KOMPARASI METODE JARINGAN SYARAF TIRUAN SOM DAN LVQ UNTUK MENGIDENTIFIKASI DATA BUNGA IRIS

Meri Azmi*
*Dosen Teknologi Informasi
Politeknik Negeri Padang
E-mail : pnp@polinpdg.ac.id

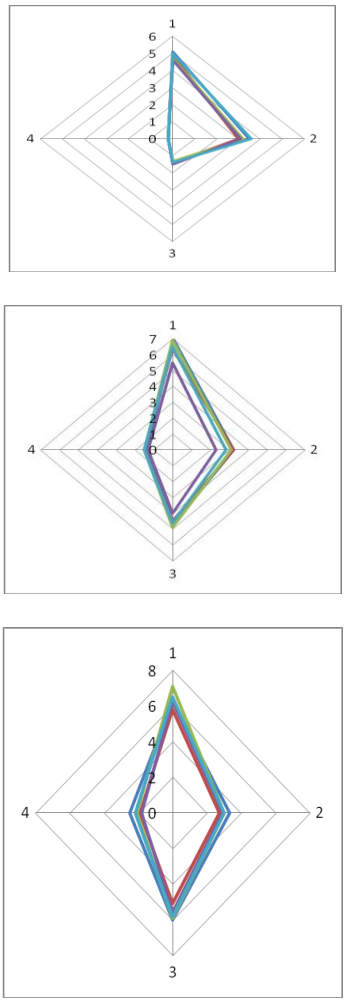
Abstract

In this journal we tried to compare the performance between SOM & LVQ Methods of Neural Network. Iris data was used to perform the comparison. We found that SOM has better performance (89.33%) than LVQ (86.67%). But LVQ has faster time process(0.11 s) than SOM (0.20 s).

Keyword : SOM, LVQ, Neural Network

1. Pendahuluan

Dalam tulisan ini penulis ingin membandingkan beberapa metode yang ada dalam jaringan syaraf tiruan. Metode yang dipilih untuk dibandingkan adalah metode SOM dan LVQ.



Gambar 1. Grafik radar data bunga Iris a) jenis pertama b) jenis kedua dan c) jenis ketiga

Data yang akan digunakan dalam menguji ke dua metode di atas adalah data bunga iris. Gambar 1 adalah representasi grafik radar dari tiga jenis bunga iris. Tiga jenis data iris yang digunakan memiliki 4 dimensi dan masing-masing jenis bunga memiliki 50 data sehingga total keseluruhan data yang ada adalah 150.

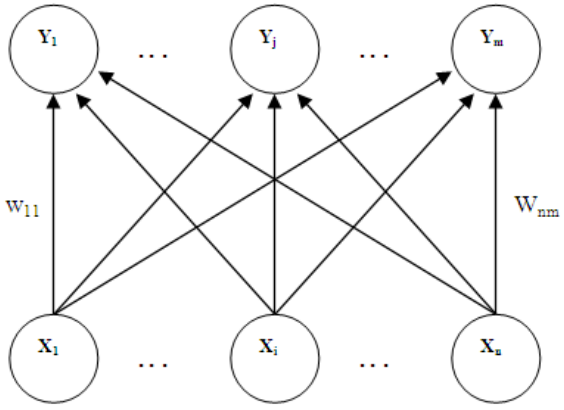
2. Tinjauan Pustaka

Perbandingan Arsitektur

Arsitektur SOM

SOM terdiri dari 2 lapis yakni input dan output, di mana antar lapisannya dihubungkan oleh bobot tertentu yang sering disebut sebagai vektor pewakil vector code book atau reference vector [Fausette, 1994]

Pada arsitektur satu dimensi, input layer (X) akan berbentuk linear dan demikian pula outputnya (Y). Pada arsitektur ini, unit yang bersebelahan akan memiliki lebih sedikit perbedaan daripada unit yang letaknya lebih jauh. Lebih jelasnya dapat dilihat pada gambar 2.3.



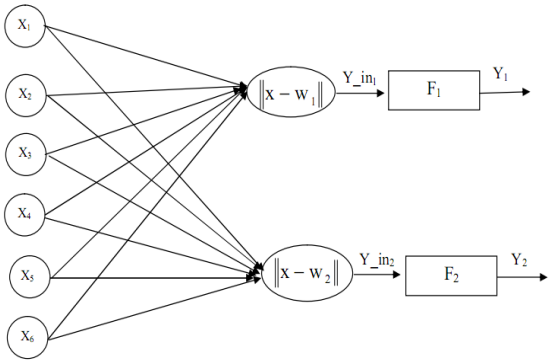
Gambar 2. Arsitektur SOM

c)

Bobot (W) digunakan sebagai salahsatu komponen untuk menentukan jarak terhadap output layer. Jarak terdekat akan menjadi referensi pengklasteran data. Apabila ada jarak ke output 2 adalah yang paling dekat maka vektor yang dimasuk akan ditempatkan di kelas output ke 2, dan kemudian bobot yang ada akan diupdate dengan mempertimbangkan masuknya vektor masukan tersebut ke output layer kelas 2. Bobot baru ini kemudian akan menjadi referensi berikutnya untuk masukan selanjutnya.

Arsitektur LVQ

Seperti halnya SOM, LVQ juga terdiri dari 2 lapisan yakni input (X) dan output (Y), di mana antar lapisannya dihubungkan oleh bobot tertentu yang sering disebut sebagai vektor pewakil (W). Informasi yang diberikan ke jaringan pada saat pembelajaran bukan hanya vektor data saja melainkan informasi kelas dari data juga ikut dimasukkan. Tujuan dari dimasukkannya informasi ini adalah untuk menambah keakuratan jaringan dalam mengklasifikasikan data.



Gambar 3. Arsitektur LVQ

X = Vektor Masukan (x1…, xi…, xn)
F = Lapisan kompetitif
Y_in = Masukan kelapisan kompetitif
Y = Keluaran (Output)
W = Vektor bobot untuk unit keluaran
||X-W|| = Selisih nilai jarak Euclidean antara vektor input dengan vektor bobot untuk unit Output

Ketika hasil pemrosesan jaringan memberikan hasil klasifikasi yang sama dengan informasi kelas yang diberikan di awal maka vektor pewakil akan disesuaikan agar lebih dekat dengan vektor masukan. Sebaliknya ketika hasil klasifikasi tidak sama dengan informasi kelas yang diberikan di awal, maka vektor pewakil akan disesuaikan agar menjauhi vektor masukan.

Perbandingan Algoritma

SOM

Algoritma metode Self-Organizing Maps dapat ditulis sebagai berikut :
Langkah 0 : Inisialisai Bobot
Langkah 1 : Jika kondisi henti gagal, lakukan langkah 2-7
Langkah 2 : Untuk setiap vektor masukan X_i, lakukan langkah 3 sampai 5
Langkah 3 : Untuk setiap j, hitung:

$$D_i = \sqrt{\sum_{j=1}^{m}(W_{ji} - X_i)^2}$$

Langkah 4 : Temukan indeks j sehingga D(j) minimum
Langkah 5 : Untuk setiap neuron J update bobotnya

$$W_{ji}\,(new) = W_{ji}\,(lama) + \alpha\,(X_i - W_{ji}\,(lama))$$

Dengan α adalah laju pemahaman / learning rate (digunakan 0.1)
Langkah 6 : Memodifikasi laju pemahaman (digunakan 0.5α)
Langkah 7 : Periksa kondisi henti

LVQ

Algoritma Learning Vector Quantization dapat ditulis sebagai berikut :
Langkah 0 : Inisialisai Bobot
Langkah 1 : Jika kondisi henti gagal, lakukan langkah 2-8
Langkah 2 : Untuk setiap vektor masukan X_i, lakukan langkah 3 sampai 6
Langkah 3 : Untuk setiap j, hitung:

$$D_i = \sqrt{\sum_{j=1}^{m}(W_{ji} - X_i)^2}$$

Langkah 4 : Temukan indeks j sehingga D(j) minimum
Langkah 5 : Periksa indeks j dan bandingkan dengan informasi kelas
Langkah 6 : Untuk setiap j

- Meng-update bobotnya jika indeks = informasi kelas

$$W_{ji}\,(new) = W_{ji}\,(lama) + \alpha\,(X_i - W_{ji}\,(lama))$$

Meng-update bobotnya jika indeks ≠ informasi kelas

$$W_{ji}\,(new) = W_{ji}\,(lama) - \alpha\,(X_i - W_{ji}\,(lama))$$

Dengan α adalah laju pemahaman / learning rate (digunakan 0.1)

Langkah 7 : Memodifikasi laju pemahaman (digunakan 0.5α)

Langkah 8 : Periksa kondisi henti

4. Hasil dan Pembahasan

Pengembangan Program SOM

Program dikembangkan dengan menggunakan MATLAB. Data disimpan dalam format txt dengan nama data.txt. Program yang dikembangkan akan membaca data dan kemudian megidentifikasi Berikut adalah hasil pengembangannya :

```
% PROGRAM SOM %

% Catatan : ubah nama file
dimana data tersimpan misal
data.txt

clc;clear;
tic;
disp('mulai')
% ---------------------
Membaca data -----------------
----- %
fid = fopen('data.txt','r');
i=1;
a=0;
data=0;
while 1
    tline = fgetl(fid);
    if ~ischar(tline),
break,   end
   %disp(tline);
   a=str2num(tline);
   [sizex sizey]=size(a);
   for j=1:sizey
       data(i,j)=a(j);
   end
   i=i+1;
end
fclose(fid);

%-----------------------
Membaca Target ---------------
--------------%
[sizex sizey]=size(data);
batas_input=sizey-1;
for i=1:sizex
    for j=1:batas_input
        x(i,j)=data(i,j);
    end
    t(i,1)=data(i,sizey);
end
x;
t;

%--------- Inisialisasi
variabel awal untuk SOM ------
---%
n= batas_input;      % jml
neuron masukan
p= max(t);           % jml
neuron keluaran / banyaknya
kelas
ndc=sizex/p;         %jml data
tiap kelas
alpha=0.6;             % laju
pemahaman
batas_perubahan=0.00001;    %
batas perubahan bobot

%----------------------
inisialisasi bobot vji--------
----------------%
% bobot diambil dengan
kelipatan ndc / data pertama
tiap kelas
for j=1:p
    v(j,:)=x((((j-
1)*ndc)+1),:);
end
v;

%-----------------
Pembelajaran self organizing
maps --------------------%
epoh=100;
nTraining=(0.5*sizex); %
persentase data dr 150 data
yang akan digunakan sebagai
data learning
for it=1:epoh
    k=1;
    v_lama=v;
    for loop=1:nTraining
        X=x(k,:);
        min=inf;
        for j=1:p
            d(j)=0;
            % menghitung jarak
antara vji dengan xi
            for i=1 :n
```

```
                d(j)= d(j) +
((X(1,i)-v(j,i))^2);
            end
            d(j)=sqrt(d(j));
            % memeriksa vji
mana yang memiliki jarak yang
paling dekat dengan xi
            if( d(j)<min)
                    min=d(j);
                    J=j;
            end
        end
        % hitung nilai
pengubah vektor pewakil/bobot
        delta_v=alpha*(X-
v(J,:));
        % update vektor bobot
yang paling dekat dengan X
        vn=v(J,:)+delta_v;
        v(J,:)=vn;

        % urutan data yg
digunakan%
        k=k+ndc;
        if(k>sizex)
            k=k-sizex+1;
        end
    end
    % uji kondisi
penghentian(selisih vji saat
ini dengan vji iterasi/epoh
sebelumnya)
    dv=abs(v-v_lama);
    % cek apakah ada data
selisih (komponen dv) yang
lebih dari batas perubahannya
    cek_dv=0;
    for i=1:n
        for j=1:p
            if
(dv(j,i)>batas_perubahan)

cek_dv=cek_dv+1;
            end
        end
    end
    % jika tidak ada perubahan
maka hentikan looping iterasi
    if (cek_dv==0)
        break
    end
    % modifikasi laju
pemahaman
    alpha= alpha*0.5;
end
% cek epoch dan bobot terakhir
it;
v;
%----------------- Akhir
Training Self Organizing Maps
---------------------%

%----------------- Testing
Self Organizing Maps ---------
------------%

% Inisialisasi variabel
testing
k;                          %
awal data testing
betul=0;                    %
konter penghitung data yang
teridentifikasi dengan benar
nTest=(sizex-nTraining);    %
persentase data dr 150 data
yang akan digunakan sebagai
data test
in=1;
yang_salah=[];
% Proses testing
for loop=1:nTest
    X=x(k,:);
    T=t(k);
    min=inf;
    for j=1:p
        % menghitung jarak
antara vji dengan xi
        d(j)=0;
        for i=1 :n
            d(j)= d(j) +
((v(j,i)- X(1,i))^2);
        end
        d(j)=sqrt(d(j));
        % memeriksa vji mana
yang memiliki jarak yang
paling dekat dengan xi
        if( d(j)<min)
                min=d(j);
                J=j;
        end
    end
    % ------- cek kesesuaian
klasifikasi ------------- %
    if (J==T)
        betul=betul+1;
    else
        yang_salah(in,:)=[k J
T];
        in=in+1;
    end
```

```
    % urutan data yg
digunakan%
    k=k+ndc;
    if(k>sizex)
        k=k-sizex+1;
    end
end

it
nTest
betul
RR=(betul/nTest)*100
yang_salah;

disp('selesai')
toc;
```

Pengembangan Program LVQ

Program dikembangkan dengan menggunakan MATLAB. Data disimpan dalam format txt dengan nama data.txt. Program yang dikembangkan akan membaca data dan kemudian megidentifikasi Berikut adalah hasil pengembangannya :

```
% PROGRAM LVQ %
% Catatan : ubah nama file
dimana data tersimpan misal
data.txt
clc;clear;
tic;
disp('mulai')
% ---------------------
Membaca data ------------------
----- %
fid = fopen('data.txt','r');
i=1;
a=0;
data=0;
while 1
    tline = fgetl(fid);
    if ~ischar(tline),
break,   end
   %disp(tline);
   a=str2num(tline);
   [sizex sizey]=size(a);
   for j=1:sizey
       data(i,j)=a(j);
   end
   i=i+1;
end
fclose(fid);
%-----------------------
Membaca Target ---------------
--------------%
[sizex sizey]=size(data);
batas_input=sizey-1;
for i=1:sizex
    for j=1:batas_input
        x(i,j)=data(i,j);
    end
    t(i,1)=data(i,sizey);
end
x;
t;
%--------- Inisialisasi
variabel awal untuk LVQ ------
---%
n= batas_input;      % jml
neuron masukan
p= max(t);           % jml
neuron keluaran / banyaknya
kelas
ndc=sizex/p;         % jml
data tiap kelas
alpha=0.000001;           %
laju pemahaman
batas_perubahan=0.00001;    %
batas perubahan bobot
%----------------------
inisialisasi bobot vji--------
----------------%
% bobot diambil dengan
kelipatan ndc / data pertama
tiap kelas
for j=1:p
    v(j,:)=x((((j-
1)*ndc)+1),:);
end
v;
%-----------------
Pembelajaran LVQ -------------
--------%
epoh=100;
nTraining=(0.5*sizex); %
persentase data dr 150 data
yang akan digunakan sebagai
data learning
for it=1:epoh
    k=1;
    v_lama=v;
    for loop=1:nTraining
        X=x(k,:);
        T=t(k);
        min=inf;
        for j=1:p
            d(j)=0;
            for i=1 :n
                d(j)= d(j) +
((X(1,i)-v(j,i))^2);%
menghitung jarak antara vji
dengan xi
```

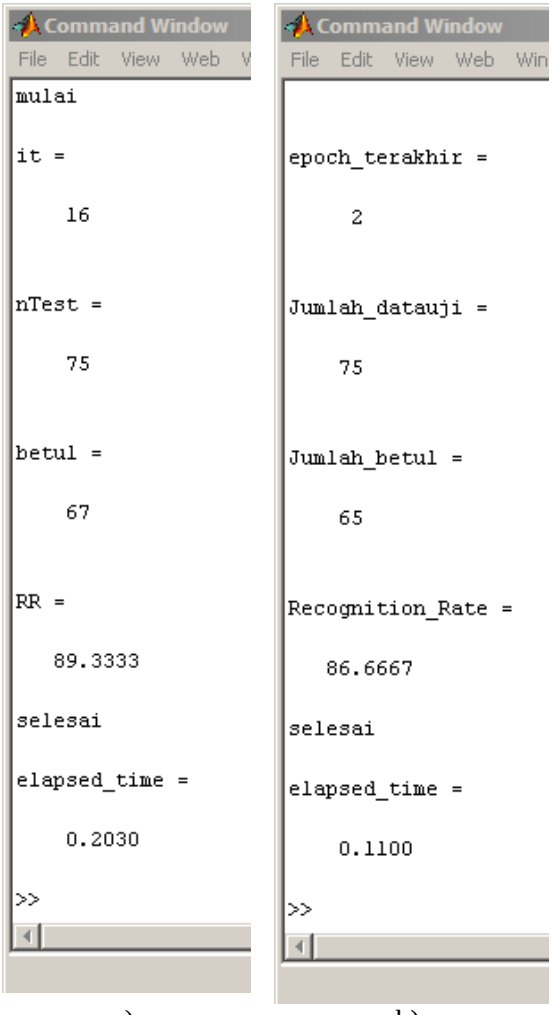
```
            end
            d(j)=d(j)^0.5;
            if( d(j)<min)
                    min=d(j);
                    J=j;
            end
        end
        % hitung nilai
pengubah vektor pewakil/bobot
        delta_v=alpha*(X-
v(J,:));
        % cek info kelas dan
update bobot
        if (J==T)
            vn=v(J,:)+delta_v;
        else
            vn=v(J,:)-delta_v;
        end
        v(J,:)=vn;
        % urutan data yg
digunakan
        k=k+ndc;
        if(k>sizex)
            k=k-sizex+1;
        end
    end
    % uji kondisi
penghentian(selisih vji saat
ini dengan vji iterasi/epoh
sebelumnya)
    for j=1:p
           for i=1:n

dv(j,i)=abs(v(j,i)-
v_lama(j,i));
           end
    end
    % cek apakah ada data
selisih (komponen dv) yang
lebih dari batas perubahannya
    cek_dv=0;
    for i=1:n
        for j=1:p
            if
(dv(j,i)>batas_perubahan)

cek_dv=cek_dv+1;
            end
        end
    end
    % jika tidak ada perubahan
maka hentikan looping iterasi
    if (cek_dv==0)
        break
    end
    % modifikasi laju
pemahaman
    alpha= alpha*0.5;
end
% cek epoch dan bobot terakhir
it;
v;
%--------------akhir LVQ------
------------%
%----------------- Testing LVQ
---------------------%
% Inisialisasi variabel
testing
k                           %
awal data testing
betul=0;                     %
konter penghitung data yang
teridentifikasi dengan benar
nTest=(sizex-nTraining);    %
persentase data dr 150 data
yang akan digunakan sebagai
data test
in=1;
yang_salah=[];
% Proses testing
for loop=1:nTest
    X=x(k,:);
    T=t(k);
    min=inf;
    for j=1:p
        % menghitung jarak
antara vji dengan xi
        d(j)=0;
        for i=1 :n
            d(j)= d(j) +
((v(j,i)- X(1,i))^2);
        end
        d(j)=d(j)^0.5;
        % memeriksa vji mana
yang memiliki jarak yang
paling dekat dengan xi
        if( d(j)<min)
                min=d(j);
                J=j;
        end
    end
    % ------- cek kesesuaian
klasifikasi ------------- %
    if (J==T)
        betul=betul+1;
    else
        yang_salah(in,:)=[k J
T];
        in=in+1;
    end
```

```
    % urutan data yg
digunakan%
    k=k+ndc;
    if(k>sizex)
        k=k-sizex+1;
    end
end
epoch_terakhir=it
Jumlah_datauji=nTest
Jumlah_betul=betul
Recognition_Rate=(betul/nTest)
*100
yang_salah;
end
disp('selesai')
toc;
```

HASIL PENGUJIAN



a) b)

Gambar 4. Hasil Perbandingan 2 Metode a) SOM b) LVQ

Dari hasil pengujian di atas terlihat bahwa SOM memiliki tingkat pengenalan yang paling baik (96%) dibandingkan dengan LVQ (86.67%). Namun pencapaian recognition rate yang tinggi dari proses SOM (0.20 s) dan LVQ (0.11 s).

5. Kesimpulan

Dari kedua metode yang telah dilakukan, maka dapat diambil kesimpulan bahwa :

1. Dari kedua metode yang diuji, untuk mendeteksi data bunga iris, metode SOM memiliki tingkat pengenalan yang lebih baik.
2. Sedangkan jika dilihat dari segi waktu proses, metode LVQ memiliki waktu proses yang lebih cepat

Daftar Pustaka

Kusumoputro , Benyamin, *Jaringan Neural Network*. Depok: Universitas Indonesia

Putra, Dwi Sudarno. 2011. *Pengembangan Jaringan Syaraf Tiruan dengan Metode SOM Fuzzy dan LVQ Fuzzy*. Depok : Universitas Indonesia

Siang , Jong Jek. 2005. *Jaringan Syaraf Tiruan & Pemrogramannya Menggunakan Matlab*. Yogyakarta : Andi,