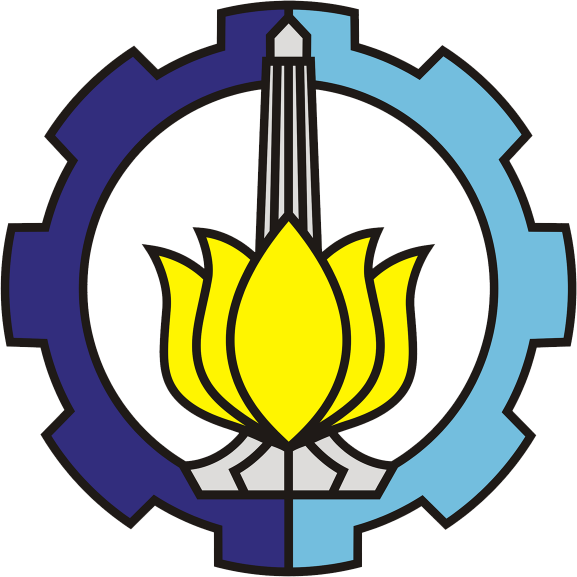
**LAPORAN TUGAS**

**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)**

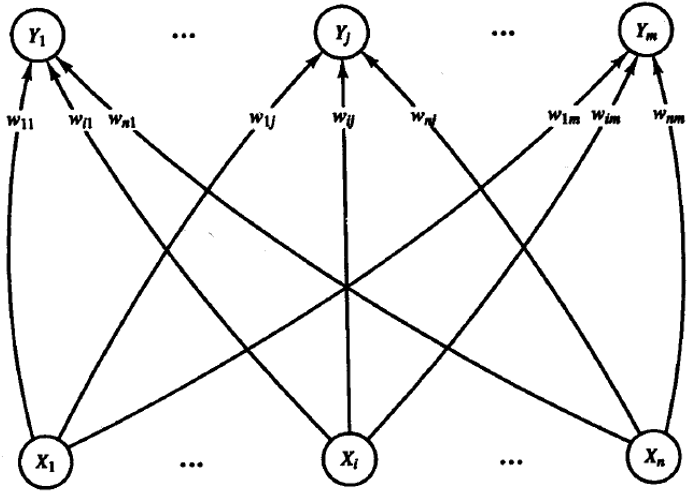
**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA**

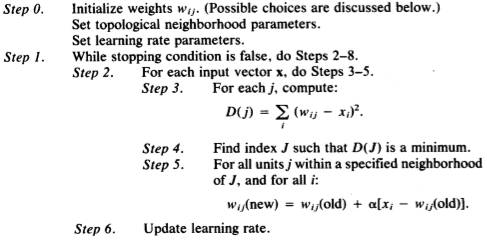
**2020**

1. **Deskripsi**

Arsitektur Kohonen SOM, sebagai berikut:

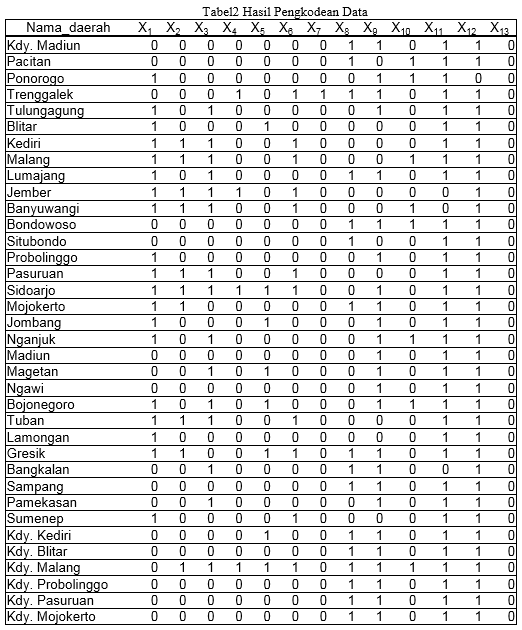


Algoritma Kohonen SOM, seperti berikut:





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



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

1. **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**]**

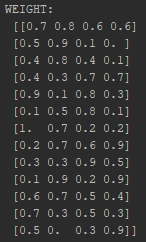
alpha **=** alpha**\***0.5

**print(**w**)**

**print(**"=================================================="**)**

1. **Hasil (Output)**

* Weight



* Sample Distance

