LAPORAN TUGAS
JARINGAN SYARAF TIRUAN

"Implementasi Clustering dengan menggunakan Kohonen SOM"

Nama Mahasiswa :

Muhammad Syahrul Romadhon (06111740000078)

Dosen :

Prof. DR. Mohammad Isa Irawan, MT

19631225 198903 1 001

DEPARTEMEN MATEMATIKA
FAKULTAS SAINS ANALITIKA DATA
(FSAD)

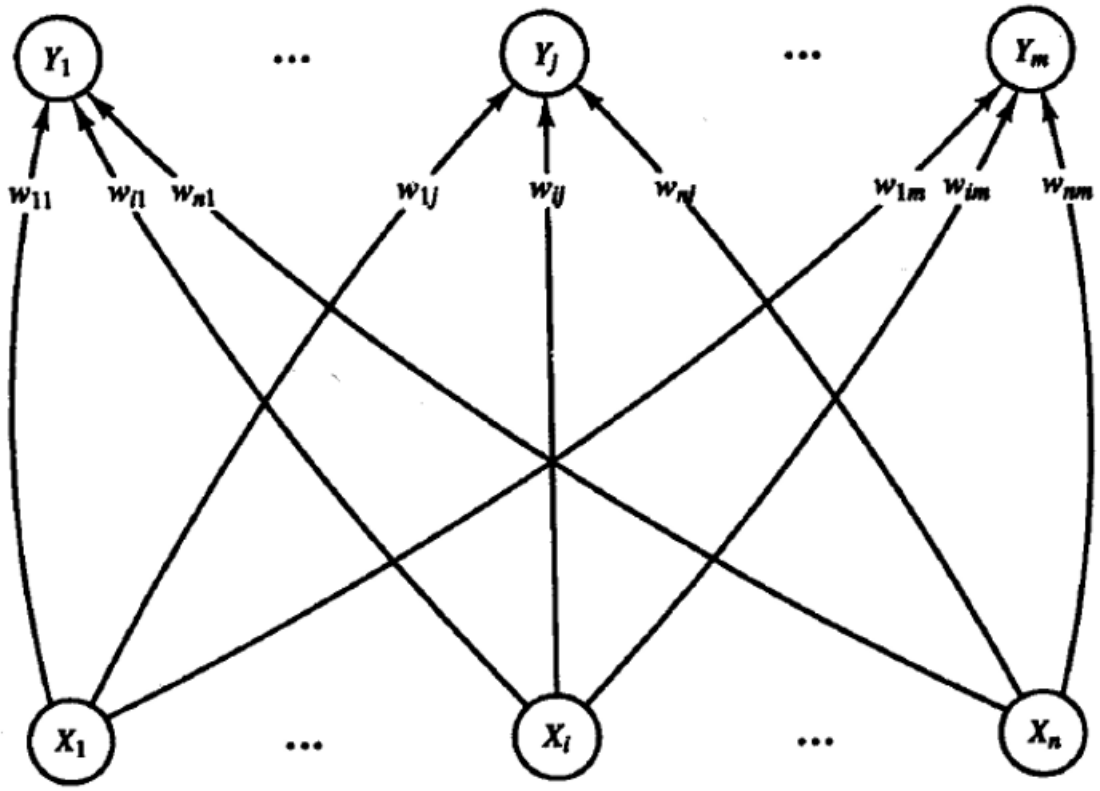
INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2020

I. Deskripsi

Arsitektur Kohonen SOM, sebagai berikut:



Algoritma Kohonen SOM, seperti berikut:

Step 0. Initialize weights w_{ij}. (Possible choices are discussed below.)
Set topological neighborhood parameters.
Set learning rate parameters.

Step 1. While stopping condition is false, do Steps 2–8.

Step 2. For each input vector **x**, do Steps 3–5.

Step 3. For each j, compute:

$$D(j) = \sum_i (w_{ij} - x_i)^2.$$

Step 4. Find index J such that $D(J)$ is a minimum.

Step 5. For all units j within a specified neighborhood of J, and for all i:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})].$$

Step 6. Update learning rate.

Step 7. Reduce radius of topological neighborhood at specified times.

Step 8. Test stopping condition.

Akan dilakukan Implementasi Clustering menggunakan Kohonen SOM dengan Data sebagai berikut:

Tabel2 Hasil Pengkodean Data

Nama_daerah	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}
Kdy. Madiun	0	0	0	0	0	0	0	1	1	0	1	1	0
Pacitan	0	0	0	0	0	0	0	1	0	1	1	1	0
Ponorogo	1	0	0	0	0	0	0	0	1	1	1	0	0
Trenggalek	0	0	0	1	0	1	1	1	1	0	1	1	0
Tulungagung	1	0	1	0	0	0	0	0	1	0	1	1	0
Blitar	1	0	0	0	1	0	0	0	0	0	1	1	0
Kediri	1	1	1	0	0	1	0	0	0	0	1	1	0
Malang	1	1	1	0	0	1	0	0	0	1	1	1	0
Lumajang	1	0	1	0	0	0	0	1	1	0	1	1	0
Jember	1	1	1	1	0	1	0	0	0	0	0	1	0
Banyuwangi	1	1	1	0	0	1	0	0	0	1	0	1	0
Bondowoso	0	0	0	0	0	0	0	1	1	1	1	1	0
Situbondo	0	0	0	0	0	0	0	1	0	0	1	1	0
Probolinggo	1	0	0	0	0	0	0	0	1	0	1	1	0
Pasuruan	1	1	1	0	0	1	0	0	0	0	1	1	0
Sidoarjo	1	1	1	1	1	1	0	0	1	0	1	1	0
Mojokerto	1	1	0	0	0	0	0	1	1	0	1	1	0
Jombang	1	0	0	0	1	0	0	0	1	0	1	1	0
Nganjuk	1	0	1	0	0	0	0	0	1	1	1	1	0
Madiun	0	0	0	0	0	0	0	0	1	0	1	1	0
Magetan	0	0	1	0	1	0	0	0	1	0	1	1	0
Ngawi	0	0	0	0	0	0	0	0	1	0	1	1	0
Bojonegoro	1	0	1	0	1	0	0	0	1	1	1	1	0
Tuban	1	1	1	0	0	1	0	0	0	0	1	1	0
Lamongan	1	0	0	0	0	0	0	0	0	0	1	1	0
Gresik	1	1	0	0	1	1	0	1	1	0	1	1	0
Bangkalan	0	0	1	0	0	0	0	1	1	0	0	1	0
Sampang	0	0	0	0	0	0	0	1	1	0	1	1	0
Pamekasan	0	0	1	0	0	0	0	0	1	0	1	1	0
Sumenep	1	0	0	0	0	1	0	0	0	0	1	1	0
Kdy. Kediri	0	0	0	0	1	0	0	1	1	0	1	1	0
Kdy. Blitar	0	0	0	0	0	0	0	1	1	0	1	1	0
Kdy. Malang	0	1	1	1	1	1	0	1	1	1	1	1	0
Kdy. Probolinggo	0	0	0	0	0	0	0	1	1	0	1	1	0
Kdy. Pasuruan	0	0	0	0	0	0	0	1	1	0	1	1	0
Kdy. Mojokerto	0	0	0	0	0	0	0	1	1	0	1	1	0

Dan inisialisasi weight setiap elemennya bernilai random antara 0 sampai 1.

Lalu dengan Radius (Neighborhood) sama dengan 0.

Dan learning rate (alpha) sama dengan 0.6

II. Source Code

```
# -*- coding: utf-8 -*-

import numpy as np
import pandas as pd
import random

print("@author: Muhammad Syahrul Romadhon (06111740000078)")

# Searching Function index distance minimal
```

```
def imin(m):
    a = m[0]
    index = 0
    for i in range(len(m)):
        if m[i-1] > m[i]:
            b = m[i]
            if a > b:
                index = i
    return index

# Data


A = pd.read_csv("data.csv")
print("DATA:\n", A.head(),
"\n=================================================================
=========")
# A = np.array(A)
# print(A)

# Initialization
c = 36
m = 4
n = 13
r = 0
alpha = 0.6

# weight
w = np.zeros((n, m), dtype=float)
for j in range(m):
    for i in range(n):
        w[i, j] = '{:03.1f}'.format(random.uniform(0, 1))
print("WEIGHT:\n", w, "\n====================")

# Distance Euclidian
print("DISTANCE:")
for i in range(c):
    print("For Input vector", A.loc[i][0])
    D = np.zeros([w.shape[1]])
    for k in range(m):
        for j in range(n):
            D[k] = D[k] + (w[j, k] - A.loc[i][j+1]) ** 2
    print(D)
    index = imin(D)
    print("Winning cluster is D(", index, ")")
    print("================================================")
    print("Update Weight (New Weight) column", index)
    for j in range(w.shape[0]):
        # print(A.loc[j][i+1])
        w[j, index] = (1-alpha)*w[j, index] + alpha*A.loc[i][j+1]
    alpha = alpha*0.5
    print(w)
    print("================================================")
```

III. Hasil (Output)

- Weight

```
WEIGHT:
 [[0.7 0.8 0.6 0.6]
 [0.5 0.9 0.1 0. ]
 [0.4 0.8 0.4 0.1]
 [0.4 0.3 0.7 0.7]
 [0.9 0.1 0.8 0.3]
 [0.1 0.5 0.8 0.1]
 [1.  0.7 0.2 0.2]
 [0.2 0.7 0.6 0.9]
 [0.3 0.3 0.9 0.5]
 [0.1 0.9 0.2 0.9]
 [0.6 0.7 0.5 0.4]
 [0.7 0.3 0.5 0.3]
 [0.5 0.  0.3 0.9]]
```

- Sample Distance

```
DISTANCE:
For Input vector Kdy. Madiun
[4.52 4.9  3.14 3.73]
Winning cluster is D( 2 )
=====================================
Update Weight (New Weight) column 2
[[0.7  0.8  0.24 0.6 ]
 [0.5  0.9  0.04 0.  ]
 [0.4  0.8  0.16 0.1 ]
 [0.4  0.3  0.28 0.7 ]
 [0.9  0.1  0.32 0.3 ]
 [0.1  0.5  0.32 0.1 ]
 [1.   0.7  0.08 0.2 ]
 [0.2  0.7  0.84 0.9 ]
 [0.3  0.3  0.96 0.5 ]
 [0.1  0.9  0.08 0.9 ]
 [0.6  0.7  0.8  0.4 ]
 [0.7  0.3  0.8  0.3 ]
 [0.5  0.   0.12 0.9 ]]
```