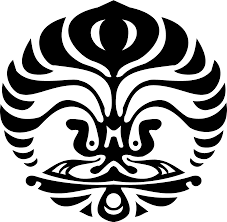
**LAPORAN PROYEK**

**SISTEM BERBASIS PENGETAHUAN**

**“IDENTIFIKASI SISTEM MENGGUNAKAN**

**ARTIFICIAL NEURAL NETWORK”**



Dibuat oleh :

Nama : Suwandi Dwi Sahputro

NPM : 1306413605

Dosen: Prof. Dr. Eng. Drs. Beyamin Kusumoputro, M.Eng.

**DEPARTEMEN TEKNIK ELEKTRO**

**FAKULTAS TEKNIK UNIVERSITAS INDONESIA**

**MARET 2015**

**DAFTAR ISI**

[**BAB 1. PENDAHULUAN** 1](#_Toc415844302)

[**1.1 LATAR BELAKANG NEURAL NETWORK** 1](#_Toc415844303)

[**1.2 ARSITEKTUR NEURAL NETWORK** 2](#_Toc415844304)

[**1.3 METODE PEMBELAJARAN** 3](#_Toc415844305)

[**BAB 2. BACKPROPAGATION** 4](#_Toc415844306)

[**2.1 ALGORITMA BACKPROPAGATION** 4](#_Toc415844307)

[**2.2 FAKTOR-FAKTOR DALAM PEMBELAJARAN** 7](#_Toc415844308)

[**BAB 3. PERCOBAAN** 8](#_Toc415844309)

[**3.1 *MENGGENERATE* DATA DARI PLANT** 8](#_Toc415844310)

[**3.2 PROSES PENGOLAHAN DATA INPUT** 9](#_Toc415844313)

[**3.3 METODE PERCOBAAN**](#_Toc415844314) 10

[**REFERENSI**](#_Toc415844320) 14

**DAFTAR GAMBAR**

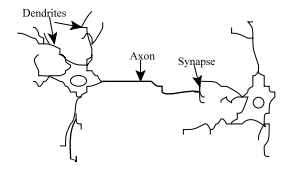
[Gambar 1.1.1 Jaringan Saraf Manusia 1](#_Toc415845167)

[Gambar 1.1.2 Struktur Sederhana ANN 2](#_Toc415845168)

# **BAB 1. PENDAHULUAN**

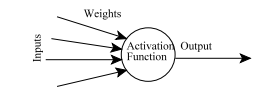
## **1.1 LATAR BELAKANG NEURAL NETWORK**

Artificial Neural Network (ANN) merupakan sebuah sistem yang mengadopsi cara kerja sistem jaringan saraf pada makhluk hidup. Jaringan saraf pada makhluk hidup memiliki kemampuan menerima impuls yang random dari sensor pada tubuh kemudian impuls tersebut masuk ke dalam jaringan saraf lalu disampaikan ke otak dan dikembalikan hasil keluarannya sebagai respon. Jaringan saraf makhluk hidup terdiri dari banyak sel. Ketika impuls masuk ke jaringan saraf, kemudian impuls tersebut disalurkan ke dalam neuron yang lain dengan cara mengaktivasikan neuron tujuannya.



Gambar 1.1.1 Jaringan Saraf Manusia

Jaringan saraf pada makhluk hidup memiliki jumlah yang sangat banyak sehingga dalam memodelkannya sangat sulit tetapi kita dapat mengadopsi cara kerja dari jaringan saraf dari saat menerima input yang berupa impuls sampai menghasilkan output yang berupa respon. Secara umum, jaringan saraf memiliki input yang berupa impuls, setiap impuls memiliki tingkat kekuatan yang berbeda terhadap suatu neuron atau juga suatu neuron memiliki tingkat respon yang berbeda terhadap input yang berbeda, yang dikenal dengan istilah bobot pada ANN. Kemudian input ini diproses di dalam neuron baru ditransmisikan lagi ke neuron yang lain, proses ini dimodelkan sebagai fungsi aktivasi pada ANN. Yang terakhir adalah output berupa respon yang merupakan hasil dari proses yang sedemikian rupa dan dimodelkan dalam bentuk fungsi matematika di dalam ANN.



Gambar 1.1.2 Struktur Sederhana ANN

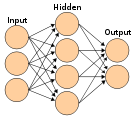
Dalam gambar 1.2 dapat dilihat bahwa terdapat bobot untuk setiap input terhadap salah satu neuron. Untuk suatu objek yang belum pernah disentuh oleh tubuh atau pada ANN proses awal memasukkan input, nilai bobot tersebut akan bersifat random. Dengan mengatur nilai bobot pada ANN , kita dapat membuat sistem tersebut menghasilkan output sesuai dengan yang kita inginkan. Karena jaringan saraf memiliki jumlah neuron yang sangat banyak, begitu juga dengan implementasinya pada ANN, maka pengaturan bobot tersebut cukup sulit dilakukan secara manual. Untuk itu, digunakanlah algoritma yang dapat mengatur bobot pada ANN untuk menghasilkan output yang kita inginkan. Proses pengaturan bobot ini dinamakan proses learning atau training.

Fungsi dari penggunaan ANN antara lain :

* Pattern recognition
* Forecasting
* Data compression
* Dan lain-lain

## **1.2 ARSITEKTUR NEURAL NETWORK**

Pada jaringan ANN, setiap neuron memiliki beberapa masukkan dan keluaran. Input pada ANN dapat berupa data mentah atau juga data hasil olahan dari neuron sebelumnya. Sementara hasil keluaran merupakan hasil akhir dari proses neuron.



Gambar 1.2.1 Skema Neural Network

Pada skema neural network, dapat dilihat bahwa terdapat 3 lapisan utama pada sistem ANN yaitu :

* **Input Layer**

Input layer berfungsi sebagai portal yang menghubungkan jaringan ke sumber data. Neuron yang berada pada input layer tidak dapat memproses data masukan tetapi hanya dapat meneruskan ke neuron pada layer selanjutnya.

* **Hidden Layer**

Hidden layer dapat terdiri dari beberapa layer tergantung dari sistem yang ingin dianalisa. Hidden layer ini berfungsi untuk menerima masukan dari input layer. Masing-masing neuron pada hidden layer ini dipengaruhi oleh semua neuron pada input layer dengan nilai bobot yang berbeda-beda. Pada ANN, setiap neuron pada hidden layer mendapatkan input dari hasil penjumlahan dari perkalian neuron pada input layer dengan bobotnya masing-masing. Setelah neuron pada hidden layer mendapatkan input, maka data tersebut tidak langsung diteruskan ke dalam layer berikutnya tetapi perlu dilakukan suatu proses yang dinamakan proses aktivasi. Proses aktivasi pada ANN ini menggunakan fungsi sigmoid baik itu unipolar maupun bipolar tergantung dari sistem yang ingin dianalisa.

* **Output Layer**

Output layer merupakan layer terakhir pada ANN yang berfungsi untuk menerima input dari layer sebelumnya. Nilai input yang diterima tidak langsung menjadi nilai keluaran sistem tetapi harus melalui proses aktivasi dengan menggunakan fungsi sigmoid.

## **1.3 METODE PEMBELAJARAN**

Pada saat ini, terdapat 3 metode pembelajaran yaitu :

* **Supervised learning**

Pada proses pembelajaran tipe ini, kita diberikan sepasang kumpulan data yaitu input dan target. Pada prosesnya, diinginkan input yang ada dapat dipetakan menjadi target yang ditentukan dengan cara mengatur bobotnya. Pengaturan bobotnya ditentukan oleh cost function yang berhubungan dengan tingkat ketidaksesuaian hasil pemetaan dengan target yang sudah ditentukan. Cost function yang biasa digunakan adalah mean-squarred error. Metode pembelajaran ini biasanya digunakan untuk pattern recognition dan regression.

* **Unsupervised learning**

Pada proses pembelajaran tipe ini, kita diberikan data input dan output yang merupakan suatu fungsi input. Cost function yang digunakan adalah fungsi kuadrat dari selisih input dengan nilai fungsi input. Metode pembelajaran ini biasa digunakan untuk permasalahan estimasi. Contohnya adalah clustering, filtering, compression dan statistical distribution.

* **Reinforcement learning**

Pada proses pembelajaran tipe ini, kita tidak diberikan data input tetapi data tersebut didapatkan dari hasil interaksi langsung dengan lingkungan luar. Pada setiap waktu, secara otomatis akan dihasilkan output, data observasi dan cost function. Biasanya lingkungan dimodelkan dengan Markov Decision Process yang menggunakan probabilitas.

Salah satu algoritma pembelajaran yang sering digunakan adalah algoritma backpropagation yang termasuk ke dalam kategori supervised learning.

# 

# **BAB 2. BACKPROPAGATION**

## **2.1 ALGORITMA BACKPROPAGATION**

Backpropagation merupakan algoritma pembelajaran untuk memperkecil tingkat error dengan menyesuaikan bobot berdasarkan perbedaan output dan target yang diinginkan.

Proses pembelajaran :

* Pengambilan input
* Indentifikasi error
* Penyesuaian bobot

Algoritma backpropagation :

Step 0 : inisialisasi bobot

* Random

Melakukan inisialisasi bobot secara acak tetapi dalam range angka tertentu. (-x sampai +x)

* Nguyen-Widrow

Metode Nguyen-Widrow sebenarnya juga memakai cara acak tetapi terdapat proses di dalamnya sehingga proses pembelajaran dapat menjadi lebih cepat.

Metode inisialisasi Nguyen-Widrow :

Menentukan faktor skala :

n : jumlah unit input layer

p : jumlah unit hidden layer

β : faktor skala

Algoritma perhitungannya :

1. Inisialisasi bobot secara random

Vij = bilangan acak antara -0.5 sampai 0.5

1. Hitung norm dari setiap bobot pada masing-masing hidden layer
2. Update bobot Vij =
3. Set bias V0j : bilangan acak antara sampai
4. Lakukan langkah 1-4 untuk setiap bobot pada masing-masing output layer yaitu W.

Step 1 : selama kondisi stopping FALSE, lakukan step 2-9

Step 2 : Untuk setiap pasangan pembelajaran, lakukan step 3-8

Step 3 : untuk setiap unit input

* Menerima input Xi
* Mengirimkannya ke semua unit hidden layer

Step 4 : untuk setiap unit hidden layer

* Menghitung semua sinyal input dengan bobotnya :

Z\_inj

* Menghitung nilai aktivasi setiap unit hidden sebagai output unit hidden layer :

Zj = f(z\_inj)

* Mengirimkan nilai aktivasi sebagai input untuk unit output layer

Step 5 : untuk setiap unit output layer

* Menghitung semua sinyal input dengan bobotnya :

y\_ink

* Menghitung nilai aktivasi setiap unit output sebagai output jaringan

Yk = f(y\_ink)

Backpropagation of error

Step 6 : untuk setiap unit output

* Menerima pola target yang bersesuaian dengan pola input
* Menghitung informasi error :

ᵟk = (tk – yk) f’(y\_ink)

* Menghitung besarnya koreksi bobot unit output :

α ∂ E(wjk)/ ∂wjk = α ᵟk zj

* Menghitung besarnya koreksi bias output :

α ᵟk

* Mengirimkan ᵟk ke unit-unit yang ada pada hidden layer

Step 7 : untuk setiap unit hidden

* Menghitung semua koreksi error :
* Menghitung nilai aktivasi koreksi error :

ᵟk = . f’(z\_inj)

* Menghitung koreksi bobot unit hidden :

α ᵟj xi

* Menghitung koreksi error bias unit hidden :

α ᵟj

Step 8 : Setiap unit output

* Update bobot dan biasnya :

wjk(baru) = wjk(lama) + ∆wjk

w0k(baru) = w0k(lama) + ∆w0k

* Setiap unit hidden

vjk(baru) = vjk(lama) + ∆vjk

v0k(baru) = v0k(lama) + ∆v0k

Step 9 : Kondisi stopping ( Toleransi error )

\*Keterangan :

- Toleransi error biasanya bernilai 0.01

- tk adalah target output yang diinginkan

- α memiliki range nilai 0.2 sampai dengan 0.4

- v0k dan w0k adalah bobot untuk bias pada hidden layer dan output layer

## **2.2 FAKTOR-FAKTOR DALAM PEMBELAJARAN**

Beberapa faktor yang mempengaruhi keberhasilan algoritma backpropagation :

* Inisialisasi bobot

Bobot awal menentukan apakah jaringan akan mencapai global minimum atau local minimum kesalahan dan seberapa cepat jaringan akan konvergen. Inisialisasi secara acak merupakan cara yang paling sering digunakan. Untuk mempercepat proses pelatihan, suatu modifikasi dari inisialisasi acak diperkenalkan oleh Nguyen dan Widrow.

* Laju pembelajaran

Merupakan parameter jaringan dalam mengendalikan proses penyesuaian bobot. Nilai laju pembelajaran yang optimal bergantung pada kasus yang dihadapi. Laju pembelajaran yang terlalu kecil menyebabkan konvergensi jaringan menjadi lebih lambat, sedangkan laju pembelajaran yang terlalu besar dapat menyebabkan ketidakstabilan pada jaringan. Laju pembelajaran memiliki range ideal yaitu 0.2 sampai 0.4.

* Momentum

Momentum digunakan untuk mempercepat pelatihan jaringan. Metode momentum melibatkan penyesuaian bobot ditambah dengan faktor tertentu dari penyesuaian sebelumnya. Penyesuaian ini dinyatakan sebagai :

Momentum ini membuat laju dari perubahan bobot berada pada jalur yang telah dibuat pada saat inisialisasi bobot dan membuat bobot selanjutnya akan berada pada arah jalur yang telah dibuat sebelumnya. Dalam hal ini, momentum digunakan untuk mempercepat proses pembelajaran. Momentum memiliki range dari 0 – 0.9 .

* Local Minimum

Dalam menentukan tingkat error yang paling kecil, backpropagation menggunakan pendekatan secara gradien. Jaringan neural akan mengikuti contour dari error yang dibuat selama proses pembelajaran kemudian mencari titik minimum dari contour tersebut. Untuk jaringan neural yang sederhana misalkan berdimensi dua. Maka untuk mencari nilai minimum cenderung lebih murah karena dinamika contour dari error yang cenderung stabil. Namun untuk jaringan neural yang lebih kompleks, maka dimensi yang dibuat oleh error bukan lagi berdimensi dua melainkan bisa lebih. Dalam jaringan neural yang kompleks, terdapat dinamika error yang dapat menyebabkan kesalahan dalam menentukan titik minimum untuk menandakan proses pembelajaran telah selesai karena titik minimum yang ada bisa merupakan titik local minimum yang dapat terbaca oleh program. Jika program membaca local minimum sebagai titik minimum yang sesungguhnya, maka proses pembelajaran akan seperti berhenti di titik tersebut tetapi sebenarnya titik yang dimaksud tersebut belum mencapai titik minimum yang sebenarnya yaitu global minimum. Kesalahan seperti ini yang seringkali membuat proses pembelajaran menjadi sangat lama dan harus sering diulang.

# **BAB 3. PERCOBAAN**

Pada percobaan kali ini, digunakan algoritma backpropagation dengan ketentuan sebagai berikut :

* + - Input layer memiliki 6 unit neuron yaitu u(k-1), u(k-2), u(k-3), y(k-1), y(k-2) dan y(k-3)
    - Output layer memiliki 1 unit neuron yaitu y(k).

## **3.1 *MENGGENERATE* DATA DARI PLANT**

function[input,output] = plant

x= 2\*rand(1000,1) - 1;

y(1)=0;

for i = 2 : 1000

y(i) = (1/(1+y(i-1)^2) - 0.25\*x(i) - 0.3\*x(i-1));

end

input = x;

output = transpose(y);

end

Untuk mendapatkan data sebagai perbandingan masukkan input acak pada range -1 sampai 1 untuk mendapatkan nilai output.

## **3.2 PROSES PENGOLAHAN DATA INPUT**

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

nilaimin=min(y);

nilaimax=max(y);

for i = 2: 1000

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

end

for i = 3 : 999

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

t(i-2,1)= y(i+1);

end

data\_urut = xy;

target = t;

end

Mengurutkan data untuk input dengan delay 3 sehingga dimensi input berjumlah 6 yang berasal dari input yang di*delay* dan output yang di*delay.*

## **3.3 METODE PERCOBAAN**

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

x = input;

t = target;

N = 497;

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);

%Training

while MSSE > 0.0001

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)))/2); %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)))/2); %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

save('hasil\_training2.mat','errortotal','v','w','v0','w0','Y','epoch');

if mod(epoch,200)==0

figure(1)

plot(errortotal/N)

end

end

finalerror=errortotal(epoch)

end

Dalam melakukan percobaan ini, digunakan software MATLAB. Pada software MATLAB, saya membuat fungsi yang berisi algoritma dari backpropagation di dalam sebuah m file dengan nama backpropagation. Di dalam fungsi tersebut, saya membuat fungsi dengan parameter input, target, alpha, momentum dan hidden layer. Fungsi tersebut mengeluarkan output berupa bobot v,w,v0,w0,jumlah epoch dan nilai error. Selain fungsi tersebut, saya juga membuat fungsi untuk melakukan testing pada data selain dari pembelajaran. Fungsi yang saya buat bernama testbackprop dengan parameter v,w,v0 dan w0 yang sudah dihasilkan dari fungsi sebelumnya serta parameter data sisa.

# 

# **REFERENSI**

<http://en.wikipedia.org/wiki/Artificial_neural_network>

<http://catatan-syam.blogspot.com/2013/05/jaringan-syaraf-tiruan-artificial.html>

<http://arxiv.org/ftp/cs/papers/0308/0308031.pdf>

<http://ariesalva.blogspot.com/2009/10/artificial-neural-network-ann.html>

<http://staff.itee.uq.edu.au/janetw/cmc/chapters/BackProp/index2.html>