan = 2000

Recognition rate = 39%

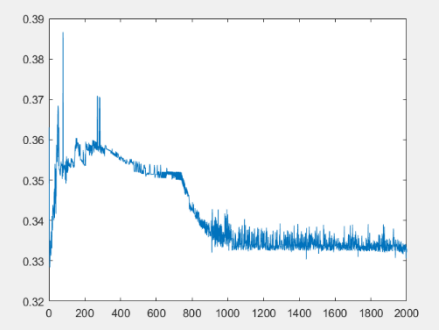


b) Nguyen-widrow Initialization

Final Error = 0. 3330

Epoch yang digunakan = 2000

Recognition rate = 39%



3.2.2 Hasil Variasi Nilai Learning Rate (α)

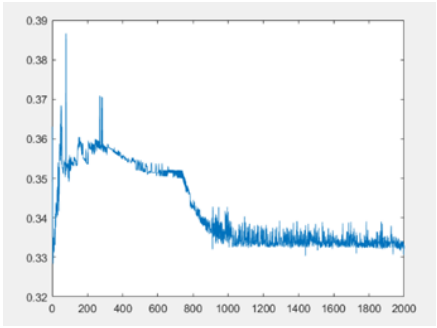
Pada percobaan ini akan mencoba membandingkan nilai learning rate yang berbeda. Nilai learning rate divariasikan dengan nilai 0.1,0.2,0.4 dan 0.8. Variabel tetap yang digunakan adalah :

- Metode Inisialisasi = Nguyen-widrow
- Ukuran hidden layer = 7 node
- Target error maksimal = 0.1
- Maximum epoch = 2000 epoch
- Momentum = Tidak Digunakan

Hasil yang diperoleh adalah sebagai berikut:

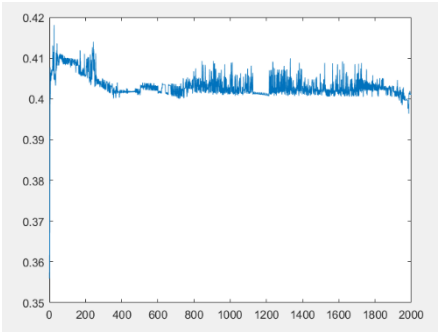
a) α = 0.1

Final Error = 0. 3330
Epoch yang digunakan = 2000
Recognition rate = 39%



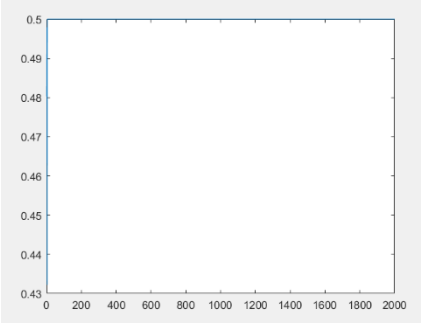
b) α = 0.2

Final Error = 0. 4286
Epoch yang digunakan = 2000
Recognition rate = 43%



c) α = 0.4

Final Error = 0. 5000
Epoch yang digunakan = 2000
Recognition rate = 58%

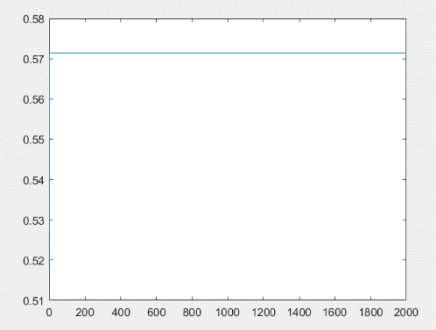


d) $\alpha = 0.8$

Final Error = 0. 5714

Epoch yang digunakan = 2000

Recognition rate = 15%



3.2.3 Hasil Variasi Ukuran Neuron Hidden Layer (J)

Pada percobaan ini akan mencoba membandingkan jumlah neuron hidden layer (J) yang berbeda. Jumlah neuron divariasikan dari 7,14,28,dan 56. Variabel tetap yang digunakan adalah :

- Metode Inisialisasi = Nguyen-widrow
- Learning rate = 0.1
- Target error maksimal = 0.1
- Maximum epoch = 2000 epoch
- Momentum = Tidak Digunakan

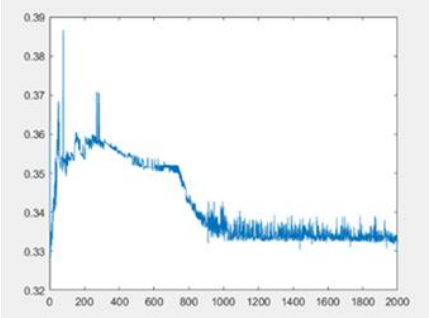
Hasil yang diperoleh adalah sebagai berikut:

a) J = 7

Final Error = 0. 3330

Epoch yang digunakan = 2000

Recognition rate = 39%

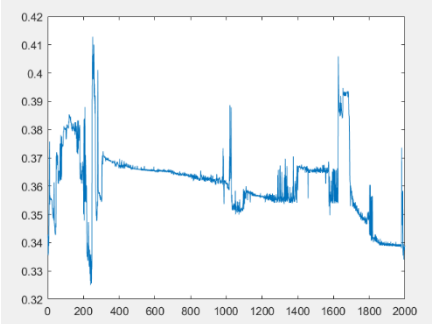


b) J = 9

Final Error = 0. 3387

Epoch yang digunakan = 2000

Recognition rate = 15%

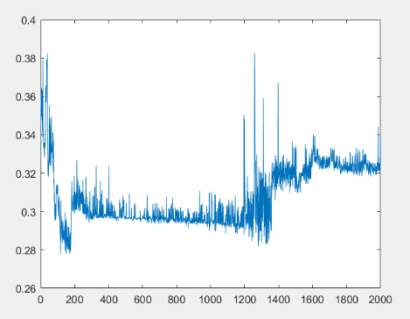


c) J = 13

Final Error = 0. 3245

Epoch yang digunakan = 2000

Recognition rate = 26%

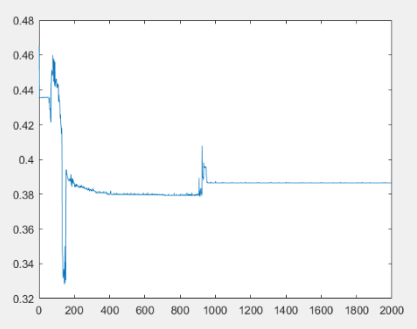


d) J = 5

Final Error = 0. 3864

Epoch yang digunakan = 2000

Recognition rate = 21%

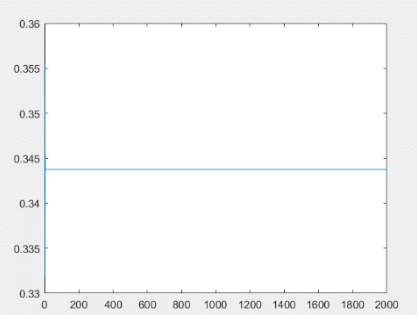


e) J = 1

Final Error = 0. 3438

Epoch yang digunakan = 2000

Recognition rate = 15%



3.2.4 Hasil Variasi Koefisien Momentum (μ)

Pada percobaan ini akan mencoba menganalisa penggunaan momentum serta mencoba beberapa nilai koefisien momentum (μ) yang berbeda. Koefisien momentum (μ) divariasikan senilai 0, 0.1, 0.2, 0.4 . Variabel tetap yang digunakan adalah :

- Metode Inisialisasi = Nguyen-widrow
- Learning rate = 0.1
- Ukuran hidden layer = 7 node
- Target error maksimal = 0.1
- Maximum epoch = 2000 epoch

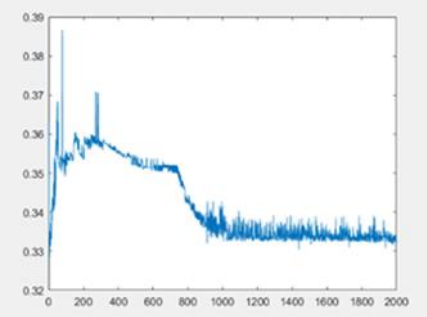
Hasil yang diperoleh adalah sebagai berikut:

a) μ = 0

Final Error = 0. 3330

Epoch yang digunakan = 2000

Recognition rate = 39%

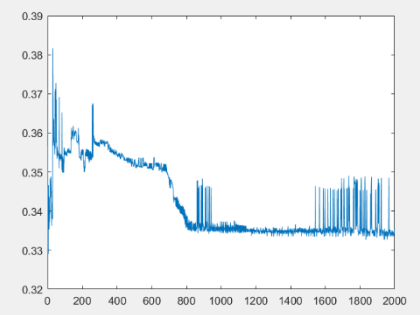


b) μ = 0.1

Final Error = 0. 3352

Epoch yang digunakan = 2000

Recognition rate = 41%

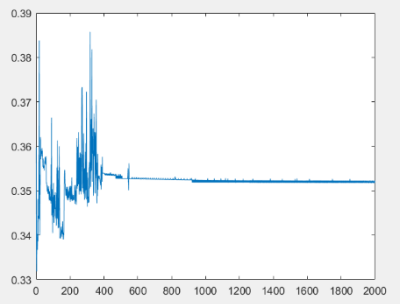


c) μ = 0.2

Final Error = 0. 3518

Epoch yang digunakan = 2000

Recognition rate = 39%

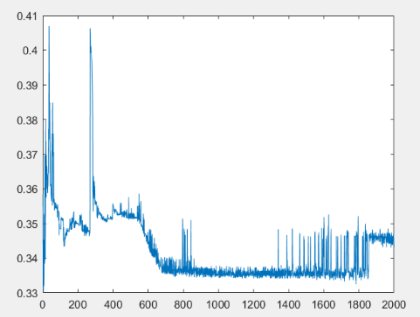


d) μ = 0.4

Final Error = 0. 4524

Epoch yang digunakan = 2000

Recognition rate = 54%

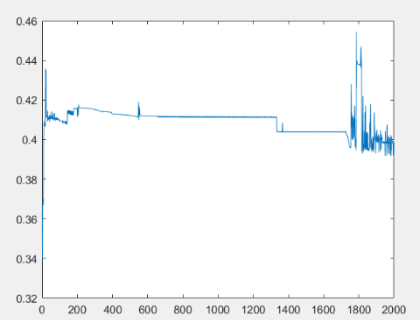


e) μ = 0.8

Final Error = 0. 4524

Epoch yang digunakan = 2000

Recognition rate = 54%



3.2 Analisis Hasil Data

3.2.1 Analisis Variasi Metode Inisialisasi

Pada hasil percobaan dengan variasi metode inisialisasi, dengan mengubah metode yang digunakan, maka dapat dianalisa pengaruh metode tersebut dalam proses dan hasil *backpropagation*. Hasil yang diperoleh bahwa metode inisialiasi nguyen-widrow dapat mempercepat ANN mencapai konvergensi. Namun secara hasil akhir tidak memberikan perbedaan hasil akhir yang signifikan dibandingkan metode inisialisasi acak. Sehingga dapat dikatakan bahwa metode nguyen-widrow cocok untuk diterapkan pada dataset atau sistem yang kompleks untuk mengurangi jumlah iterasi yang digunakan

3.2.2 Analisis Variasi Nilai Learning Rate

Pada hasil percobaan dengan variasi nilai learning rate, dengan mengubah nilai alpha, maka dapat dianalisa pengaruh alpha terdapat proses dan hasil *backpropagation*. Hasil yang diperoleh adalah semakin besar alpha, maka akan mempercepat jaringan mencapai konvergensi. Namun, hal ini menimbulkan kekurangan yaitu semakin besar alpha maka akan semakin besar potensi sistem tidak mencapai konvergensi. Sebaliknya adalah semakin kecil alpha, maka semakin lama waktu yang dibutuhkan untuk mencapai konvergensi namun semakin besar potensi sistem untuk mencapai konvergensi. Jaringan gagal untuk mencapai konvergensi dapat terlihat pada grafik plot dengan nilai alpha 0.4 dan 0.8.

Pada dataset ini nilai alpha yang kecil lebih cocok untuk digunakan karena jumlah instance pada dataset yang relative kecil yaitu 151 sehingga kemampuan komputasi masih sanggup untuk menangani jumlah iterasi yang banyak.

3.2.3 Analisis Variasi Jumlah Neuron Hidden Layer (J)

Pada hasil percobaan dengan variasi jumlah neuron hidden layer, dengan mengubah jumlah neuron, maka dapat dianalisa pengaruh dari jumlah neuron terhadap proses dan hasil *backpropagation*. Hasil yang diperoleh adalah idealnya semakin banyak neuron yang digunakan maka akan memberikan hasil yang lebih baik. Hal ini disebabkan karena semakin banyak neuron yang digunakan pada hidden layer maka akan semakin banyak bobot dan hubungan yang dapat di tune untuk memberikan hasil yang lebih baik.

3.2.4 Analisis Variasi Koefisien Momentum (μ)

Pada hasil percobaan dengan variasi koefisien momentum, dengan mengubah jumlah koefisien tersebut, kita dapat melihat pengaruh dari momentum terhadap proses dan hasil *backpropagation*. Momentum akan membuat sistem kita menjadi dinamis dengan

mempertimbangkan data sebelumnya[5]. Hasil yang diperoleh adalah penggunaan momentum dapat mempercepat ANN untuk mencapai konvergensi. Seperti terlihat pada $\mu = 0.1$ bahwa ANN sudah mencapai error disekitar 0.335 pada epoch sekitar 800 sedangkan ketika tidak menggunakan momentum, jaringan baru mencapai error disekitar 0.3335 pada epoch sekitar 900.

Namun penggunakaan koefisien momentum yang terlalu besar dapat memperlambat proses pelatihan karena semakin besar penalti yang diberikan terhadap nilai update bobot. Hal ini terlihat pada nilai koefisien 0.8.

BAB IV
KESIMPULAN

Setelah melakukan percobaan dengan berbagai variasi parameter dan metode, diperoleh kesimpulan yaitu:

1. *Backpropagation* merupakan metode ANN yang merupakan metode pembelajaran sistem yang mengubah nilai bobot dari neuron relatif terhadap error yang diperoleh
2. Inisialisasi nguyen-widrow mempercepat ANN untuk mencapai konvergensi
3. Nilai learning rate (α) berpengaruh terhadap durasi ANN untuk mencapai konvergensi dan potensi untuk mencapai konvergensi tersebut.
4. Jumlah neuron hidden layer berpengaruh terhadap hasil luaran ANN dimana semakin banyak jumlah neuron akan memberikan hasil yang lebih baik namun perlu diwaspadai pengaruh dari jumlah neuron yang terlalu banyak
5. Penggunaan momentum akan mengurangi noise pergerakan ANN dan mempercepat ANN untuk mencapai konvergensi

DAFTAR REFERENSI

[1] D. Wahyuningsih, I. Zuhroh, and - Zainuri, "Prediksi Inflasi Indonesia Dengan Model Artificial Neural Network," *J. Indones. Appl. Econ.*, vol. 2, no. 2, pp. 2–2008, 2008, doi: 10.21776/ub.jiae.2008.002.02.7.

[2] D. B. Kusumoputro, "*BACKPROPAGATION*," 2021.

[3] "Artificial_Neural_Network @ Www.Saedsayad.Com." [Online]. Available: http://www.saedsayad.com/artificial_neural_network.htm.

[4] "Teaching Assistant Evaluation," *Building*. pp. 2006–2006, 2006, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation.

[5] "stochastic-gradient-descent-with-momentum-a84097641a5d @ towardsdatascience.com." [Online]. Available: https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d#:~:text=Momentum %5B1%5D or SGD with,models are trained using it.