1. Perumusan Input-Output

Pada percobaan NN Identifikasi dan NN Invers, digunakan input dengan maksimum delay bernilai 3 sehingga input layer sebanyak 7 dengan perincian sebagai berikut :

* Input untuk NN Identifikasi

input(k,:) = [x(k-3) x(k-2) x(k-1) x(k) y(k-3) y(k-2) y(k-1)]

* Input untuk NN Invers

input(k,:) = [x(k-3) x(k-2) x(k-1) y(k-3) y(k-2) y(k-1) y(k)]

Sedangkan untuk target pada NN Identifikasi merupakan output dari plant yang sudah dinormalisasi y(k) dan output pada NN Invers merupakan input dari plant yang sudah dinormalisasi x(k).

Pada percobaan Open Loop, input untuk Invers-nya adalah output dari plant r(k) dan keluaran dari NN Invers u(k) dengan delay maksimum bernilai 3 sehingga input layer sebanyak 7 dengan perincian sebagai :

input(k,:) = [u(k-3) u(k-2) u(k-1) r(k-3) r(k-2) r(k-1) r(k)]

Sedangkan untuk input Identifikasi-nya adalah output dari Invers u(k) dan output dari identifikasi y(k) itu sendiri, yaitu :

input(k,:) = [u(k-3) u(k-2) u(k-1) u(k) y(k-3) y(k-2) y(k-1)]

1. Grafik Hasil

Data Percobaan

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Percobaan | Alpha | Hidden Layer | Momentum | Epoch | MSE Training | MSE Testing |
| 1 | NN Identifikasi | 0,20 | 14 | 0,50 | 163445 | 1,82E-05 | 7,21E-06 |
| 2 | NN Inverse | 0,20 | 14 | 0,50 | 170564 | 4,91E-05 | 4,78E-05 |
| 3 | NN Identifikasi PPR | 0,20 | 14 | 0,50 | 58468 | 3,14E-05 | 8,44E-05 |
| 4 | NN Identifikasi PPR | 0,20 | 14 | 0,50 | 20547 | 2,50E-03 | 2,70E-03 |

1. NN Identifikasi Math. Model

* Grafik MSE



* Perbandingan data training



* Perbandingan data testing



1. NN Invers Math. Model

* Grafik MSE



* Perbandingan data training



* Perbandingan data testing



1. NN Identifikasi PPR

* Grafik MSE



* Perbandingan data training



* Perbandingan data testing



1. NN Invers Math. Model

* Grafik MSE



* Perbandingan data training



* Perbandingan data testing



1. Lampiran program

* Program untuk perumusan Input-Output

1. NN Identifikasi Math. Model

function[data\_urut, target]=urutkan(x,y)

nilaimin=min(y(2:1000));

nilaimax=max(y(2:1000));

for i = 2: 1000

yt(i,:) = (2\*(y(i)-nilaimin))./(nilaimax-nilaimin) -1;

end

xy(1,:)=[0 0 0 x(1) 0 0 0];

xy(2,:)=[0 0 x(1) x(2) 0 0 yt(1)];

xy(3,:)=[0 x(1) x(2) x(3) 0 yt(1) yt(2)];

for i = 4 : 1000

xy(i,:) = [x(i-3) x(i-2) x(i-1) x(i) yt(i-3) yt(i-2) yt(i-1)];

end

data\_urut = xy;

target = yt;

end

1. NN Invers Math. Model

function[data\_urutinv, targetinv]=urutkan\_inv(x,y)

nilaimin=min(y(2:1000));

nilaimax=max(y(2:1000));

for i = 2: 1000

yt(i,:) = (2\*(y(i)-nilaimin))./(nilaimax-nilaimin) -1;

end

xy(1,:)=[0 0 0 0 0 0 yt(1)];

xy(2,:)=[0 0 x(1) 0 0 yt(1) yt(2)];

xy(3,:)=[0 x(1) x(2) 0 yt(1) yt(2) yt(3)];

for i = 4 : 1000

xy(i,:) = [x(i-3) x(i-2) x(i-1) yt(i-3) yt(i-2) yt(i-1) yt(i)];

end

data\_urutinv = xy;

targetinv = x;

end

1. NN Identifikasi PPR

function[data\_urutppr, target]=urutkanppr(x,y)

nilaimin\_x=min (x);

nilaimax\_x=max (x);

nilaimin\_y=min (y);

nilaimax\_y=max (y);

for i = 1: 5000

yt(i,:) = (2\*(y(i)-nilaimin\_y))./(nilaimax\_y-nilaimin\_y) -1;

xt(i,:) = (2\*(x(i)-nilaimin\_x))./(nilaimax\_x-nilaimin\_x) -1;

end

xy(1,:)=[0 0 0 xt(1) 0 0 0];

xy(2,:)=[0 0 xt(1) xt(2) 0 0 yt(1)];

xy(3,:)=[0 xt(1) xt(2) xt(3) 0 yt(1) yt(2)];

for i = 4 : 5000

xy(i,:) = [xt(i-3) xt(i-2) xt(i-1) xt(i) yt(i-3) yt(i-2) yt(i-1)];

end

data\_urutppr = xy;

target = yt;

end

1. NN Invers PPR

function[data\_urut, target]=urutkanppr\_inv(x)

nilaimin\_x=min(x(:,2));

nilaimax\_x=max(x(:,2));

nilaimin\_y=min(x(:,3));

nilaimax\_y=max(x(:,3));

for i = 1: 5000

x\_norm(i,:) = (2\*(x(i,2)-nilaimin\_x))./(nilaimax\_x-nilaimin\_x) -1;

y\_norm(i,:) = (2\*(x(i,3)-nilaimin\_y))./(nilaimax\_y-nilaimin\_y) -1;

end

xy(1,:)=[0 0 0 0 0 0 y\_norm(1)];

xy(2,:)=[0 0 x\_norm(1) 0 0 y\_norm(1) y\_norm(2)];

xy(3,:)=[0 x\_norm(1) x\_norm(2) 0 y\_norm(1) y\_norm(2) y\_norm(3)];

for i = 4 : 5000

xy(i,:) = [x\_norm(i-3) x\_norm(i-2) x\_norm(i-1) y\_norm(i-3) y\_norm(i-2) y\_norm(i-1) y\_norm(i)];

end

data\_urut = xy;

target = x\_norm;

end

* Program Training (backpropagation)

function [finalerror,v,w,v0,w0,Y,epoch] = backpropaagation(input,target,alpha,lh,momentum)

x = input;

t = target;

N = 0.5\*length(x(:,1));

lx = length(x(1,:));

ly = length(t(1,:));

%Inisialisasi nilai bobot

beta = 0.7\*lh^(1/lx);

v = rand(lx,lh)-0.5\*ones(lx,lh);

w = rand(lh,ly)-0.5\*ones(lh,ly);

v0 = -beta + (beta+beta)\*rand(1,lh);

w0 = -beta + (beta+beta)\*rand(1,ly);

norm\_v = zeros(1,lh);

norm\_w = zeros(1,ly);

for j = 1 : lh

for i = 1 : lx

norm\_v(j) = norm\_v(j) + v(i,j)^2;

v(i,j) = (beta/sqrt(norm\_v(j)))\*v(i,j);

end

end

for j = 1 : ly

for i = 1 : lh

norm\_w(j) = norm\_w(j) + w(i,j)^2;

w(i,j) = (beta/sqrt(norm\_w(j)))\*w(i,j);

end

end

wjk = transpose(w);

errortotal = 100;

MSSE = errortotal/N;

error = 0;

epoch = 0;

%Momentum

w1=zeros(lh,ly);

v1=zeros(lx,lh);

w00=zeros(1,ly);

v00=zeros(1,lh);

tic

%Training

while MSSE > 1.86\*10^-7

for n = 1:N

%menghitung semua sinyal input dengan bobotnya

for i = 1 : lh

z\_in(i) = v0(i) + x(n,:) \* v(:,i);

z(i) = (1 - exp(-z\_in(i)))/(1 + exp(-z\_in(i)));

end

for j = 1 : ly

y\_in(j) = w0(j) + z \* w(:,j);

y(j) = (1 - exp(-y\_in(j)))/(1 + exp(-y\_in(j)));

end

%Backpropagation of error dari bobot w

for i = 1: lh

for j = 1 : ly

dk(j) = (t(n,j)-y(j))\* (((1+y(j))\*(1-y(j)))\*0.5); %menghitung informasi error :

deltaw(i,j) = alpha \* dk(j) \* z(i) + momentum \* w1(i,j); %menghitung besarnya koreksi bobot unit output

end

end

deltaw0 = alpha \* dk + momentum \* w00; %menghitung koreksi error bias unit output

w1 = deltaw;

w00 = deltaw0;

d\_in = dk \* wjk; %menghitung semua koreksi error

for j = 1 : lx

for i = 1 : lh

dj(i) = d\_in(i) \* (((1+z(i))\*(1-z(i)))\*0.5); %menghitung nilai aktivasi koreksi error

deltav(j,i) = alpha \* dj(i) \* x(n,j) + momentum \* v1(j,i); %menghitung koreksi bobot unit hidden

end

end

deltav0 = alpha \* dj + momentum \* v00; %menghitung koreksi error bias unit hidden

v1 = deltav;

v00 = deltav0;

%update bobot

w = w + deltaw;

w0 = w0 + deltaw0;

v = v + deltav;

v0 = v0 + deltav0;

error(n)= 0.5\*(t(n,:)-y)\*(t(n,:)-y);

Y(n,:)=y;

end

epoch = epoch+1;

errortotal(epoch) = sum(error);

clc

epoch

MSSE=sum(error)/N

time=toc

save('Hasil\_training.mat','MSSE','errortotal','v','w','v0','w0','Y','epoch');

end

finalerror=errortotal(epoch)/N;

end

* Program Testing (feedforward)

function [YTest, MSSETest] = feedforwardNNID(input,output)

load('Hasil\_Training.mat');

x = input;

t = output;

N = 0.5\*length(x(:,1));

lx = length(x(1,:));

ly= length(t(1,:));

for n = (N+1) : (2\*N)

%menghitung semua sinyal input dengan bobotnya

for i = 1 : lh

z\_in(i) = v0(i) + x(n,:) \* v(:,i);

z(i) = (1 - exp(-z\_in(i)))/(1 + exp(-z\_in(i)));

end

for j = 1 : ly

y\_in(j) = w0(j) + z \* w(:,j);

y(j) = (1 - exp(-y\_in(j)))/(1 + exp(-y\_in(j)));

end

error(n-N)= 0.5\*(t(n,:)-y)\*(t(n,:)-y);

YTest(n-N,:)=y;

end

MSSETest=sum(error)/N;

save('Hasil\_Testing.mat','YTest','MSSETest');

end

* Program Open Loop

function[Y\_ID]=Open\_Loop(reference)

load('Hasil\_TrainingID.mat');

vID=v;

v0ID=v0;

wID=w;

w0ID=w0;

load('Hasil\_TrainingINV.mat');

vInv1=v;

v0Inv1=v0;

wInv1=w;

w0Inv1=w0;

r = reference;

N = length(r(:,1));

lx = 7;

ly = 1;

%Open Loop System

for n = 1:N

if n==1

xInv1(n,:)=[0 0 0 0 0 0 r(1)];

elseif n==2

xInv1(n,:)=[0 0 Y\_Inv1(1) 0 0 r(1) r(2)];

elseif n==3

xInv1(n,:)=[0 Y\_Inv1(1) Y\_Inv1(2) 0 r(1) r(2) r(3)];

else

xInv1(n,:) = [Y\_Inv1(n-3) Y\_Inv1(n-2) Y\_Inv1(n-1) r(n-3) r(n-2) r(n-1) r(n)];

end

%menghitung semua sinyal input dengan bobotnya

for i = 1 : lh

z\_inInv1(i) = v0Inv1(i) + xInv1(n,:) \* vInv1(:,i);

zInv1(i) = (1 - exp(-z\_inInv1(i)))/(1 + exp(-z\_inInv1(i)));

end

for j = 1 : ly

y\_inInv1(j) = w0Inv1(j) + zInv1 \* wInv1(:,j);

yInv1(j) = (1 - exp(-y\_inInv1(j)))/(1 + exp(-y\_inInv1(j)));

end

Y\_Inv1(n,:)=yInv1;

x\_Inv1(n,:)=xInv1(n,:);

if n==1

xID(n,:)=[0 0 0 Y\_Inv1(1) 0 0 0];

elseif n==2

xID(n,:)=[0 0 Y\_Inv1(1) Y\_Inv1(2) 0 0 Y\_ID(1)];

elseif n==3

xID(n,:)=[0 Y\_Inv1(1) Y\_Inv1(2) Y\_Inv1(3) 0 Y\_ID(1) Y\_ID(2)];

else

xID(n,:) = [Y\_Inv1(n-3) Y\_Inv1(n-2) Y\_Inv1(n-1) Y\_Inv1(n) Y\_ID(n-3) Y\_ID(n-2) Y\_ID(n-1)];

end

%menghitung semua sinyal input dengan bobotnya

for i = 1 : lh

z\_inID(i) = v0ID(i) + xID(n,:) \* vID(:,i);

zID(i) = (1 - exp(-z\_inID(i)))/(1 + exp(-z\_inID(i)));

end

for j = 1 : ly

y\_inID(j) = w0ID(j) + zID \* wID(:,j);

yID(j) = (1 - exp(-y\_inID(j)))/(1 + exp(-y\_inID(j)));

end

error(n)= 0.5\*(r(n,:)-yID)\*(r(n,:)-yID);

Y\_ID(n,:)=yID;

x\_ID(n,:)=xID(n,:);

end

MSSE\_DIC=sum(error)/N;

save('Hasil\_OpenLoop.mat','MSSE\_DIC','Y\_Inv1','x\_Inv1','Y\_ID','x\_ID');

end