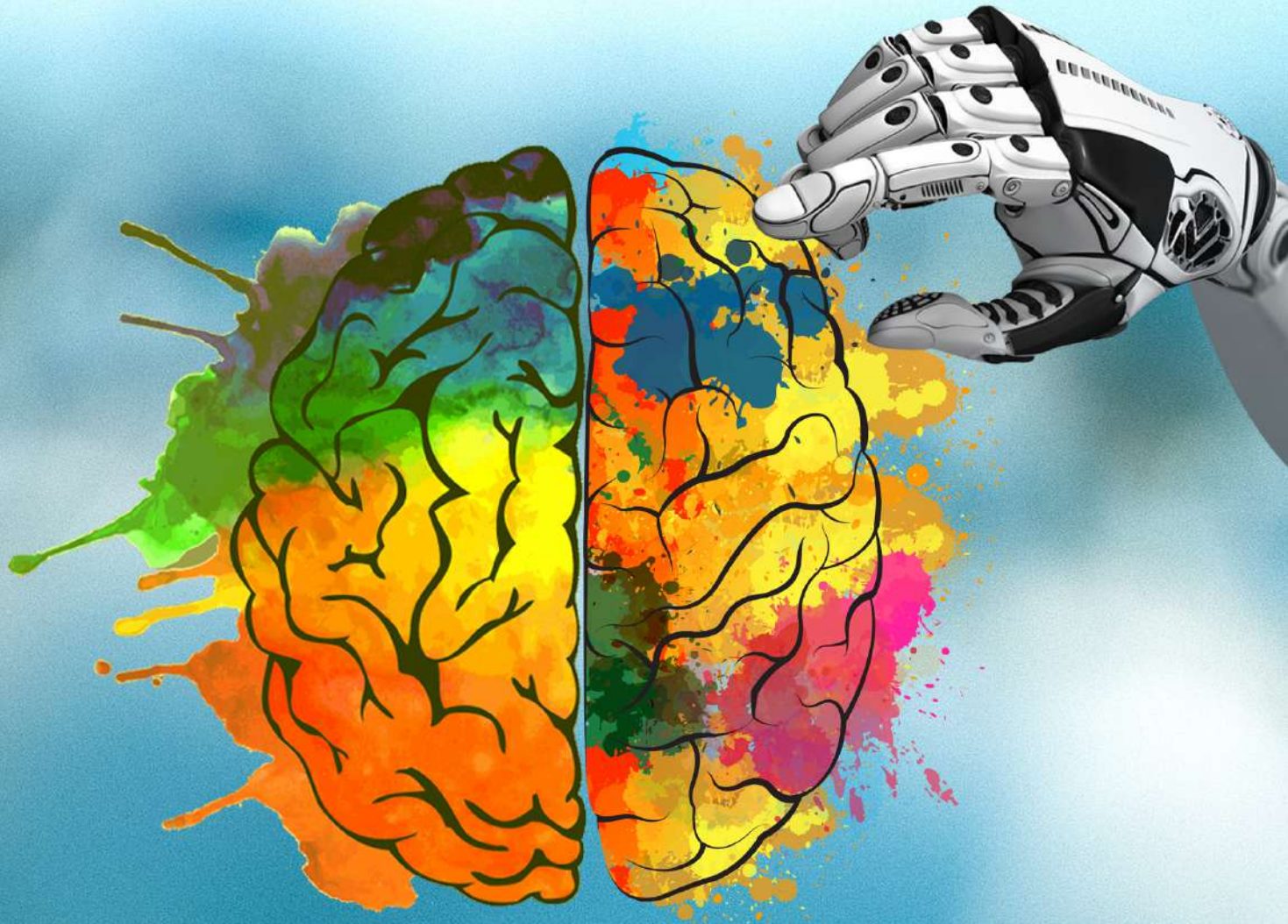
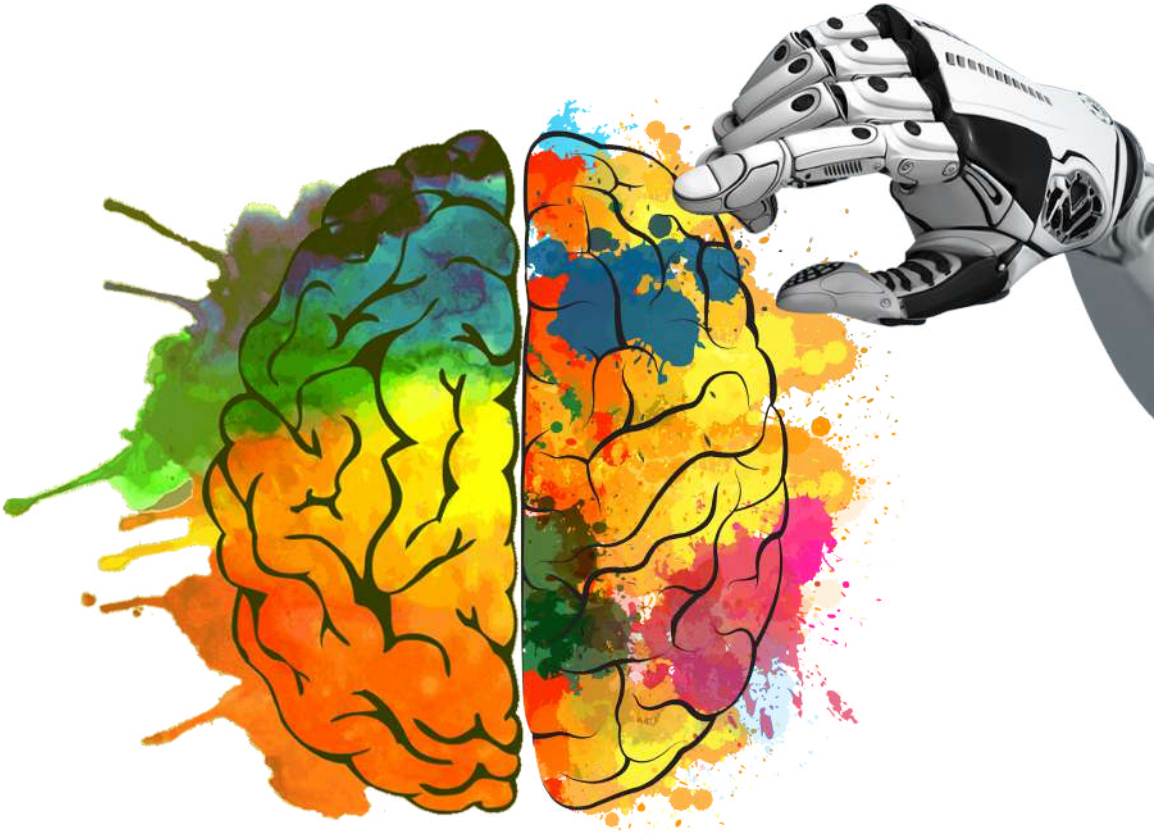


JARINGAN Saraf Tiruan

Algoritma Prediksi & Implementasi

•Agus Perdana Windarto •Darmeli Nasution •Anjar Wanto
•Frinto Tambunan •Muhammad Said Hasibuan
•Muhammad Noor Hasan Siregar •Muhammad Ridwan Lubis
•Solikhun •Yusra Fadhillah •Dicky Nofriansyah



JARINGAN Saraf Tiruan

Algoritma Prediksi & Implementasi

UU 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Perlindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

a. penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
b. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
c. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
d. penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah).

Jaringan Saraf Tiruan: Algoritma Prediksi dan Implementasi

Penulis:

Agus Perdana Windarto, Darmeli Nasution
Anjar Wanto, Frinto Tambunan, Muhammad Said Hasibuan
Muhammad Noor Hasan Siregar, Muhammad Ridwan Lubis
Solikhun, Yusra Fadhillah, Dicky Nofriansyah

Penerbit Yayasan Kita Menulis

Jaringan Saraf Tiruan:
Algoritma Prediksi dan Implementasi

Copyright © Yayasan Kita Menulis, 2020

Penulis:
Agus Perdana Windarto, Darmeli Nasution, Anjar Wanto
Frinto Tambunan, Muhammad Said Hasibuan
Muhammad Noor Hasan Siregar, Muhammad Ridwan Lubis
Solikhun, Yusra Fadhillah, Dicky Nofriansyah

Editor: Janner Simarmata
Desain Cover: Tim Kreatif Kita Menulis

Penerbit
Yayasan Kita Menulis
Web: kitamenulis.id
e-mail: press@kitamenulis.id
Kontak WA: +62 821-6453-7176

Agus Perdana Windarto, dkk.
Jaringan Saraf Tiruan: Algoritma Prediksi dan Implementasi
Yayasan Kita Menulis, 2020
xviii; 234 hlm; 16 x 23 cm
ISBN: 978-623-7645-83-2 (print)
E-ISBN: 978-623-7645-84-9 (online)
Cetakan 1, Mei 2020
I. Jaringan Saraf Tiruan: Algoritma Prediksi dan Implementasi
II. Yayasan Kita Menulis

Katalog Dalam Terbitan

Hak cipta dilindungi undang-undang
Dilarang memperbanyak maupun mengedarkan buku tanpa
Ijin tertulis dari penerbit maupun penulis

Kata Pengantar

Puji syukur kehadirat Tuhan Yang Maha Esa yang telah memberikan taufik dan hidayahnya, sehingga kami mampu menyelesaikan buku Jaringan Saraf Tiruan: Algoritma Peramalan/ Prediksi dan Implementasi ini. Neural Network (Jaringan Saraf Tiruan) merupakan topik yang hangat dibicarakan dan mengundang banyak kekaguman dalam beberapa tahun terakhir. Hal ini disebabkan karena kemampuan JST untuk meniru sifat sistem yang diinputkan.

Maksud dan tujuan penulisan buku ini adalah untuk memberikan pemahaman tentang Jaringan Saraf Tiruan sebagai disiplin ilmu yang berkembang pesat dalam berbagai bidang aplikasi terutama pada bidang otomatisasi. Disamping itu harapan penulis adalah agar buku ini dapat memberikan nilai positif berupa pengetahuan Jaringan Saraf Tiruan dengan berbagai penerapan implementasi menggunakan bantuan perangkat lunak. Penyajian yang disampaikan kepada pembaca merupakan pengalaman menulis dari penulis yang dibantu dengan beberapa sumber baik literatur maupun internet yang mengarah kepada pokok pembahasan buku.

Secara garis besar, buku ini terdiri dari 10 (sepuluh) bab, yaitu :

Bab 1 Optimasi PSO (Particle Swarm Optimization) pada Algoritma Backpropagation
Bab 2 Algoritma Perseptron dan Penerapan Aplikasi
Bab 3 Algoritma Conjugate Gradient Polak Rebiere untuk Prediksi Data
Bab 4 Algoritma Habb dan Penerapan
Bab 5 Prediksi Gaya Belajar VARK dengan Artificial Neural Network
Bab 6 Prediksi dengan Algoritma Levenberg-Marquardt
Bab 7 Prediksi dengan Quantum Neural Network
Bab 8 Penerapan Quantum Perceptron Dalam Klasifikasi Data
Bab 9 Fuzzy Neural Network, (Application System Control)
Bab 10 Algoritma Kohonen Self Organizing Map (SOM)

Akhir kata penulis mengucapkan banyak terima kasih kepada teman-teman sejawat yang telah memberikan masukan-masukan positif selama penulisan buku ini.

Medan, Maret 2020

Penulis

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab 1

Optimasi PSO (Particle *Swarm* Optimization) pada Algoritma *Backpropagation*

1.1 Pendahuluan

Peramalan (*forecasting*) merupakan masalah yang sering dihadapi dalam berbagai bidang persoalan seperti peramalan cuaca dan iklim, peramalam finansial, peramalan pergeseran harga saham dan lain sebagainya (Utomo, 2015). Salah satu faktor terpenting dalam pemilihan metode adalah keakuratan peramalan (Sahagun, Cruz dan Garcia, 2017). Banyak metode yang digunakan untuk menyelesaikan kasus peramalan (Maulana dan Kumalasari, 2019; Ogie, Rho dan Clarke, 2019). Metode yang paling banyak digunakan dalam melakukan peramalan adalah jaringan saraf tiruan (*artificial neural network*) (Koesriputranto, 2015) karena pemodelan ini bersifat dinamis dan *real time* (Sahagun, Cruz dan Garcia, 2017).

Jaringan saraf tiruan juga merupakan metode peramalan yang memiliki tingkat *error* data yang cukup rendah dan cukup baik dalam proses generalisasi karena didukung oleh data *training* yang cukup dan proses pembelajaran yang menyesuaikan bobot sehingga model ini mampu untuk meramalkan data *time series* untuk beberapa periode waktu ke depan (Nugraha dan SN, 2014). Salah satu metode yang digunakan pada jaringan saraf tiruan dalam melakukan peramalan adalah algortima *backpropagation* (Sumijan *et al.*, 2016; Budiharjo *et al.*, 2018a, 2018b). Namun pada penerapannya algortima ini masih memiliki kelemahan di antaranya masalah *overfitting* pada jaringan saraf tiruan (Truatmoraka, Waraporn dan Suphachotiwatana, 2016), masalah waktu pelatihan yang lama untuk mencapai konvergen (Adnan *et al.*, 2012) dan

proses penentuan parameter (*learning rate* dan *momentum*) yang tepat dalam proses pelatihan (Ruslan, Zakaria dan Adnan, 2013). Untuk mengatasi masalah tersebut maka solusi yang diberikan adalah dengan menggunakan *Particle Swarm Optimization* (PSO) yang merupakan salah satu teknik kecerdasan buatan terbaik untuk optimasi dan perkiraan parameter (Nugraha dan SN, 2014).

Pada bab ini akan dibahas bagaimana hasil peramalan dengan optimasi (*particle Swarm optimization* + *backpropagation*) dan tanpa menggunakan optimasi (*backpropagation)*. Proses dilakukan dengan menggunakan bantuan *software* RapidMiner 5.3.015.

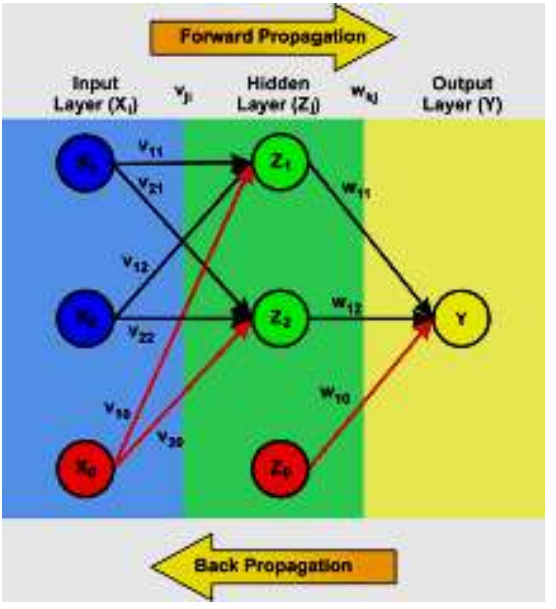
Gambar 1.1: *Software* RapidMiner 5.3.015.

1.2 Algortima *Backpropagation*

1.2.1 Pendahuluan

Backpropagation merupakan salah satu algoritma pembelajaran dalam jaringan saraf tiruan (Amrin, 2016). Proses pembelajaran *backpropagation* dilakukan dengan penyesuaian bobot-bobot jaringan saraf tiruan dengan arah mundur berdasarkan nilai *error* dalam proses pembelajaran (Windarto, Lubis dan Solikhun, 2018). Ciri khas *backpropagation* melibatkan tiga lapisan

(*layer*) utama: (1) lapisan masukan (*input layer*) berfungsi sebagai penghubung jaringan ke dunia luar (sumber data), (2) lapisan tersembunyi (*hidden layer*) di mana jaringan dapat memiliki lebih dari satu *hidden layer* atau bahkan bisa juga tidak memilikinya sama sekali; dan lapisan luaran (*output layer*) di mana hasil dari masukan yang diberikan oleh *input layer* (Wanto dan Windarto, 2017) dengan menggunakan fungsi *Sigmoid* (Khairani, 2014). Keluaran dari pada lapisan ini sudah dianggap sebagai hasil dari proses.



Gambar 1.2: Arsitektur *Algoritma Backpropagation*

Berdasarkan gambar 1.2, *Backpropagation* merupakan *supervised learning* (Febriadi *et al.*, 2018). Dikatakan *supervised learning* (Youllia Indrawaty, Asep Nana Hermana, 2012) karena teknik pembelajaran dilakukan dengan membuat fungsi dari data pelatihan untuk mempelajari fungsi pemetaan dari *input* ke *output*. Dengan kata lain *supervised learning* memiliki target (*Goal*) untuk memperkirakan fungsi pemetaannya, sehingga ketika kita mempunya *input* baru, algortima dapat memprediksi *output* untuk *input* tersebut (Windarto dan Lubis, 2016). Beberapa contoh algoritma regresi, SVM (*Support Vector Machine*), algoritma Hebbian (Hebb Rule), algoritma Perceptron, algoritma Adaline, algoritma Boltzman, algoritma Hapfield dan lainya (Budiharjo *et al.*, 2018a).

1.2.2 Kelebihan Algortima *Backpropagation*

Beberapa kelebihan Algortima *Backpropagation* adalah:

a) Dapat diaplikasikan pada penyelesaian suatu masalah berkaitan dengan identifikasi, prediksi, peramalan, pengenalan pola dan sebagainya (Dessy dan Irawan, 2012),
b) kemampuannya untuk belajar (bersifat adaptif) dan kebal terhadap kesalahan (*Fault Tolerance*) sehingga dapat mewujudkan sistem yang tahan kerusakan (*robust*) dan bekerja secara konsisten (Dessy dan Irawan, 2012),
c) Melatih jaringan untuk mendapat keseimbangan selama proses pelatihan sehingga dapat memberikan respon yang benar terhadap pola masukan yang serupa dengan pola yang dipakai selama pelatihan (Wanto *et al*., 2019).

1.2.3 Kekurangan Algortima *Backpropagation*

Beberapa kekurangan Algortima *Backpropagation* adalah:

a) Membutuhkan waktu cukup lama dalam proses pembelajaran untuk mencapai konvergen (Nugraha dan SN, 2014),
b) Parameter *learning rate* atau tingkat pembelajaran akan selalu berubah-ubah sesuai dengan kondisi perubahan error pada tiap iterasinya (Khairani, 2014),
c) Dalam menghitung perubahan bobot algoritma *backpropagation* dapat menyebabkan masalah lokal minimum sehingga tidak stabil (Suhendra dan Wardoyo, 2015).

1.2.4 Langkah Penyelesaian Algortima *Backpropagation*

Beberapa langkah penyelesaian algoritma *backpropagation* adalah:

Langkah 1 : Inisialisasi bobot dengan bilangan nilai acak kecil

Langkah 2 : Selama kondisi berhenti salah, kerjakan langkah 3 s.d. 8

Umpan Maju (*Feedforward*)

Langkah 3 : Tiap unit masukan (x_i, i=1,….., n) menerima isyarat masukan xi dan diteruskan ke unit-unit tersembunyi (*hidden layer*)

Langkah 4 : Tiap unit tersembunyi (z_j, z=1,…., p) menjumlahkan bobot sinyal input.

$$Z\_in_{jk} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (1.1)$$

dengan menerapakan fungsi aktivasi hitung:

$$Z_j = f(Z\_in_j) \quad (1.2)$$

misal, fungsi aktivasi yang digunakan adalah sigmoid:

$$y = f(x) = \frac{1}{1+e^{-x}} \quad (1.3)$$

dan mengirimkan isyarat ini ke semua unit pada unit keluaran

Langkah 5 : Tiap unit keluaran (y_k, k=1,….., m) menjumlahkan isyarat masukan berbobot

$$Y\_in_k = w_{0j} + \sum_{k=1}^{p} z_j v_{jk} \quad (1.4)$$

dengan menerpakan fungsi aktivasi hitung:

$$Y_j = f(Y\_in_k) \quad (1.5)$$

Perambatan Galat Mundur (*Backpropagation*)

Langkah 6 : Tiap unit keluaran (y_k, k=1,….., m) menerima pola pelatihan masukannya.

Hitung galat (error) informasinya:

$$\boldsymbol{\delta_k = (t_k - y_k)\, f'(y\_in_k)} \quad (1.6)$$

Hitung koreksi bobot dan biasnya:

$$\boldsymbol{\Delta w_{jk} = \alpha \delta_k x_j} \quad (1.7)$$

$$\boldsymbol{\Delta w_{0k} = \alpha \delta_k} \quad (1.8)$$

Langkah 7 : Tiap unit tersembunyi (z_j, z=1,….., p) menjumlahkan delta masukannya (dari unit yang berada pada lapisan atasanya).

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (1.9)$$

1.3 PSO (*Particle Swarm Optimization*)

Particle Swarm Optimization (PSO) merupakan salah satu teknik dasar dari *Swarm intelligence system* untuk menyelesaikan masalah optimasi dalam pencarian ruang sebagai suatu solusi (Mansur, Prahasto dan Farikhin, 2014) yang pertama kali diusulkan oleh James Kennedy dan Eberhart pad tahun 1995 (Nugraha dan SN, 2014).

Particle Swarm Optimization (PSO) didasarkan pada perilaku sekawanan burung atau ikan yang meniru perilaku sosial organisme ini. *Swarm intelligence system* melakukan penyebaran kecerdasan yang inovatif dalam menyelesaikan masalah optimasi dengan mengambil inspirasi dari contoh biologis, seperti fenomena kelompok (*Swarm*) pada hewan, di mana setiap kelompok memiliki prilaku individu dalam melakukan tindakan bersama untuk mencapai tujuan yang sama (Mansur, Prahasto dan Farikhin, 2014).

Gambar 1.3: Ilustrasi algoritma PSO yang terinspirasi dari perilaku kawanan (koloni) burung (*bird flocking*) atau ikan (*fish schooling)*

1.3.1 Parameter yang digunakan dalam proses algoritma PSO

Beberapa parameter yang digunakan dalam proses algoritma PSO (Mansur, Prahasto dan Farikhin, 2014):

a) Jumlah partikel (*Number of particles*) merupakan faktor yang dianggap sangat penting dalam melakukan penyelesaian masalah.
b) Bobot Inersia (*Inertia Weight*), memainkan peran yang sangat penting dalam kecepatan partikel dari algoritma PSO.
c) Faktor pembelajaran (*Learning factors*).

Parameter c1 merupakan pengakuan koefisien, sedangkan c2 adalah komponen social. Hal ini tidak terlalu penting untuk perilaku konvergensi dari algoritma PSO

d) Rentang dan dimensi partikel (*Range and dimension of particles*). Dimensi partikel dan rentang ditentukan berdasarkan masalah yang dioptimalkan.
e) Kecepatan (*Velocity*) merupakan perubahan maksimum pada setiap partikel, dapat diambil selama iterasi yang didefenisikan sebagai kecepatan maksimum.
f) Menghentikan kondisi (*Stopping condition*) merupakan salah satu cara apabila kriteria yang dicari sudah tercapai.

Istilah yang digunakan dalam penerapan algoritma PSO sebagai berikut (Nikentari *et al*., 2018):

a) *Swarm*: populasi dari sekawanan partikel.
b) *Particle*: individu pada suatu *Swarm*.
Setiap partikel mempresentasikan suatu solusi dari permasalahan yang diselesaikan.
c) *Pbest*: suatu partikel yang menunjukkan posisi terbaik.
d) *Gbest*: posisi terbaik dari seluruh partikel yang ada dalam suatu *Swarm*.
e) *Velocity*: kecepatan yang dimiliki oleh setiap partikel dalam menentukan arah perpindahan suatu partikel untuk memperbaiki posisi semula.
f) c1 dan c2: c1 merupakan konstanta pembelajaran kognitif, dan c2 konstanta pembelajaran sosial.

1.3.2 Langkah penyelesaian algoritma PSO

Ada 4 tahapan algoritma PSO yang digunakan untuk menyelesaikan masalah:

a) Initialization
b) Evaluation fungsi fitness
c) Update
d) Termination (Mansur, Prahasto dan Farikhin, 2014).

1.4 Implementasi *Backpropagation*

Sebelum menerapkan algoritma dengan menggunakan bantuan *software* RapidMiner 5.3.015, penulis menggunakan sampel data harga emas secara *times series* dari url: https://id.investing.com/commodities/gold-historical-data periode 1 Januari 2020 s/d 13 April 2020. Berikut data yang diolah dalam melakukan peramalan harga emas seperti pada tabel berikut:

Tabel 1.1: Data harga emas berkala periode 1 Januari 2020 s/d 13 April 2020

Tanggal	Terakhir	Pembukaan	Tertinggi	Terendah
01/01/2020	1.520,95	1.520,85	1.523,95	1.520,85
02/01/2020	1.540,10	1.535,50	1.545,60	1.532,00
03/01/2020	1.564,50	1.543,90	1.567,60	1.543,90
06/01/2020	1.580,90	1.577,90	1.600,00	1.575,00
07/01/2020	1.586,30	1.578,60	1.590,40	1.570,00
08/01/2020	1.572,30	1.590,30	1.624,50	1.565,90
09/01/2020	1.566,40	1.569,50	1.574,00	1.553,40
10/01/2020	1.572,10	1.566,50	1.575,30	1.559,00
13/01/2020	1.562,70	1.575,10	1.575,10	1.559,00
14/01/2020	1.556,50	1.560,10	1.561,40	1.548,40
15/01/2020	1.565,60	1.559,30	1.570,40	1.559,20
16/01/2020	1.562,20	1.567,70	1.569,70	1.560,00
17/01/2020	1.572,30	1.564,80	1.572,90	1.561,40
19/01/2020	1.557,05	1.557,40	1.559,05	1.556,45
20/01/2020	1.561,25	1.557,20	1.562,70	1.556,65
21/01/2020	1.569,90	1.570,40	1.580,70	1.557,80
22/01/2020	1.568,70	1.568,00	1.570,80	1.562,30
23/01/2020	1.577,30	1.572,00	1.579,30	1.563,70
24/01/2020	1.583,90	1.574,00	1.587,20	1.568,40
27/01/2020	1.589,50	1.592,50	1.599,70	1.587,50
28/01/2020	1.581,70	1.592,40	1.592,90	1.576,50
29/01/2020	1.582,00	1.579,30	1.588,60	1.574,00
30/01/2020	1.595,10	1.587,30	1.596,50	1.583,10

Tanggal	Terakhir	Pembukaan	Tertinggi	Terendah
31/01/2020	1.593,80	1.584,90	1.601,20	1.581,20
03/02/2020	1.588,30	1.603,60	1.603,90	1.579,10
04/02/2020	1.561,40	1.588,40	1.589,60	1.558,70
05/02/2020	1.568,70	1.563,40	1.571,70	1.556,90
06/02/2020	1.575,80	1.564,50	1.577,60	1.562,00
07/02/2020	1.579,20	1.575,70	1.583,30	1.570,00
10/02/2020	1.585,40	1.581,60	1.586,30	1.577,80
11/02/2020	1.575,90	1.581,60	1.583,10	1.570,90
12/02/2020	1.577,30	1.576,70	1.579,20	1.570,50
13/02/2020	1.584,50	1.574,60	1.587,00	1.574,50
14/02/2020	1.591,90	1.585,30	1.593,60	1.581,90
16/02/2020	1.586,25	1.585,50	1.586,25	1.583,45
17/02/2020	1.587,05	1.586,15	1.588,45	1.581,85
18/02/2020	1.609,30	1.590,40	1.613,80	1.587,50
19/02/2020	1.617,50	1.610,20	1.621,50	1.608,10
20/02/2020	1.626,20	1.620,00	1.632,00	1.612,60
21/02/2020	1.654,60	1.628,30	1.657,40	1.627,60
24/02/2020	1.682,40	1.660,30	1.697,00	1.658,80
25/02/2020	1.655,80	1.670,00	1.670,00	1.632,60
26/02/2020	1.648,90	1.643,00	1.662,70	1.632,40
27/02/2020	1.648,20	1.649,90	1.668,20	1.642,40
28/02/2020	1.571,80	1.651,70	1.656,10	1.569,10
02/03/2020	1.599,50	1.598,40	1.616,50	1.581,20
03/03/2020	1.648,90	1.594,30	1.654,70	1.594,30
04/03/2020	1.647,20	1.643,00	1.658,60	1.637,40
05/03/2020	1.672,00	1.642,70	1.679,20	1.641,50
06/03/2020	1.676,10	1.676,90	1.696,50	1.646,20
09/03/2020	1.678,60	1.694,50	1.707,80	1.660,80
10/03/2020	1.663,30	1.683,40	1.683,50	1.644,50
11/03/2020	1.645,40	1.652,40	1.674,70	1.635,70

Tanggal	Terakhir	Pembukaan	Tertinggi	Terendah
12/03/2020	1.593,20	1.638,70	1.654,00	1.563,40
13/03/2020	1.519,50	1.581,00	1.600,40	1.507,00
16/03/2020	1.488,60	1.564,80	1.576,00	1.453,00
17/03/2020	1.528,10	1.505,80	1.556,40	1.468,60
18/03/2020	1.480,60	1.533,70	1.548,90	1.476,00
19/03/2020	1.482,30	1.489,70	1.505,20	1.462,80
20/03/2020	1.488,10	1.474,70	1.522,50	1.460,90
23/03/2020	1.572,70	1.508,40	1.574,80	1.489,40
24/03/2020	1.663,30	1.566,30	1.693,50	1.566,30
25/03/2020	1.634,30	1.665,50	1.698,00	1.609,00
26/03/2020	1.660,30	1.642,20	1.677,20	1.612,00
27/03/2020	1.654,10	1.653,40	1.661,00	1.630,70
30/03/2020	1.643,20	1.663,40	1.673,60	1.632,00
31/03/2020	1.596,60	1.643,40	1.645,60	1.588,20
01/04/2020	1.591,40	1.589,40	1.612,40	1.576,00
02/04/2020	1.637,70	1.602,40	1.645,60	1.595,20
03/04/2020	1.645,70	1.635,30	1.652,80	1.624,40
06/04/2020	1.693,90	1.647,70	1.715,80	1.638,20
07/04/2020	1.683,70	1.707,10	1.742,60	1.672,00
08/04/2020	1.684,30	1.678,70	1.695,80	1.670,70
09/04/2020	1.752,80	1.680,50	1.754,50	1.676,50
10/04/2020	1.740,60	1.740,60	1.740,60	1.740,60
12/04/2020	1.738,70	1.742,85	1.747,40	1.732,10

(Sumber: https://id.investing.com/commodities/gold-historical-data)

Berdasarkan tabel 1.1, data yang terdiri dari 76 *record* akan dinormalisasi dengan melakukan transformasi data ke dalam range 0.1 - 0.9, formula yang digunakan adalah:

$$X' = \frac{0{,}8\,(X - b)}{(a - b)} + 0{,}1 \qquad (1.10)$$

di mana:

X' = data hasil normalisasi

X = data asli/data awal

a = nilai maksimum data asli

b = nilai minimum data asli

Dalam kasus data ditransfirmasi ke dalam range 0.1-0.9 dengan menggunakan formula (1.10) karena jaringan menggunakan fungsi aktivasi *sigmoid biner* (*logsig*). Data akan dibagi menjadi 2 bagian, yakni data pelatihan dan data pengujian (Windarto, Lubis dan Solikhun, 2018). Dalam hal ini data pelatihan dan pengujian akan menggunakan sumber data yang sama (data yang tertera pada tabel 1.1). Berikut adalah hasil tranformasi data seperti yang ditunjukkan pada tabel berikut:

Tabel 1.2: Data tranformasi

Tanggal	Terakhir	Pembukaan	Tertinggi	Terendah
01/01/2020	0,180299	0,180033	0,188259	0,180033
02/01/2020	0,231111	0,218905	0,245705	0,209619
03/01/2020	0,295854	0,241194	0,304080	0,241194
06/01/2020	0,339370	0,331410	0,390050	0,323715
07/01/2020	0,353698	0,333267	0,364577	0,310448
08/01/2020	0,316551	0,364312	0,455058	0,299569
09/01/2020	0,300896	0,309121	0,321061	0,266401
10/01/2020	0,316020	0,301161	0,324511	0,281260
13/01/2020	0,291078	0,323980	0,323980	0,281260
14/01/2020	0,274627	0,284179	0,287629	0,253134
15/01/2020	0,298773	0,282056	0,311509	0,281791
16/01/2020	0,289751	0,304345	0,309652	0,283914
17/01/2020	0,316551	0,296650	0,318143	0,287629
19/01/2020	0,276086	0,277015	0,281393	0,274494
20/01/2020	0,287231	0,276484	0,291078	0,275025
21/01/2020	0,310182	0,311509	0,338839	0,278076
22/01/2020	0,306998	0,305141	0,312570	0,290017

Tanggal	Terakhir	Pembukaan	Tertinggi	Terendah
23/01/2020	0,329818	0,315755	0,335124	0,293731
24/01/2020	0,347330	0,321061	0,356086	0,306202
27/01/2020	0,362189	0,370149	0,389254	0,356882
28/01/2020	0,341493	0,369884	0,371211	0,327695
29/01/2020	0,342289	0,335124	0,359801	0,321061
30/01/2020	0,377048	0,356352	0,380763	0,345207
31/01/2020	0,373599	0,349983	0,393234	0,340166
03/02/2020	0,359005	0,399602	0,400398	0,334594
04/02/2020	0,287629	0,359270	0,362454	0,280464
05/02/2020	0,306998	0,292935	0,314959	0,275688
06/02/2020	0,325837	0,295854	0,330614	0,289221
07/02/2020	0,334859	0,325572	0,345738	0,310448
10/02/2020	0,351310	0,341227	0,353698	0,331144
11/02/2020	0,326103	0,341227	0,345207	0,312836
12/02/2020	0,329818	0,328226	0,334859	0,311774
13/02/2020	0,348922	0,322653	0,355556	0,322388
14/02/2020	0,368557	0,351045	0,373068	0,342023
16/02/2020	0,353566	0,351575	0,353566	0,346136
17/02/2020	0,355688	0,353300	0,359403	0,341891
18/02/2020	0,414726	0,364577	0,426667	0,356882
19/02/2020	0,436484	0,417114	0,447098	0,411542
20/02/2020	0,459569	0,443118	0,474959	0,423483
21/02/2020	0,534925	0,465141	0,542355	0,463284
24/02/2020	0,608690	0,550050	0,647430	0,546070
25/02/2020	0,538109	0,575788	0,575788	0,476551
26/02/2020	0,519801	0,504146	0,556418	0,476020
27/02/2020	0,517944	0,522454	0,571012	0,502554
28/02/2020	0,315224	0,527231	0,538905	0,308060
02/03/2020	0,388723	0,385804	0,433831	0,340166
03/03/2020	0,519801	0,374925	0,535191	0,374925

Tanggal	Terakhir	Pembukaan	Tertinggi	Terendah
04/03/2020	0,515290	0,504146	0,545539	0,489287
05/03/2020	0,581095	0,503350	0,600199	0,500166
06/03/2020	0,591973	0,594096	0,646103	0,512637
09/03/2020	0,598607	0,640796	0,676086	0,551376
10/03/2020	0,558010	0,611343	0,611609	0,508126
11/03/2020	0,510514	0,529088	0,588259	0,484776
12/03/2020	0,372007	0,492736	0,533333	0,292935
13/03/2020	0,176451	0,339635	0,391111	0,143284
16/03/2020	0,094461	0,296650	0,326368	0,000000
17/03/2020	0,199270	0,140100	0,274362	0,041393
18/03/2020	0,073234	0,214129	0,254461	0,061028
19/03/2020	0,077745	0,097380	0,138507	0,026003
20/03/2020	0,093134	0,057579	0,184411	0,020962
23/03/2020	0,317612	0,146998	0,323184	0,096584
24/03/2020	0,558010	0,300630	0,638143	0,300630
25/03/2020	0,481061	0,563847	0,650083	0,413930
26/03/2020	0,550050	0,502023	0,594892	0,421891
27/03/2020	0,533599	0,531741	0,551907	0,471509
30/03/2020	0,504677	0,558275	0,585340	0,474959
31/03/2020	0,381028	0,505207	0,511045	0,358740
01/04/2020	0,367231	0,361924	0,422952	0,326368
02/04/2020	0,490083	0,396418	0,511045	0,377313
03/04/2020	0,511310	0,483715	0,530149	0,454793
06/04/2020	0,639204	0,516617	0,697313	0,491410
07/04/2020	0,612139	0,674229	0,768425	0,581095
08/04/2020	0,613731	0,598872	0,644245	0,577645
09/04/2020	0,795489	0,603648	0,800000	0,593035
10/04/2020	0,763118	0,763118	0,763118	0,763118
12/04/2020	0,758076	0,769088	0,781161	0,740564

(Sumber: Data olahan)

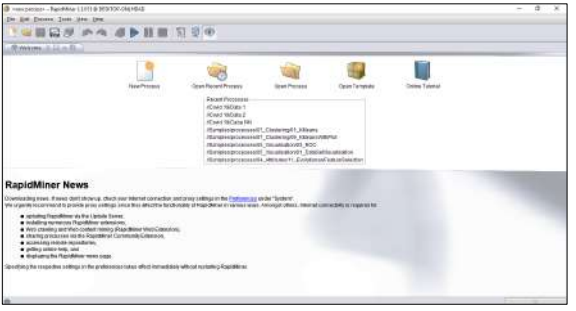
1.4.1 Langkah Langkah Analisa

Dalam hal ini langkah analisa perlu dilakukan untuk mencapai tujuan penelitian ini adalah sebagai berikut:

a) Download dataset harga emas secara *times series* dari url: https://id.investing.com/commodities/gold-historical-data periode 1 januari 2020 s/d 13 april 2020.
b) Mengubah format file dataset dari CSV menjadi XLS. Gunakan *convert* secara online (CSV to XLS).
c) Melakukan pengelolaan data dengan melakukan *preprocessing* data. Hasil *convert* harus diperhatikan untuk menyesuaikan dengan penelitian yang sedang dilakukan. Dalam hal ini tabel 1.1 adalah hasil *convert*. Perlu diperhatikan adalah kebutuhan dataset yang digunakan misalnya penggunan field tanggal (format *date*), field terakhir (format *integer* diganti ke *real* karena hasil berupa pecahan) dan lain lain.
d) Import dataset ke dalam Rapid Miner dengan menggunakan *Read Excel.*
e) Masukkan *multiply* agar dataset dapat melakukan *learning* dan *testing* sekaligus pada tiap model algoritma.
f) Masukkan model *x-validation* untuk tiap model algortima.
g) Masukkan *apply model* dan *performance*.
h) Kemudian klik *start*.
i) Menunggu proses berlangsung hingga selesai.

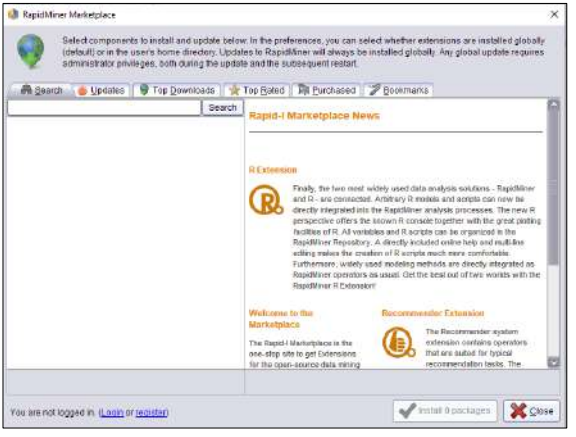
1.4.2 Penerapan Software RapidMiner 5.3.015

Lakukan penginstalan dilaptop/ PC untuk *software* RapidMiner 5.3.015. *Software* ini dapat bersifat *multiplatform* yang bisa berjalan di semua Sistem Operasi. Berikut tampilan utama *software* RapidMiner 5.3.015.



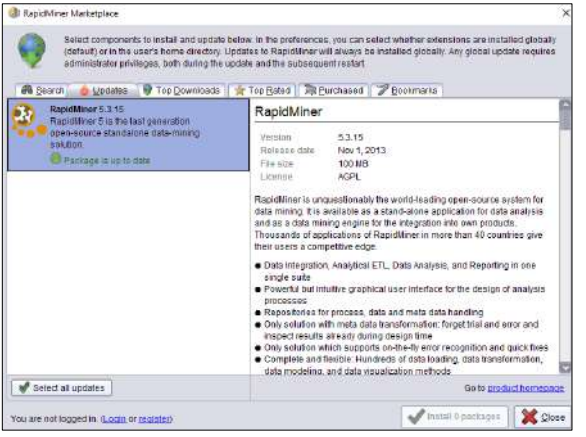
Gambar 1.4: *software* RapidMiner 5.3.015

Jika *tools Neural Network* tidak ditemukan pada *software* RapidMiner 5.3.015, maka lakukan update *software* di *marketplace* dengan cara pilih menu **Help** – **Updates dan Extensions (Marketplace)** seperti yang ditunjukkan pada tampilan berikut:



Gambar 1.5: Menu *Updates dan Extensions (a)(Marketplace)*

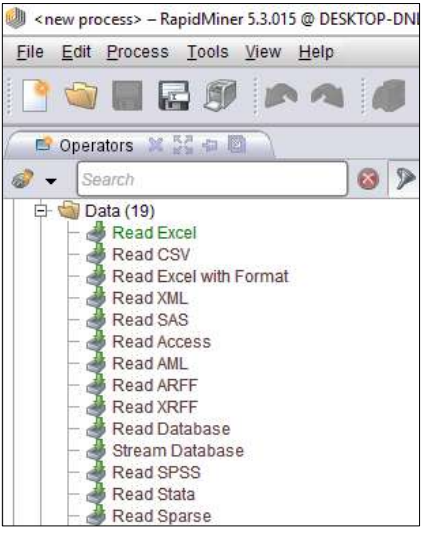
Berdasarkan gambar 1.5, klik tabs *updates* dan biarkan sistem melakukan pencarian terhadap *package* yang perlu di *updates*. Berikut tampilan *software* RapidMiner 5.3.015 yang sudah melakukan pembaharuan *package* di *marketplace*. Jika belum lakukan pembaharuan *package* dengan mengklik salah satu *package* dan *install*. Biarkan proses selesai sampai 100%. Jika berhasil maka tampilan seperti gambar berikut:



Gambar 1.6: Menu *Updates dan Extensions (b)(Marketplace)*

Pengukuran dan perbandingan prediksi dataset harga emas akan dilakukan dengan tahapan sebagai berikut:

a) Pastikan kita memilih *new process* untuk memulai *project* baru
b) *Import* dataset harga emas dengan menggunakan *read excel* seperti yang ditunjukkan pada gambar berikut:

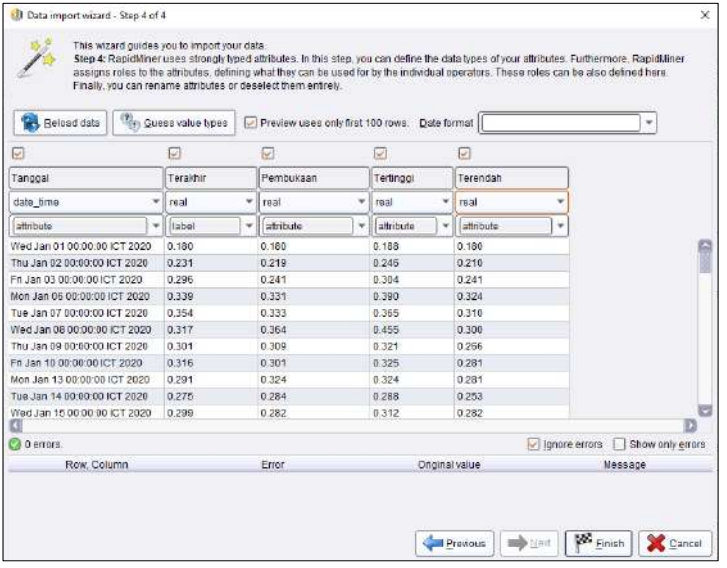


Gambar 1.7: Operator *Read Excel*

Lakukan *drag and drop* pada operator *Read Excel* ke main process seperti yang ditunjukkan pada gambar berikut:

Gambar 1.8: *drag and drop* pada operator *Read Excel*

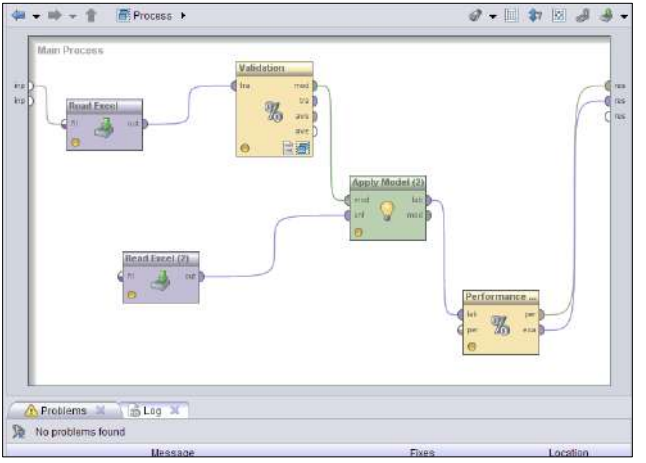
Lakukan *import* dataset pada *read excel* dengan cara klik *read excel* dan pilih *import configuration wizard*. Pilih file excel yang sudah disiapkan (tabel 1.2).



Gambar 1.9: Penentuan atribut dan label dataset harga emas

Pada gambar 1.9, menentukan *atribut* dan *label* dari dataset harga emas. Dalam hal ini yang dijadikan *output* adalah field terakhir (*label*) dengan tipe data *real*, *input* adalah field pembukaan, tertinggi, terendah dengan tipe data *real*, tanggal dengan tipe data *date_time* (*atribut*).

c) Pembentukan proses *learning* dan *testing* pada setiap model algoritma seperti yang ditunjukkan pada gambar berikut:



Gambar 1.10: Proses *learning* dan *testing* pada model algoritma Backpropogation

Pada gambar 1.10, operator *x-validation* memiliki port input yaitu, *training example set (tra)* sebagai *port input* memperkirakan *ExampleSet* untuk melatih sebuah model (*training data set*). *ExampleSet* yang sama akan digunakan selama subproses pengujian untuk menguji model.

Gambar 1.11: operator *X-Validation*

Selain itu, operator ini juga memiliki port output sebagai berikut:

***model (mod)*,** Pelatihan *subprocess* harus mengembalikan sebuah model yang dilatih pada *input ExampleSet*.

***training example set (tra)*,** *The ExampleSet* yang diberikan sebagai masukan pada *port input* pelatihan dilewatkan tanpa mengubah ke *output* melalui *port*

ini. Port ini biasa digunakan untuk menggunakan kembali *ExampleSet* sama di operator lebih lanjut atau untuk melihat *ExampleSet* dalam *Workspace Result*.

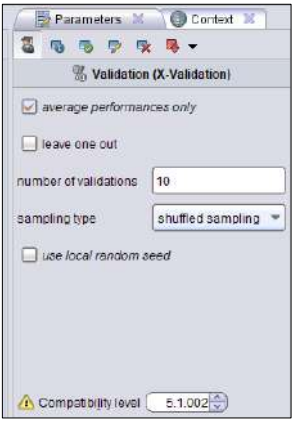
***averagable (ave)*,** subproses pengujian harus mengembalikan *Vector* Kinerja. Hal ini biasanya dihasilkan dengan menerapkan model dan mengukur kinerjanya.

Operator *X-Validation* juga memiliki parameter yang perlu diatur, di antaranya:

***linear_sampling*,** *Linear sampling* hanya membagi *ExampleSet* ke partisi tanpa mengubah urutan.

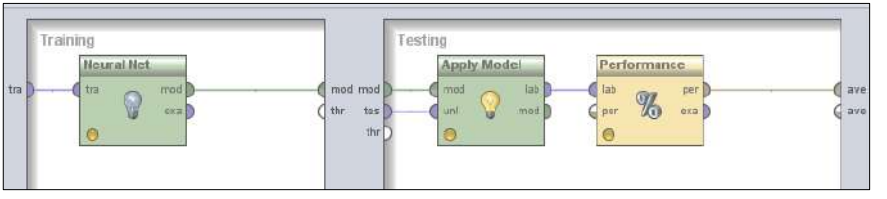
***shuffled_sampling*,** *Shuffled Sampling* membangun subset acak *ExampleSet*.

stratified_sampling, *Stratified Sampling* membangun subset acak dan memastikan bahwa distribusi kelas dalam himpunan adalah sama seperti dalam ExampleSet seluruh.



Gambar 1.12: Parameter *X-Validation*

Jika *didouble klik X-Validation* (gambar 1.12) maka akan tampilan arsitektur backpropogation seperti yang ditunjukkan pada gambar berikut:

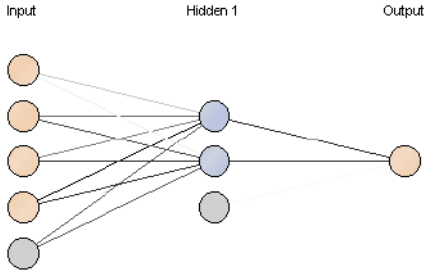


Gambar 1.13: arsitektur *backpropagation* pada operator *X-Validation*

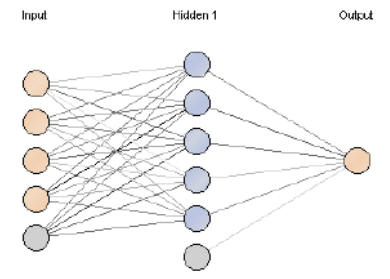
Untuk parameter yang kita gunakan pada arsitektur *backpropagation* adaalah sebagai berikut:

Input	: 4
Training clycles	: 1000
Learning rate	: 0.1
Momentum	: 0.2
Output	: 1
Error target	: 0.0001
Hidden	: 1 *layer*, 2 *layer*
Hidden 1 layer	: 2, 5, 8, 10 *neuron*
Hidden 2 layer	: 2-5, 5-8, 5-10, 8-10 *neuron*

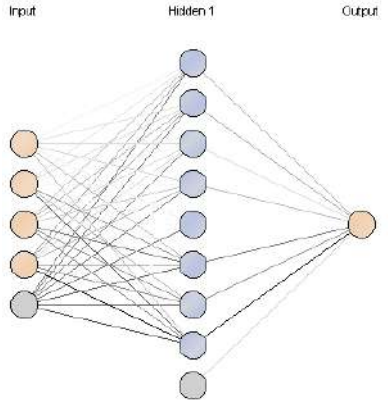
Berikut adalah arsitektur *backpropogation*:



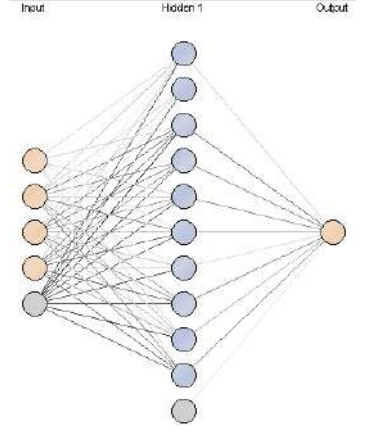
Model 4-2-1



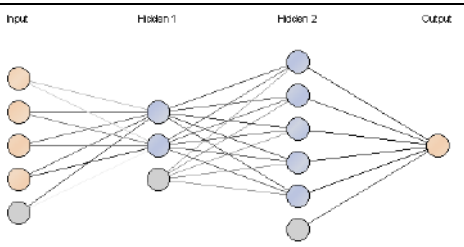
Model 4-5-1



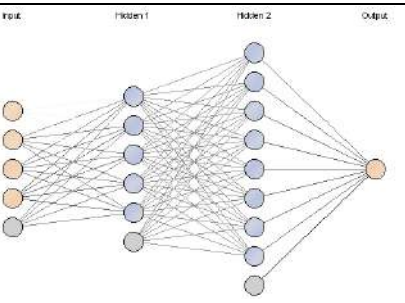
Model 4-8-1



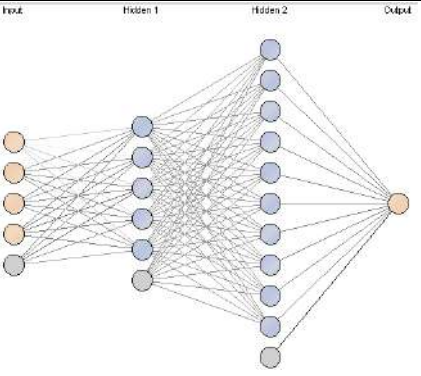
Model 4-10-1



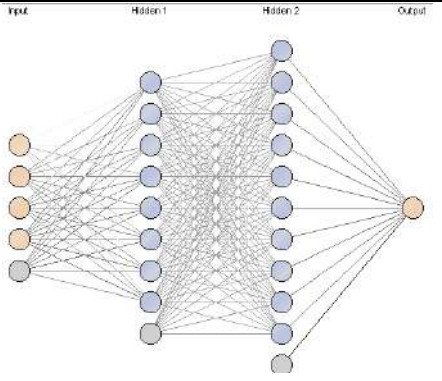
Model 4-2-5-1



Model 4-5-8-1

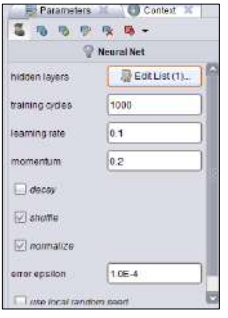


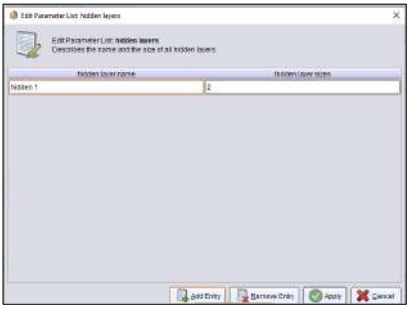
Model 4-5-10-1



Model 4-8-10-1

Model arsitektur tersebut akan di *training* untuk memilih model arsitektur yang paling baik yang dilihat dari *Root Mean Square Error* (RMSE). Untuk membuat model *backpropagation* di *RapidMiner* dengan meng *klik neural network* (gambar 1.14) dan pada parameter pilih *hidden layers* seperti yang ditunjukkan pada gambar berikut:





Gambar 1.14: parameter pada *hidden layers*

Untuk menambahkan *hidden* cukup dengan meng klik *add entry* dan tentukan *neuron* yang diinginkan. Untuk contoh gambar tersebut adalah 1 *hidden* dengan 2 *neuron.*

Berikut adalah hasil *Root Mean Square Error* (RMSE) dari 8 model arsitektur *backpropagation* yang diuji.

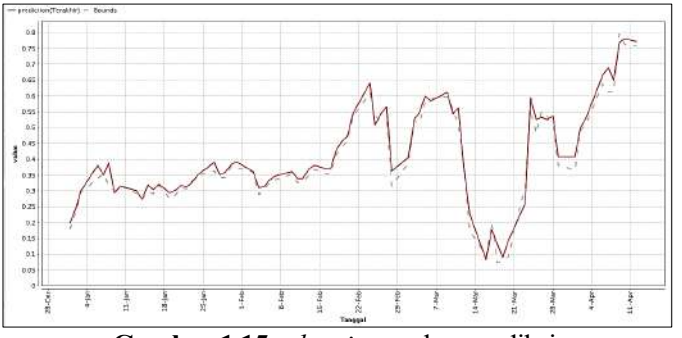
Tabel 1.3: Hasil training *Root Mean Square Error* (RMSE) dengan Algoritma *backpropagation*

No	Model Arsitektur	Training Cycles	Lr	Momentum	RMSE
1	4-2-1	1000	0.1	0.2	0.023 +/- 0.000
2	4-5-1	1000	0.1	0.2	0.022 +/- 0.000
3	4-8-1	1000	0.1	0.2	0.027 +/- 0.000
4	4-10-1	1000	0.1	0.2	0.024 +/- 0.000
5	4-2-5-1	1000	0.1	0.2	0.023 +/- 0.000
6	4-5-8-1	1000	0.1	0.2	0.024 +/- 0.000
7	4-5-10-1	1000	0.1	0.2	0.025 +/- 0.000
8	4-8-10-1	1000	0.1	0.2	0.025 +/- 0.000

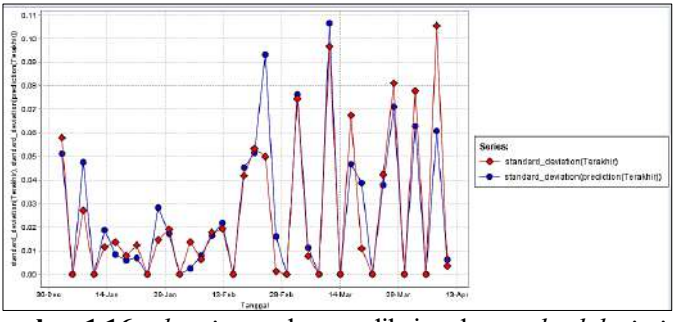
(Sumber: Data olahan)

Berdasarkan serangkaian uji coba model arsitektur, maka pemilihan dilakukan dengan melihat *value* dari *Root Mean Square Error* (RMSE). Semakin kecil nilai RMSE maka semakin baik model arsitektur tersebut. Dari hasil tabel 1.3 dapat diperoleh bahwa model terbaik dari 8 model yang di uji coba adalah **model 4-8-1** dengan RMSE: 0.027 +/- 0.000.

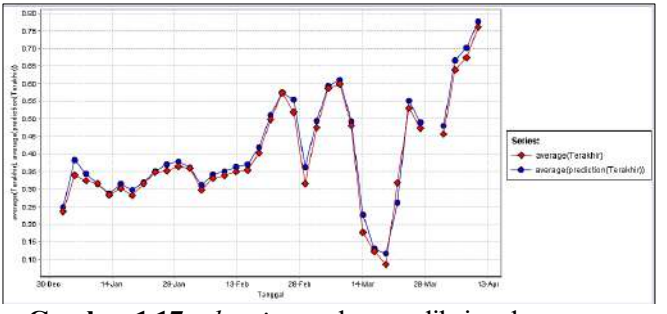
Berikut hasil beberapa *plot view* dari model terbaik (4-8-1) seperti yang ditunjukkan pada gambar berikut:



Gambar 1.15: *plot view* real vs prediksi



Gambar 1.16: *plot view* real vs prediksi pada *standard deviation*



Gambar 1.17: *plot view* real vs prediksi pada *average*

Berikut hasil prediksi dengan model arsitektur *backpropagation* 4-8-1 seperti yang ditunjukkan pada tabel berikut:

Tabel 1.4: Hasil prediksi *backpropagation* model arsitektur

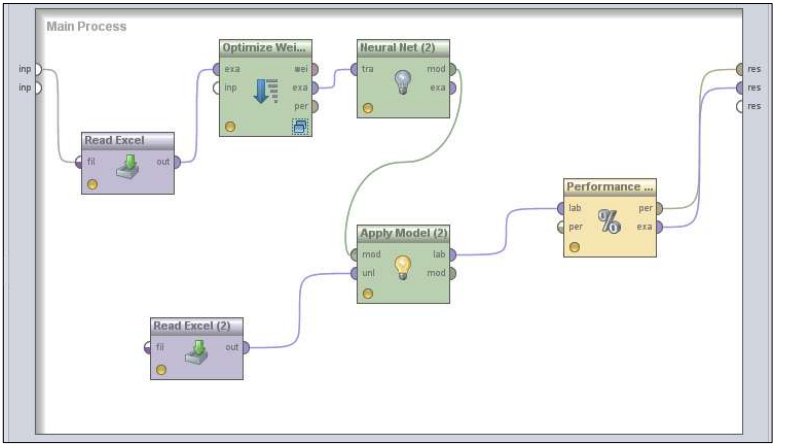
Row No.	Terakhir	prediction(Terakhir)
1	1520.950	1528.125
2	1540.100	1544.800
3	1564.500	1566.607
4	1580.900	1596.929
5	1586.300	1584.769
6	1572.300	1599.237
7	1566.400	1563.625
8	1572.100	1571.637
9	1562.700	1566.052
10	1556.500	1556.063
11	1565.600	1573.194
12	1562.200	1567.903
13	1572.300	1573.585
14	1557.050	1563.577
15	1561.250	1566.742
16	1569.900	1572.854
17	1568.700	1571.023
18	1577.300	1576.203
19	1583.900	1585.258
20	1589.500	1600.049
21	1581.700	1584.988
22	1582	1588.278
23	1595.100	1597.349
24	1593.800	1600.946
25	1588.300	1588.742

Row No.	Terakhir	prediction(Terakhir)
26	1561.400	1569.505
27	1568.700	1570.837
28	1575.800	1579.406
29	1579.200	1583.659
30	1585.400	1589.382
31	1575.900	1580.641
32	1577.300	1580.643
33	1584.500	1591.904
34	1591.900	1596.630
35	1586.250	1592.375
36	1587.050	1592.244
37	1609.300	1614.365
38	1617.500	1625.771
39	1626.200	1631.346
40	1654.600	1658.790
41	1682.400	1694.467
42	1655.800	1644.830
43	1648.900	1657.647
44	1648.200	1686.184
45	1571.800	1589.431
46	1599.500	1606.266
47	1648.900	1651.080
48	1647.200	1659.861
49	1672	1679.329
50	1676.100	1673.313

51	1676.600	1683.033
52	1663.300	1658.123
53	1645.400	1664.857
54	1593.200	1592.252
55	1519.500	1538.200
56	1488.600	1484.121
57	1528.100	1519.386
58	1480.600	1502.607
59	1482.300	1486.566
60	1488.100	1507.232
61	1572.700	1551.361
62	1663.300	1676.916
63	1634.300	1650.847
64	1660.300	1653.868
65	1654.100	1650.908
66	1643.200	1654.950
67	1596.600	1606.700
68	1591.400	1607.042
69	1637.700	1642.301
70	1645.700	1652.099
71	1693.900	1703.879
72	1683.700	1713.125
73	1684.300	1697.186
74	1752.800	1742.318
75	1740.600	1747.218

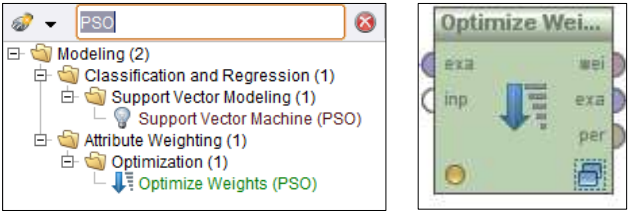
1.5 Implementasi *Backpropagation* + PSO

Pada sub ini model arsitektur yang sudah ditetapkan pada pembahasan sub **1.4.2** di mana model 4-8-1 adalah model arsitektur *backpropagation* terbaik. Model ini akan dioptimasi dengan menggunakan *Particle Swarm Optimization* (PSO). Berikut model pembentukan proses *learning* dan *testing* pada model algoritma *backpropagation* + PSO seperti yang ditunjukkan pada gambar berikut:



Gambar 1.18: Proses *learning* dan *testing* pada model algoritma *Backpropogation* + PSO

Untuk parameter PSO pada *software* RapidMiner 5.3.015 adalah sebagai berikut:



Gambar 1.19: parameter *Particle Swarm Optimization*

Pada gambar 1.19 dapat dijelaskan bahwa untuk *read excel* (1): data *training* dan *read excel* (2): data *testing* yang diisi dengan data sama (data yang

digunakan ketika mencari model arsitektur *backpropagation*). Untuk *neural network* kita atur ke model 4-8-1 (proses konfigurasi sama: lihat gambar 1.14).

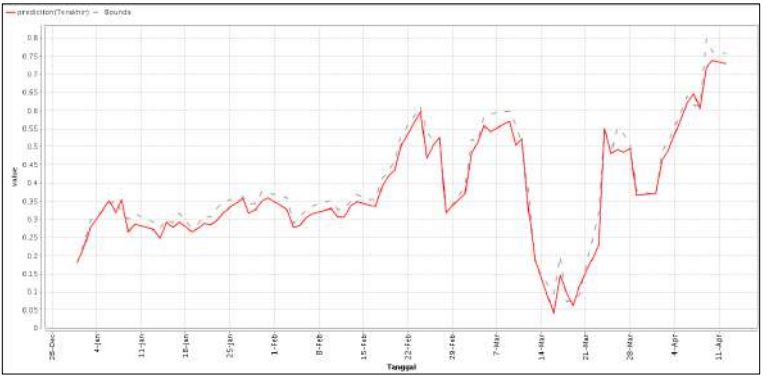
Berikut adalah hasil *Root Mean Square Error* (RMSE) dengan optimasi *Particle Swarm Optimization* (PSO).

Tabel 1.5: Perbandingan Hasil *training* RMSE *backpropagation* dan *backpropagation* + PSO

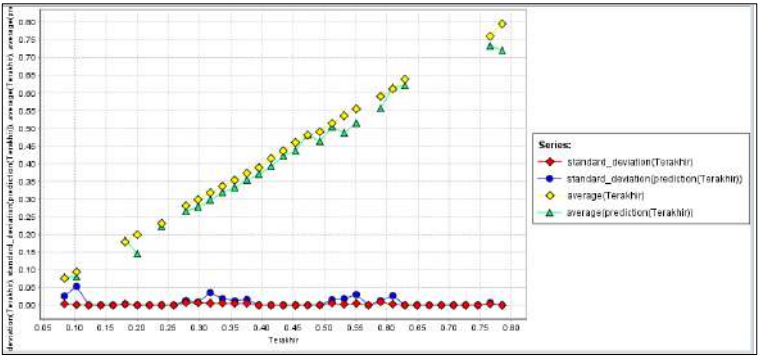
No	Model Arsitektur	Training Cycles	Model	RMSE
1	4-8-1	1000	Backpropagation	0.027 +/- 0.000
2	4-8-1	1000	Backpropagation + PSO	0.030 +/- 0.000

Berdasarkan hasil percobaan yang dilakukan, dapat dijelaskan bahwa hasil dari tabel 1.5 menyatakan bahwa optimasi *backpropagation* + PSO jauh lebih baik dibandingkan dengan *backpropagation*. Dapat dijelaskan bahwa *Root Mean Square Error* (RMSE) menjadi lebih baik setelah dioptimasi dari 0.027 +/- 0.000 menjadi 0.030 +/- 0.000 (lebih baik 0.003). Berikut hasil beberapa *plot view* dari model terbaik (4-8-1) hasil optimasi *backpropagation* + PSO seperti yang ditunjukkan pada gambar berikut:

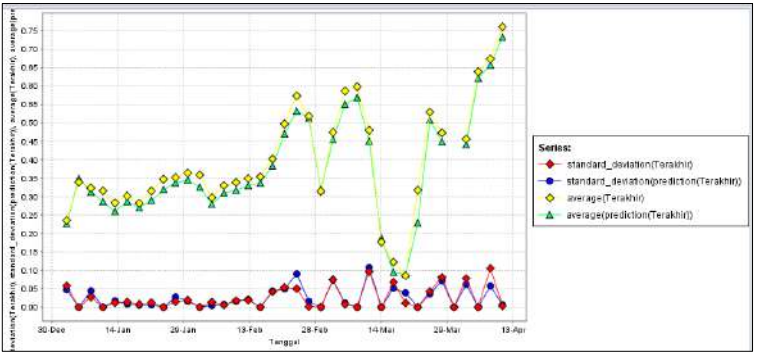
Gambar 1.20: *plot view* real vs prediksi berdasarkan real



Gambar 1.21: *plot view* real vs prediksi berdasarkan tanggal



Gambar 1.22: *plot view* real vs prediksi berdasarkan real yang dilihat dari *average* dan standard *deviation*



Gambar 1.23: *plot view* real vs prediksi berdasarkan tanggal yang dilihat dari *average* dan standard *deviation*

Berikut hasil prediksi dengan model arsitektur 4-8-1 hasil optimasi *backpropagation* + PSO seperti yang ditunjukkan pada tabel berikut:

Tabel 1.6: Hasil prediksi model arsitektur 4-8-1 hasil optimasi *backpropagation* + PSO

Row No.	Terakhir	prediction(Terakhir)
1	1520.950	1521.682
2	1540.100	1536.804
3	1564.500	1556.979
4	1580.900	1584.655
5	1586.300	1573.215
6	1572.300	1586.025
7	1566.400	1553.198
8	1572.100	1561.002
9	1562.700	1555.291
10	1556.500	1546.140
11	1565.600	1562.609
12	1562.200	1557.295
13	1572.300	1562.770
14	1557.050	1553.456
15	1561.250	1556.408
16	1569.900	1561.576
17	1568.700	1560.113
18	1577.300	1564.841
19	1583.900	1573.439
20	1589.500	1587.351
21	1581.700	1572.685
22	1582	1576.197
23	1595.100	1584.792
24	1593.800	1588.168
25	1588.300	1575.781

Row No.	Terakhir	prediction(Terakhir)
26	1581.400	1557.156
27	1568.700	1559.504
28	1575.800	1567.768
29	1579.200	1571.647
30	1585.400	1577.120
31	1575.900	1568.483
32	1577.300	1568.660
33	1584.500	1579.602
34	1591.900	1583.984
35	1586.250	1579.974
36	1587.050	1579.741
37	1609.300	1600.833
38	1617.500	1611.906
39	1626.200	1617.121
40	1654.600	1643.455
41	1682.400	1677.764
42	1655.800	1629.645
43	1648.900	1642.436
44	1648.200	1650.812
45	1571.800	1573.295
46	1599.500	1592.524
47	1649.900	1635.751
48	1647.200	1644.863
49	1672	1663.425
50	1676.100	1657.531

Row No.	Terakhir	prediction(Terakhir)
51	1678.600	1667.234
52	1663.300	1642.817
53	1645.400	1649.499
54	1593.200	1576.231
55	1519.500	1523.435
56	1488.600	1468.740
57	1528.100	1507.569
58	1480.600	1489.696
59	1482.300	1476.314
60	1488.100	1497.057
61	1572.700	1539.040
62	1663.300	1659.795
63	1634.300	1634.695
64	1660.300	1638.323
65	1654.100	1636.019
66	1643.200	1639.741
67	1596.600	1591.279
68	1591.400	1593.153
69	1637.700	1627.523
70	1645.700	1637.343
71	1693.900	1687.002
72	1683.700	1696.581
73	1684.300	1681.582
74	1752.600	1724.216
75	1740.600	1730.879

Bab 2

Algoritma Perseptron dan Penerapan Aplikasi

2.1 Pendahuluan

Artificial Neural Network (selanjutnya disingkat ANN), menghasilkan model yang sulit dibaca dan dimengerti oleh manusia karena memiliki banyak layer (kecuali model jaringan Perceptron) dan sifat non-linear (merujuk pada fungsi aktivasi) (Poustie *et al.*, 1999). Model jaringan Perceptron merupakan pengembangan dari model jaringan Hebb. Persamaan itu terletak pada proses perubahan bobot pembelajarannya ditambahkan laju pembelajaran (*learning rate*) α yang berfungsi untuk mengontrol perubahan bobot pada setiap iterasi (Sugiarti *et al.*, 2019). Model jaringan perceptron ditemukan Rosenblatt (1962) dan Minsky-Papert (1969) yang merupakan model yang memiliki aplikasi dan pelatihan yang paling baik pada era tersebut (Lukito, 2017). Model jaringan Perceptron terdiri dari sebuah bias (yang ada secara *default*) dan beberapa masukan yang menghasilkan sebuah keluaran (Sebatubun and Nugroho, 2017). Berikut adalah sejarah perkembangan *Artificial Neural Network*:

Tahun 1940-an, para ilmuwan menemukan bahwa psikologi otak sama dengan mode pemrosesan yang dilakukan oleh komputer:

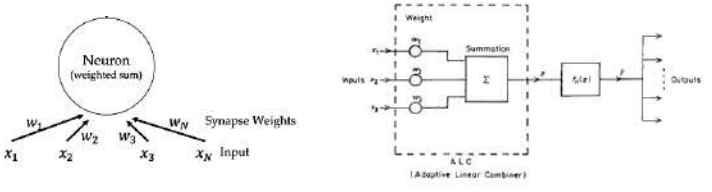
a) **Tahun 1943**, McCulloch dan Pitts merancang model formal yang pertamakali sebagai perhitungan dasar *neuron*.
b) **Tahun 1954**, Farley dan Clark mensetup model-model untuk relasiadaptif stimulus-respon dalam jaringan random.
c) **Tahun 1958**, Rosen blatt mengembangkan konsep dasar tentang perception untuk klasifikasi pola.

d) **Tahun 1960**, Widrow dan Hoff mengembangkan ADALINE yang dilatih dengan pembelajaran *Least Mean Square (LMS).*
e) **Tahun 1974**, Werbos memperkenalkan algoritma *backpropagation*
f) **Tahun 1975**, Little dan Shaw menggambarkan jaringan saraf dengan probabilistik.
g) **Tahun 1982**, Kohonen mengembangkan metode pembelajaran yang tidak terawasi untuk pemetaan pada jaringan saraf tiruan.
h) **Tahun 1982**, Grossberg mengembangkan teori jaringan.
i) **Tahun 1982**, Hopfield mengembangkan jaringan saraf *reccurent.*
j) **Tahun 1985**, algoritma pembelajaran dengan mensin Boltzmann.
k) **Tahun 1987**, Kosko mengembangkan model jaringan *Adaptive Bidirectional Associative Memory (BAM).*
l) **Tahun 1988**, dikembangkan fungsi radial bebas (Musthofa, Umma and Handayani, 2017).

2.1.1 Arsitektur Jaringan

Bentuk terkecil (minimal) sebuah ANN adalah model arsitektur Perceptron yang hanya terdiri dari sebuah *neuron* (Sumijan *et al.*, 2016). Model arsitektur Perceptron terdiri dari 1 lapisan *input* dan 1 lapisan *output*. Setiap unit pada lapisan *input* dihubungkan dengan semua unit di lapisan *output* dengan suatu bobot keterhubungan (Rouza, Jufri and Fimawahib, 2021). Fungsi aktivasi bukan merupakan fungsi biner atau bipolar, tetapi memiliki kemungkinan nilai -1, 0 atau 1 (Yanto, Sovia and Wiyata, 2018). Model arsitektur Perceptron juga memiliki Unit bias. Unit bisa merupakan satu unit masukan di suatu lapisan yang nilai *input*nya selalu 1 (Poustie *et al.*, 1999).

Model arsitektur ini dipakai untuk mengklasifikasi satu kelas tunggal. Yaitu menentukan apakah suatu pola *input* termasuk dalam satu kelas atau tidak. Jika masuk dalam satu kelas maka respon dari unit *output* adalah 1 sedangkan jika tidak maka responnya –1 (Yanto, 2017).



Gambar 2.1: Model arsitektur Perseptron

Pada gambar 2.1, jumlah *output* dari sejumlah *i* perceptron adalah z_i dan *input*nya, X_{i1} ... X_{in} hubungan penjumlahan perceptron sebagai berikut (Musthofa, Umma and Handayani, 2017):

$$Z_i = \sum_{j=1}^{m} W_{ij} X_{ij} \quad (2.1)$$

Di mana W_{ij} merupakan bobot dari *input* ke *j* pada sel ke *i*. Persamaan 2.1 dapat ditulis dalam format vector berikut:

$$Z_i = W_i^T X_I \quad (2.2)$$

Di mana:

$$W_i = [W_{i1} \ldots W_{in}]^T \quad (2.3)$$

$$X_i = [X_{i1} \ldots X_{in}]^T \quad (2.4)$$

2.2.2 Fungsi Aktivasi

Proses pada model jaringan Perseptron, dimulai dari *input* yang diterima oleh *neuron* beserta dengan nilai bobot dari tiap-tiap *input* yang ada. Setelah masuk ke dalam *neuron*, nilai *input* akan dijumlahkan oleh suatu fungsi perambatan (*summing function* (∑)). Penjumlahan akan diproses oleh **fungsi aktivasi** setiap *neuron*, disini akan dibandingkan hasil penjumlahan dengan *threshold* (nilai ambang) tertentu. Jika nilai melebihi *threshold*, maka aktivasi *neuron* akan dibatalkan, sebaliknya, jika masih di bawah nilai *threshold*, *neuron* akan diaktifkan. Setelah aktif, *neuron* akan mengirimkan nilai *output* melalui bobot-bobot *output*nya ke semua *neuron* yang berhubungan dengannya. Proses ini akan terus berulang pada *input-input* selanjutnya (Lukito, 2017). Dengan kata lain Fungsi aktivasi adalah fungsi step biner dgn nilai threshold θ tertentu. Sinyal yg dikirim ke unit *output* adalah sinyal biner (1 atau 0).

Output Perceptron $y = f(y\_in)$ (2.5)

Di mana fungsi aktivasinya adalah :

$$y = \begin{cases} 1 & jika\ y\_in > +\theta \\ 0 & jika\ -\theta \le y\_in \le +\theta \\ -1 & jika\ y\_in < -\theta \end{cases} \quad (2.6)$$

2.2.3 Proses Pembelajaran Perceptron

Berikut langkah pembelajaran model jaringan Perceptron di mana proses terdiri dari pembelajaran dan pengenalan seperti berikut (Lukito, 2017):

Algoritma Pembelajaran

Langkah 0 : Inisialisasi nilai bobot (w_i dan b), laju pembelajaran (α), dan nilai ambang (θ)
$w_i = 0$ (i = 1..n) ; b = 0

Langkah 1 : Selama syarat henti iterasi tidak dipenuhi kerjakan langkah 2-6

Langkah 2 : Untuk setiap pasangan pelatihan (s,t) kerjakan langkah 3-5

Langkah 3 : Tentukan nilai aktivasi untuk unit *input*.
$X_i = S_i$

Langkah 4 : Hitung respon untuk unit *output*

$$y\_in = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 & jika\ y\_in > +\theta \\ 0 & jika\ -\theta \le y\_in \le +\theta \\ -1 & jika\ y\_in < -\theta \end{cases} \quad (2.7)$$

Langkah 5 : Jika masih ada *error*, *update* nilai bobot dan bias

if y ≠ t

$w_i(baru) = w_i(lama) + \alpha.t.x$

$b(baru) = b(lama) + \alpha.t$

else

$w_i(baru) = w_i(lama)$

$b(baru) = b(lama)$

Langkah 6 : Uji kondisi henti, jika tidak ada perubahan bobot pada langkah 2 (untuk 1 *epoch*) berhenti, jika tidak lanjutkan.

Algoritma Pengenalan

Langkah 0 : Inisialisasi w_i , b, α, θ dari proses pembelajaran sebelumnya.

Langkah 1 : Tentukan nilai aktivasi unit *input*.
$X_i = S_i$

Langkah 2 : Hitung respon untuk unit *output*

$$y\_in = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 & jika\ y\_in > +\theta \\ 0 & jika\ -\theta \le y\_in \le +\theta \\ -1 & jika\ y\_in < -\theta \end{cases} \quad (2.8)$$

2.2.4 Contoh Perhitungan Perceptron

Berikut adalah penyelesaian model jaringan Perceptron untuk fungsi logika AND dengan *input* biner dan target bipolar seperti yang ditunjukkan pada tabel berikut:

Tabel 2.1: Fungsi Logika AND

Masukan			**Target**
X1	X2	b	t
1	1	1	1
1	0	0	-1
0	1	0	-1
0	0	0	-1

Di mana:

Nilai $\alpha = 0.7$

Bobot (w) dan bias (b) diberi nilai 0

Nilai $\theta = 0.3$

a) Iterasi pertama:

Data pertama: (1 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = 0 + (1).(0) + (1).(0) = 0$

hasil aktivasi = y = 0 (-0.3 <y_in<0.3) target t = 1

$y \neq t$

bobot baru :

$w_1(baru) = w_1(lama) + \alpha t x_1$

$w_1 = 0+(0.7)(1)(1) = 0.7$

$w_2 = 0+(0.7)(1)(1) = 0.7$

bias baru :

$b(baru) = b(lama) + \alpha t$

$b = 0 + (0.7)(1) = 0.7$

Data kedua: (1 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = 0.7 + (1)(0.7) + (0)(0.7) = 1.4$

hasil aktivasi = y = 1 (y_in > 0.3) target t = -1

$y \neq t$

bobot baru :

$w_1(baru) = w_1(lama) + \alpha t x_1$

$w_1 = 0.7+(0.7)(-1)(1) = 0$

$w_2 = 0.7+(0.7)(-1)(0) = 0.7$

bias baru :

$b(baru) = b(lama) + \alpha t$

$b = 0.7 + (0.7)(-1) = 0$

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = 0 + (0)(0) + (1)(0.7) = 0.7$

hasil aktivasi = y = 1 (y_in > 0.3) target t = -1

$y \neq t$

bobot baru:

$w_1(baru) = w_1(lama) + \alpha t x_1$

$w_1 = 0+(0.7)(-1)(0) = 0$

$w_2 = 0.7+(0.7)(-1)(1) = 0$

bias baru:

$b(baru) = b(lama) + \alpha t$

$b = 0 + (0.7)(-1) = -0.7$

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -0.7 + (0)(0) + (0)(0) = -0.7$

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

$y = t$

b) Iterasi kedua:

Data pertama: (1 1)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = -0.7 + (0)(0) + (0)(0) = -0.7

hasil aktivasi = y = -1 (y_in <- 0.3) target t = 1

y ≠ t

bobot baru:

w_i(baru) = w_i(lama) + $\alpha t x_i$

w_1 = 0+(0.7)(1)(1) = 0.7

w_2 = 0+(0.7)(1)(1) = 0.7

bias baru:

b(baru) = b(lama) + αt

b = -0.7 + (0.7)(1) = 0

Data kedua: (1 0)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = 0 + (0.7)(1) + (0.7)(0) = 0.7

hasil aktivasi = y = 1 (y_in > 0.3) target t = -1

y ≠ t

bobot baru:

w_i(baru) = w_i(lama) + $\alpha t x_i$

w_1 = 0.7+(0.7)(-1)(1) = 0

w_2 = 0.7+(0.7)(-1)(0) = 0.7

bias baru:

b(baru) = b(lama) + αt

b = 0 + (0.7)(-1) = -0.7

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = -0.7 + (0)(0) + (0)(0.7) = -0.7

hasil aktivasi = y = 0 (-0.3<y_in < 0.3) target t = -1

$y \neq t$
bobot baru:
$w_i(baru) = w_i(lama) + \alpha t x_i$
$w_1 = 0+(0.7)(-1)(0) = 0$
$w_2 = 0.7+(0.7)(-1)(1) = 0$
bias baru:
$b(baru) = b(lama) + \alpha t$
$b = -0.7 + (0.7)(-1) = -1.4$

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -1.4 + (0)(0) + (0)(0) = -1.4$
hasil aktivasi = y = -1 (y_in < -0.3) target t = -1
y = t

c) Iterasi ketiga:

Data pertama: (1 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -1.4 +(1)(0) + (1)(0) = -1.4$
hasil aktivasi = y = -1 (y_in <- 0.3) target t = 1
$y \neq t$
bobot baru:
$w_i(baru) = w_i(lama) + \alpha t x_i$
$w_1 = 0+(0.7)(1)(1) = 0.7$
$w_2 = 0+(0.7)(1)(1) = 0.7$
bias baru:
$b(baru) = b(lama) + \alpha t$
$b = -1.4 + (0.7)(1) = -0.7$

Data kedua: (1 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

y_in = -0.7 + (1)(0.7) + (0)(0.7) = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3) target t = -1
y ≠ t
bobot baru:
w_i(baru) = w_i(lama) + $\alpha t x_i$
w_1 = 0.7+(0.7)(-1)(1) = 0
w_2 = 0.7+(0.7)(-1)(0) = 0.7
bias baru:
b(baru) = b(lama) + αt
b = -0.7 + (0.7)(-1) = -1.4

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i$ = b + $x_1 w_1$ + $x_2 w_2$

y_in = -1.4 + (0)(0) + (1)(0.7) = -0.7
hasil aktivasi = y = -1 (y_in < -0.3) target t = -1
y = t

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i$ = b + $x_1 w_1$ + $x_2 w_2$

y_in = -1.4 + (0)(0) + (0)(0.7) = -1.4
hasil aktivasi = y = -1 (y_in < -0.3) target t = -1
y = t

d) Iterasi keempat:

Data pertama: (1 1)

$y\_in = b + \sum_i x_i w_i$ = b + $x_1 w_1$ + $x_2 w_2$

y_in = -1.4 + (1)(0) + (1)(0.7) = -0.7
hasil aktivasi = y = -1 (y_in <- 0.3) target t = 1
y ≠ t
bobot baru:
w_i(baru) = w_i(lama) + $\alpha t x_i$

$w_1 = 0+(0.7)(1)(1) = 0.7$
$w_2 = 0.7+(0.7)(1)(1) = 1.4$
bias baru:
b(baru) = b(lama) + αt
b = -1.4 + (0.7)(1) = -0.7

Data kedua: (1 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -0.7 + (1)(0.7) + (0)(1.4) = 0$

hasil aktivasi = y = 0 (-0.3<y_in < 0.3) target t = -1
$y \neq t$
bobot baru:
w_i(baru) = w_i(lama) + αtx_i
$w_1 = 0.7+(0.7)(-1)(1) = 0$
$w_2 = 1.4+(0.7)(-1)(0) = 1.4$
bias baru:
b(baru) = b(lama) + αt
b = -0.7 + (0.7)(-1) = -1.4

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -1.4 + (0)(0) + (1)(1.4) = 0$

hasil aktivasi = y = 0 (-0.3<y_in < 0.3) target t = -1
$y \neq t$
bobot baru:
w_i(baru) = w_i(lama) + αtx_i
$w_1 = 0+(0.7)(-1)(0) = 0$
$w_2 = 1.4+(0.7)(-1)(1) = 0.7$
bias baru:
b(baru) = b(lama) + αt
b = -1.4 + (0.7)(-1) = -2.1

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = -2.1 + (0)(0) + (0)(0.7) = -2.1

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

y = t

e) Iterasi kelima:

Data pertama : (1 1)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = -2.1 + (1)(0) + (1)(0.7) = -1.4

hasil aktivasi = y = -1 (y_in <- 0.3) target t = 1

$y \neq t$

bobot baru:

w_i(baru) = w_i(lama) + $\alpha t x_i$

w_1 = 0+(0.7)(1)(1) = 0.7

w_2 = 0.7+(0.7)(1)(1) = 1.4

bias baru:

b(baru) = b(lama) + αt

b = -2.1 + (0.7)(1) = -1.4

Data kedua: (1 0)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = -1.4 + (1)(0.7) + (0)(1.4) = -0.7

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

y = t

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = -1.4 + (0)(0.7) + (1)(1.4) = 0

hasil aktivasi = y = 0 (-0.3<y_in < 0.3) target t = -1

$y \neq t$
bobot baru:
$w_i(baru) = w_i(lama) + \alpha t x_i$
$w_1 = 0.7+(0.7)(-1)(0) = 0.7$
$w_2 = 1.4+(0.7)(-1)(1) = 0.7$
bias baru:
$b(baru) = b(lama) + \alpha t$
$b = -1.4 + (0.7)(-1) = -2.1$

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.1 + (0)(0.7) + (0)(0.7) = -2.1$

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1
y = t

f) Iterasi keenam:

Data pertama: (1 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.1 + (1)(0.7) + (1)(0.7) = -0.7$

hasil aktivasi = y = -1 (y_in <- 0.3) target t = 1
$y \neq t$
bobot baru:
$w_i(baru) = w_i(lama) + \alpha t x_i$
$w_1 = 0.7+(0.7)(1)(1) = 1.4$
$w_2 = 0.7+(0.7)(1)(1) = 1.4$
bias baru:
$b(baru) = b(lama) + \alpha t$
$b = -2.1 + (0.7)(1) = -1.4$

Data kedua: (1 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -1.4 + (1)(1.4) + (0)(1.4) = 0$

hasil aktivasi = y = 0 (-0.3<y_in < 0.3) target t = -1

$y \neq t$

bobot baru:

$w_i(baru) = w_i(lama) + \alpha t x_i$

$w_1 = 0.7+(0.7)(-1)(0) = 0.7$

$w_2 = 1.4+(0.7)(-1)(0) = 1.4$

bias baru:

$b(baru) = b(lama) + \alpha t$

$b = -1.4 + (0.7)(-1) = -2.1$

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.1 + (0)(0.7) + (1)(1.4) = -0.7$

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

y = t

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.1 + (0)(0.7) + (0)(1.4) = -2.1$

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

y = t

g) Iterasi ketujuh:

Data pertama: (1 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.1 + (1)(0.7) + (1)(1.4) = 0$

hasil aktivasi = y = 0 (-0.3<y_in< 0.3) target t = 1

$y \neq t$
bobot baru:
$w_i(baru) = w_i(lama) + \alpha t x_i$
$w_1 = 0.7+(0.7)(1)(1) = 1.4$
$w_2 = 1.4+(0.7)(1)(1) = 2.1$
bias baru:
$b(baru) = b(lama) + \alpha t$
$b = -2.1 + (0.7)(1) = -1.4$

Data kedua: (1 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -1.4 + (1)(1.4) + (0)(2.1) = 0$
hasil aktivasi = y = 0 ($-0.3 < y\_in < 0.3$) target t = -1
$y \neq t$
bobot baru:
$w_i(baru) = w_i(lama) + \alpha t x_i$
$w_1 = 1.4+(0.7)(-1)(0) = 0.7$
$w_2 = 2.1+(0.7)(-1)(0) = 2.1$
bias baru:
$b(baru) = b(lama) + \alpha t$
$b = -1.4 + (0.7)(-1) = -2.1$

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.1 + (0)(0.7) + (1)(2.1) = 0$
hasil aktivasi = y = 0 ($-0.3 < y\_in < 0.3$) target t = -1
$y \neq t$
bobot baru:
$w_i(baru) = w_i(lama) + \alpha t x_i$
$w_1 = 0.7+(0.7)(-1)(0) = 0.7$
$w_2 = 2.1+(0.7)(-1)(1) = -1.4$
bias baru:

b(baru) = b(lama) + αt
b = -2.1 + (0.7)(-1) = -2.8

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = -2.8 + (0)(0.7) + (0)(-1.4) = -2.8
hasil aktivasi = y = -1 (y_in < -0.3) target t = -1
y = t

h) Iterasi kedelapan:

Data pertama: (1 1)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = -2.8 + (1)(0.7) + (1)(1.4) = -0.7
hasil aktivasi = y = -1 (y_in <- 0.3) target t = 1
y ≠ t
bobot baru:
w_i(baru) = w_i(lama) + αtx_i
w_1 = 0.7+(0.7)(1)(1) = 1.4
w_2 = 1.4+(0.7)(1)(1) = 2.1
bias baru:
b(baru) = b(lama) + αt
b = -2.8 + (0.7)(1) = -2.1

Data kedua : (1 0)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in = -2.1 + (1)(1.4) + (0)(2.1) = -0.7
hasil aktivasi = y = -1 (y_in < -0.3) target t = -1
y = t

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.1 + (0)(1.4) + (0)(2.1) = 0$

hasil aktivasi = y = 0 (-0.3<y_in < 0.3) target t = -1

$y \neq t$

bobot baru:

$w_i(baru) = w_i(lama) + \alpha t x_i$

$w_1 = 1.4+(0.7)(-1)(0) = 1.4$

$w_2 = 2.1+(0.7)(-1)(1) = 1.4$

bias baru:

$b(baru) = b(lama) + \alpha t$

$b = -2.1 + (0.7)(-1) = -2.8$

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.8 + (0)(1.4) + (0)(1.4) = -2.8$

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

y = t

i) Iterasi Kesembilan:

Data pertama: (1 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.8 + (1)(1.4) + (1)(1.4) = 0$

hasil aktivasi = y = 0 (-0.3<y_in< 0.3) target t = 1

$y \neq t$

bobot baru:

$w_i(baru) = w_i(lama) + \alpha t x_i$

$w_1 = 1.4+(0.7)(1)(1) = 2.1$

$w_2 = 1.4+(0.7)(1)(1) = 2.1$

bias baru:

$b(baru) = b(lama) + \alpha t$

b = -2.8 + (0.7)(1) = -2.1

Data kedua: (1 0)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in =-2.1 + (1)(2.1) + (0)(2.1) = 0

hasil aktivasi = y = 0 (-0.3<y_in < 0.3) target t = -1

y ≠ t

bobot baru:

w_i(baru) = w_i(lama) + $\alpha t x_i$

w_1 = 2.1+(0.7)(-1)(1) = 1.4

w_2 = 2.1+(0.7)(-1)(0) = 2.1

bias baru:

b(baru) = b(lama) + αt

b = -2.1 + (0.7)(-1) = -2.8

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in =-2.8 + (0)(1.4) + (1)(2.1) = -0.7

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

y = t

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i$ = b + x_1w_1 + x_2w_2

y_in =-2.8 + (0)(1.4) + (0)(2.1)= -2.8

hasil aktivasi = y = -1 (y_in < -0.3)

target t = -1

y = t

j) Iterasi Kesepuluh:

Data pertama: (1 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.8 + (1)(1.4) + (1)(2.1) = 0.7$

hasil aktivasi = y = 1 (y_in > 0.3) target t = 1

y = t

Data kedua: (1 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.8 + (1)(1.4) + (0)(2.1) = -1.4$

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

y = t

Data ketiga: (0 1)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.8 + (0)(1.4) + (1)(2.1) = -0.7$

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

y = t

Data keempat: (0 0)

$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$

$y\_in = -2.8 + (0)(1.4) + (0)(2.1) = -2.8$

hasil aktivasi = y = -1 (y_in < -0.3) target t = -1

y = t

Pada **iterasi kesepuluh** tidak terjadi perubahan bobot di mana nilai $y\_in = t$ pada proses pelatihan. Oleh karena itu proses dihentikan. Hasil akhir dari proses pembelajaran adalah:

$w_1 = 1.4$;

$w_2 = 2.1$;

$b = -2.8$

Tabel 2.2: Hasil Testing Fungsi Logika AND: *input biner* dan *target bipolar*

Input		Bobot		Bias	$y\_in = b + \sum_i x_i w_i$	Output	Target
X1	X2	W1	W2	b		Y=f(net)	
1	1	1.4	2.1	-2.8	0.7	1	1
1	0	1.4	2.1	-2.8	-1.4	-1	-1
0	1	1.4	2.1	-2.8	-0.7	-1	-1
0	0	1.4	2.1	-2.8	-2.8	-1	-1

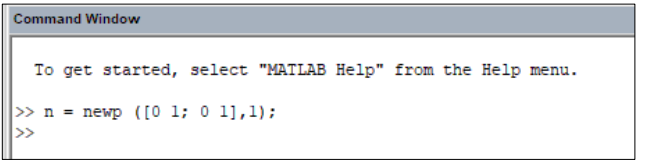
2.2 Penerapan Aplikasi

Model jaringan Perseptron dapat dimplementasikan dengan cara yang sederhana melalui software *matlab 6.1*. Pengguna bisa menggunakan software *matlab* versi tinggi yang disesuaikan dengan pemakaian laptop/ PC masing – masing. Untuk penggunaan software *matlab 6.1* (versi terendah), minimal *Prosesor* i3 dan *Random Access Memory* (RAM) 4 Gb. Semakin tinggi semakin baik. Pada Matlab telah tersedia *toolbox* yang dapat dimanfaatkan untuk menyelesaikan sebuah model jaringan perceptron. Untuk mengeksekusi perceptron pada Matlab, pengguna dapat menggunakan perintah **newp**, perintah ini membentuk sebuah perceptron dengan spesifikasi terntentu berdasarkan jumlah *input*, jumlah neuron, fungsi aktivasi atau lainnya.

Contoh kasus: adalah ingin membuat perceptron untuk mengenali logika AND dua variabel X1 dan X2. Logika AND tersebut dengan dua variabel masing masing memiliki range nilai masukan [0, 1] dan sebuah target. Maka sintaks yang dituliskan pada Command Window:

```
n = newp([0 1; 0 1],1);
```

Penjelasan: Matrik [0 1] pertama menunjukkan nilai X1 dan yang kedua menunjukkan nilai X2 dan 1 menginisialisasi jaringan yang hanya memiliki satu target/ neuron. Matlab memberikan nilai beupa bobot awal, bias hingga perubahan bobot dengan nilai *default*.

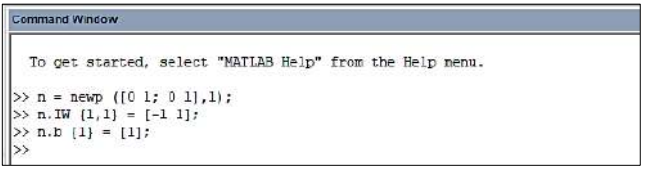


Gambar 2.2: Pembentukan model jaringan Perceptron

Langkah selanjutnya adalah memasukkan nilai bobot dan bias pada model jaringan Perseptron dengan sintaks yang dituliskan pada Command Window:

```
n.IW {1,1} = [-1 1];

n.b {1} = [1];
```

Penjelasan: sintaks n.IW {1,1} = [-1 1] merupakan bobot awal (w) yang diberikan di mana nilai -1 untuk bobot *input* X1 dan 1 untuk bobot *input* X2. Sedangkan untuk sintaks n.b {1} = [1]; untuk memberikan bobot bias awal dengan nilai 1.

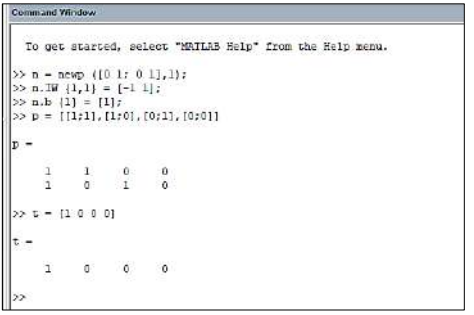


Gambar 2.3: Sintaks pemberian bobot dan bias model jaringan Perceptron

Langkah selanjutkan memasukkan nilai *input* dan target pada model jaringan Perseptron dengan sintaks yang dituliskan pada Command Window:

```
p = [[1;1],[1;0],[0;1],[0;0]];

t = [1 0 0 0];
```

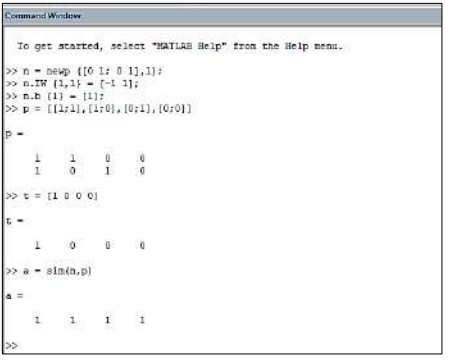
Penjelasan: variabel p merupakan nama *input* yang diberikan dengan menggunakan matrik ordo 2x2 [[1;1],[1;0],[0;1],[0;0]]; dan variabel t merupakan nama target yang diberikan [1 0 0 0];.



Gambar 2.4: Sintaks memasukkan nilai *input* dan target model jaringan Perceptron

Langkah berikutnya setelah membuat input ialah melakukan perhitungan untuk menghasilkan suatu keluaran jaringan. Pada Matlab pengguna dapat menggunakan perintah **sim** untuk menghitung keluaran Perceptron dengan sintaks yang dituliskan pada Command Window:

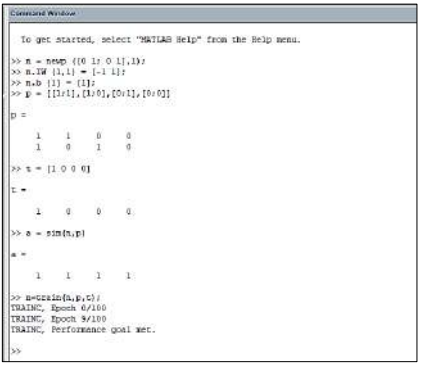
```
a = sim(n,p)
```



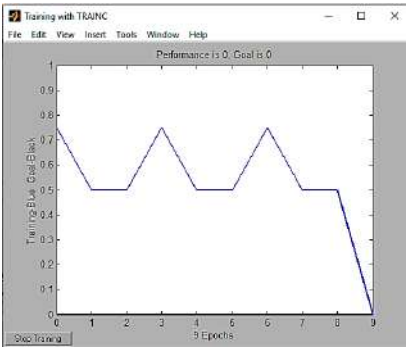
Gambar 2.5: Sintaks menghitung keluaran Perceptron

Penjelasan: Luaran yang dihasilkan pada sistem belum sesuai dengan target yang diinginkan. Target yang diinginkan [1 0 0 0], sementara hasil yang diberikan lewat variabel a = [1 1 1 1]. Artinya sistem harus di *training* untuk mendapatkan target sesuai dengan yang diinginkan. Berikut sintaks yang dituliskan pada Command Window:

```
n = train(n, p , t)
```



Gambar 2.6: Proses training yang sudah mencapai *goal* (target)

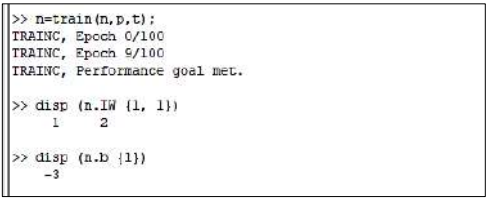


Gambar 2.7: *Goal* met

Pada gambar 2.7 dapat dijelaskan bahwa sistem telah berhasil mencapai *goal* dengan target yang sudah ditentukan (t = [1 0 0 0]). Proses training berhenti pada iterasi 9 (9 *epochs*). Untuk melihat bobot (w) terakhir dan bias (b), berikut sintaks yang dituliskan pada Command Window:

```
disp (n.IW {1, 1})

disp (n.b {1})
```

Penjelasan: Sintaks ini untuk menampilkan nilai bobot (w) dan bias (b) pada proses training ketika sistem sudah mencapai *goal*.



Gambar 2.8: Nilai bobot (w) dan bias (b)

Dapat disimpukan bahwa untuk fungsi logika AND menggunakan model jaringan Perseptron bantuan software *matlab 6.1* menghasilkan hasil akhir dengan nilai

$w_1 = 1$;

$w_2 = 2$;

$b = -3$

Bab 3

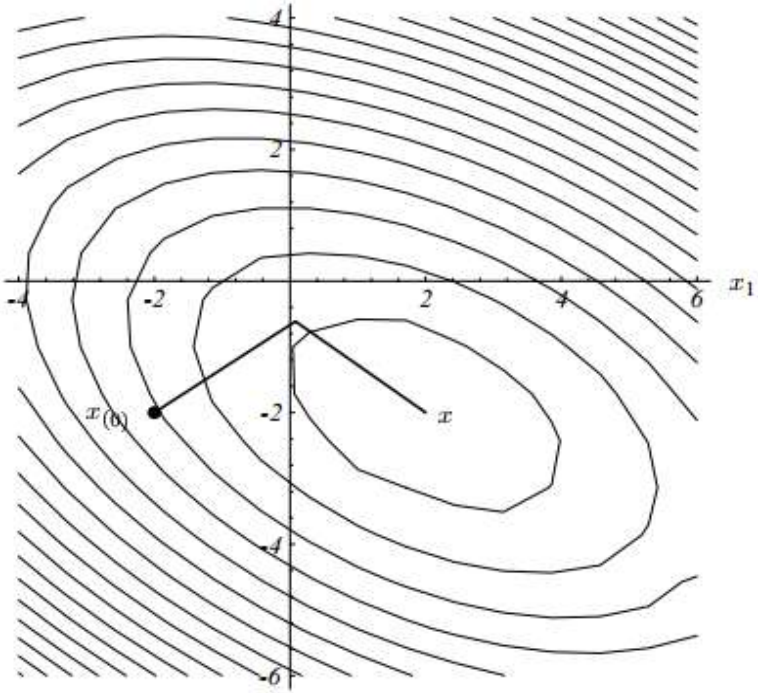
Algoritma Conjugate Gradient Polak Rebiere untuk Prediksi Data

3.1 Pendahuluan

Dalam ilmu matematika, *Conjugate gradient* adalah algoritma yang digunakan untuk solusi numerik dari sistem persamaan linear tertentu, yaitu matriks yang simetris dan bernilai positif. Algoritma conjugate gradien akan bekerja pada sistem di mana matriks A simetris, dan pasti positif (tidak perlu dominan secara diagonal dalam kasus ini). Definitif positif berarti bahwa untuk setiap x yang tidak semuanya nol (McClarren, 2018). Algoritma *Conjugate gradient* sering diimplementasikan sebagai algoritma iteratif, terutama untuk penyebaran sistem yang terlalu besar untuk ditangani oleh implementasi langsung atau metode langsung lainnya seperti dekomposisi Cholesky. Penyebaran Sistem sering muncul ketika memecahkan persamaan diferensial parsial atau masalah optimasi secara numerik. Algoritma *Conjugate gradient* juga dapat digunakan untuk memecahkan masalah optimasi yang tidak terbatas seperti minimalisasi energi seperti yang dikembangkan oleh (Hestenes and Stiefel, 1952; Straeter, 1971). Algoritma *Conjugate gradient* biasanya jauh lebih efisien daripada metode berbasis gradient descent, karena waktu penyelesaian yang lebih cepat dan iterasi yang tidak terlalu banyak (Berisha and Nagy, 2014).

Algoritma *Conjugate gradient* merupakan teknik perulangan sederhana dan kuat untuk menyelesaikan masalah minimalisasi linier dan nonlinier. Metode ini dijelaskan secara rinci dalam sejumlah buku (Alifanov and M, 1994; Ozisik

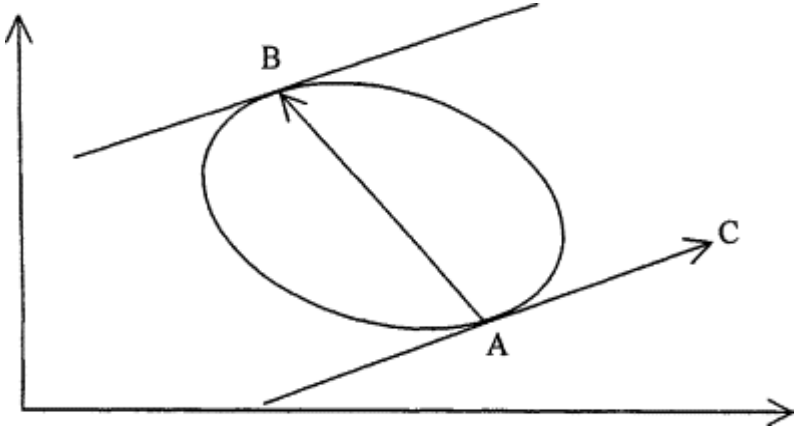
and Orlande, 2000). Pada algoritma ini, arah turunan ditemukan sebagai konjugasi dari arah gradien dan dari arah turunan sebelumnya (Modest, 2013). Kinerja *Conjugate gradient* pada beberapa masalah dapat ditunjukkan pada gambar berikut.



Gambar 3.1: Algoritma *Conjugate gradient*
(Shewchuk, 1994)

Metode *Conjugate gradient* mengeksploitasi konsep konjugasi dengan menggunakan informasi gradien. Untuk memahami properti konjugasi elips, dapat dilihat Gambar 1.2. Menurut properti konjugasi, garis AB yang menghubungkan titik-titik hubungan dua garis singgung sejajar dengan elips harus melewati pusat elips. Pada Gambar 1.2, arah AB dan AC merupakan arah konjugasi. Secara matematis, untuk kasus n-dimensional, jika A adalah n * n matriks simetris definitif positif, maka kumpulan arah {Si} dikatakan konjugasi jika,

Si A Sj = 0 untuk semua i≠ j dan i = 1, 2, ..., n; j = 1,2, ... n (1)



Gambar 3.2: Konsep Arah Konjugasi
(S.K.Jain and V.P.Singh, 2003)

Metode yang menemukan fungsi kuadrat minimum variabel n dalam langkah-langkah yang nomornya terkait dengan n, dikenal sebagai konvergen kuadratik. Konvergensi dari metode penurunan curam dapat ditingkatkan secara signifikan dengan mengubahnya menjadi metode *Conjugate gradient* (konjugasi gradien). Untuk menyelesaikan pencarian fungsi di sepanjang n arah konjugasi, diperlukan siklus minimalisasi. Setelah siklus ini, semua arah pencarian saling konjugat (pencarian juga telah dilakukan sepanjang arah koordinat atau non-konjugat) dan minimum kuadratik akan ditemukan.

Pada bab ini akan dibahas mengenai jenis-jenis algoritma *Conjugate gradient* secara umum yang sering digunakan untuk melakukan optimasi prediksi data dari metode backpropagation. Hal ini karena perhitungan dan pelatihan yang dilakukan algoritma backpropagation membutuhkan waktu yang relatif cukup lama, maka digunakan algoritma *Conjugate gradient* untuk mempercepat kinerja dari Algoritma Backpropagation. Algoritma *Conjugate gradient* (CG) merupakan salah satu metode optimasi yang arah pencariannya di dasarkan pada arah konjugasi yang nilainya ortogonal (Wanto, 2017).

Secara umum Algoritma *Conjugate gradient* terdiri dari tiga metode yaitu Fletcher-Reeves (CGF), *Polak-Ribiere* (CGP) dan Powell-Beale Restarts (CGB). Setiap metode memiliki kelebihan dan kekurangan masing-masing tergantung kasus/pola data yang ingin diselesaikan. Akan tetapi pada bab ini penulis hanya akan membahas *Conjugate gradient Polak-Ribiere* dan menjelaskan fungsi dan paramater-parameter yang akan digunakan

berdasarkan Tools Matlab. Karena pada tahap akhir nanti akan dilakukan pengujian dengan menggunakan aplikasi Matlab 2011b.

3.2 Conjugate gradient Polak-Ribiere

Conjugate gradient Polak-Ribiere adalah fungsi pelatihan jaringan yang memperbarui nilai bobot dan nilai bias sesuai dengan konjugasi gradient propagasi balik dengan pembaruan *Polak-Ribiere* (MathWorks, 1994).

Versi lain dari algoritma *Conjugate gradient* diusulkan oleh Polak dan Ribiere. Seperti halnya algoritma Fletcher-Reeves (traincgf), arah pencarian pada setiap iterasi ditentukan oleh rumus:

$$P_k = -g_k + \beta_k P_k - 1 \quad (2)$$

Untuk pembaruan *Polak-Ribiere*, konstanta β_k dihitung berdasarkan rumus:

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}} \quad (3)$$

Ini adalah produk terdalam pada perubahan gradien sebelumnya dengan gradien saat ini dibagi dengan norma kuadrat dari gradien sebelumnya. Untuk pembahasan algoritma *Conjugate gradient Polak-Ribiere* yang lebih rinci dapat dibaca pada pembahasan (Fletcher and Reeves, 1964; Hagan, Demuth and Beale, 1996)

Rutin *traincgp* memiliki kinerja yang mirip dengan *traincgf*. Sulit untuk memprediksi algoritma mana yang akan bekerja paling baik pada masalah yang diberikan. Persyaratan penyimpanan untuk *Polak-Ribiere* (empat vektor) sedikit lebih besar daripada *Fletcher-Reeves* (tiga vektor).

3.2.1 Algoritma

Pada aplikasi Matlab, *Conjugate gradient Polak-Ribiere* ditulis dengan "*traincgp*". *Conjugate gradient Polak-Ribiere* (*traincgp*) dapat melatih jaringan apa pun asalkan bobotnya, *input* jaringan, dan fungsi transfer memiliki fungsi turunan, sama halnya seperti pada algoritma *Conjugate gradient* Fletcher-Reeves. Backpropagation digunakan untuk menghitung kinerja turunan (*performance*) dengan hubungan bobot dan variabel bias X. Setiap variabel disesuaikan dengan persamaan berikut:

$$X = X + a * dX \quad (4)$$

di mana dX adalah arah pencarian. Parameter a dipilih untuk meminimalkan kinerja di sepanjang arah pencarian. Fungsi pencarian baris *searchFcn* digunakan untuk menemukan titik minimum. Arah pencarian pertama bernilai negatif dari kinerja gradien. Dalam iterasi berikutnya, arah pencarian dihitung dari gradien baru dan arah pencarian sebelumnya, sesuai dengan rumus:

$$dX = -gX + dX\_old * Z \quad (5)$$

di mana *gX* adalah gradien. Parameter *Z* dapat dihitung dengan beberapa cara berbeda. Untuk variasi *Polak-Ribiere* dari *Conjugate gradient* itu dihitung menurut persamaan berikut:

$$Z = ((gX - gX\_old)'*gX)/norm\_sqr \quad (6)$$

di mana *norm_sqr* adalah kuadrat norma dari gradien sebelumnya dan *gX_old* adalah gradien pada iterasi sebelumnya. Lihat halaman 78 dari buku (Scales, 1985) untuk pembahasan algoritma yang lebih rinci.

Pelatihan (*training*) pada algoritma *Conjugate gradient Polak-Ribiere* akan berhenti ketika salah satu dari kondisi ini terjadi:

- Jumlah maksimum iterasi (pengulangan) tercapai.
- Jumlah waktu maksimum terlampaui.
- Kinerja diminimalkan untuk mencapai tujuan (goal).
- Kinerja Gradien berada di bawah *min_grad*.
- Performa validasi telah meningkat lebih dari *max_fail* kali sejak terakhir kali menurun (saat menggunakan validasi)

Kita dapat membuat jaringan standar menggunakan *traincgp* dengan *feedforwardnet* atau *cascadeforwardnet.* Untuk mempersiapkan jaringan khusus untuk dilatih dengan *traincgp* dapat dilakukan dengan cara:

1. Terapkan *net.trainFcn* ke '*traincgp*'. Terapkan *net.trainParam* ke *traincgp's* untuk melatih parameter default.
2. Terapkan properti *net.trainParam* ke nilai yang diinginkan.

Dalam kedua kasus ini, memanggil *train* dengan jaringan hasil pelatihan menggunakan jaringan *traincgp*.

3.2.2 Deskripsi

Traincgp merupakan fungsi pelatihan jaringan yang memperbarui nilai bobot dan nilai bias sesuai algoritma *Conjugate gradient backpropagation* dengan *Polak-Ribiere updates*. Sintaksis:

net.trainFcn = 'traincgp' // untuk menetapkan properti jaringan *trainFcn*.
[net,tr] = train(net,...) // untuk melatih jaringan dengan *traincgp.*

Pelatihan (*training*) terjadi sesuai dengan parameter pelatihan *traincgp*. Adapun parameter default dari algoritma *Conjugate gradient Polak-Ribiere updates* yang akan digunakan di Aplikasi Matlab untuk proses prediksi dapat dilihat pada tabel berikut:

Tabel 3.1: Parameter Default *Conjugate gradient Polak-Ribiere*

Parameter	Nilai	Keterangan
net.trainParam.epochs	1000	Jumlah maksimum epoch untuk dilatih
net.trainParam.show	25	Epoch di antara tampilan (NaN tanpa tampilan)
net.trainParam.showCommandLine	False	Hasilkan *output* baris perintah
net.trainParam.showWindow	True	Tampilkan pelatihan GUI
net.trainParam.goal	0	Tujuan kinerja
net.trainParam.time	Inf	Maksimum waktu pelatihan dalam hitungan detik
net.trainParam.min_grad	1e-10	Gradien kinerja minimum
net.trainParam.max_fail	6	Kegagalan validasi maksimum
net.trainParam.searchFcn	'srchcha'	Nama rutin pencarian baris untuk digunakan

Parameter yang terkait dengan metode pencarian baris (tidak semua digunakan untuk semua metode) dapat dilihat pada tabel berikut:

Tabel 3.2: Parameter Terkait *Conjugate gradient Polak-Ribiere*

Parameter	Nilai	Keterangan
net.trainParam.scal_tol	20	Membagi menjadi delta untuk menentukan toleransi untuk pencarian linier.
net.trainParam.alpha	0.001	Faktor skala yang menentukan pengurangan *performance* yang memadai
net.trainParam.beta	0.1	Faktor skala yang menentukan ukuran langkah yang cukup besar
net.trainParam.delta	0.01	Ukuran langkah awal dalam langkah lokasi interval
net.trainParam.gama	0.1	Parameter untuk menghindari pengurangan kecil dalam kinerja, biasanya ditetapkan ke 0,1 (lihat srch_cha)
net.trainParam.low_lim	0.1	Batas bawah pada perubahan ukuran langkah
net.trainParam.up_lim	0.5	Batas atas perubahan ukuran langkah
net.trainParam.maxstep	100	Panjang langkah maksimum
net.trainParam.minstep	1.0e-6	Panjang langkah minimum
net.trainParam.bmax	26	Ukuran langkah maksimum

Catatan : Parameter yang tertera pada tabel 3.3 dan 3.4 pada pembahasan *Conjugate gradient Polak-Ribiere* sama dengan parameter yang tertera pada tabel 3.1. dan 3.2 pada pembahasan *Conjugate gradient Fletcher-Reeves*. Perbedaan nya adalah pada fungsi pelatihan yang digunakan, yakni jika *Fletcher-Reeves menggunakan traincgf*, sedangkan *Polak-Ribiere* menggunakan *traincgp*.

3.3 Contoh Kasus : Prediksi Data dengan Algoritma Conjugate gradient Polak-Ribiere

Pada pembahasan ini akan diberikan contoh kasus prediksi data dengan menggunakan algoritma *Conjugate gradient Polak-Ribiere*. Penyelesaian dilakukan dengan menggunakan microsoft Excel (untuk perhitungan) dan Tools Matlab (untuk proses *training* dan *Testing* data dengan algoritma *Conjugate gradient Polak-Ribiere*).

Contoh Kasus:

Angka Harapan Hidup (AHH) merupakan rata-rata jumlah tahun kehidupan yang masih dijalani oleh seseorang yang telah berhasil mencapai umur tertentu. Manfaat mengetahui Angka Harapan Hidup adalah untuk mengevaluasi kinerja pemerintah pada sebuah negara dalam meningkatkan kesejahteraan rakyat pada umumnya, dan meningkatkan derajat kesehatan pada khususnya. Oleh karena itu, pada contoh kasus ini akan membahas tentang prediksi Angka Harapan Hidup Penduduk Dunia yang terdiri dari 38 negara. Data AHH ini nantinya akan dianalisis menggunakan algoritma *Conjugate gradient Polak-Ribiere* sehingga menghasilkan prediksi data untuk tahun-tahun berikutnya. Data AHH penduduk dunia dapat dilihat pada tabel berikut:

Tabel 3.3: Angka Harapan Hidup Penduduk Dunia (Umur/Tahun

No	Negara	Umur/Tahun			
		1995-2000	2000-2005	2005-2010	2010-2015
1	United States	76,4	77,1	78,1	78,9
2	Saudi Arabia	71,6	73,1	74,3	75,4
3	Australia	78,9	80,4	81,7	82,4
4	Bangladesh	64,1	66,4	68,4	70,5
5	Netherlands	77,8	78,7	80,2	80,9
6	Belgium	77,3	78,3	79,5	80,4
7	Brazil	69,4	71,0	72,4	73,8
8	China	70,9	73,4	74,4	75,2
9	Denmark	76,0	77,3	78,6	79,3
10	Russian	65,7	65,0	67,2	67,9
11	The Philippines	66,4	67,1	67,8	68,6

No	Negara	Umur/Tahun			
		1995-2000	2000-2005	2005-2010	2010-2015
12	Finland	77,0	78,3	79,5	80,5
13	Hong Kong	79,4	81,3	82,4	83,3
14	India	61,2	63,1	64,9	66,3
15	Indonesia	66,0	67,8	69,1	70,1
16	English	77,1	78,4	79,6	80,4
17	Italy	78,7	80,2	81,5	82,3
18	Japan	80,5	81,8	82,7	83,5
19	German	77,2	78,6	79,8	80,7
20	Cambodia	59,8	64,5	69,5	71,6
21	Canada	78,5	79,7	80,5	81,4
22	Kazakhstan	63,0	64,6	65,7	66,4
23	South Korea	74,9	77,4	80,0	81,4
24	Kuwait	72,9	73,4	73,8	74,2
25	Malaysia	72,3	73,3	74,0	74,9
26	Mexico	68,0	69,0	69,9	71,1
27	Egypt	73,7	75,0	76,3	77,4
28	Myanmar	61,3	62,8	64,2	65,1
29	Nigeria	46,3	47,3	50,2	52,3
30	Norway	78,2	79,2	80,6	81,4
31	Pakistan	63,1	64,5	65,7	66,5
32	France	78,3	79,5	80,9	81,7
33	Singapore	77,7	79,2	81,2	82,2
34	Sri Lanka	69,1	73,2	73,4	74,2
35	Sweden	79,2	80,1	81,1	81,7
36	Thailand	70,6	71,5	73,3	74,3
37	Venezuela	72,1	72,8	73,7	74,5
38	Vietnamese	73,0	74,4	75,1	75,9

Sumber:

1. United Nations: “World Population Prospect: The 2010 Revision Population Database”.
2. Badan Pusat Statistik (BPS) Indonesia.

Penyelesaian:
Tahapan penyelesaian untuk Prediksi Angka Harapan Hidup (AHH) Penduduk Dunia menggunakan algoritma *Conjugate gradient Polak-Ribiere* dapat di jabarkan sebagai berikut:

1. Pembagian Data

Langkah pertama yang harus dilakukan adalah membagi data menjadi 2 bagian, yakni untuk data *training* dan data *Testing*. Pada contoh kasus ini penulis menjadikan dataset tahun 1995-2000, 2000-2005, 2005-2010 sebagai data *training* dengan target tahun 2005-2010. Sedangkan untuk data *Testing* penulis mengambil tahun 2000-2005, 2005-2010, 2010-2015 dengan target tahun 2010-2015.

Tabel 3.4: Data *Training*

Data	*Input* (Umur/Tahun)			Target (2005-2010)
	1995-2000	2000-2005	2005-2010	
1	76,4	77,1	78,1	78,1
2	71,6	73,1	74,3	74,3
...	...	...	...	...
37	72,1	72,8	73,7	73,7
38	73,0	74,4	75,1	75,1

Tabel 3.5: Data *Testing*

Data	*Input* (Umur/Tahun)			Target (2010-2015)
	2000-2005	2005-2010	2010-2015	
1	77,1	78,1	78,9	78,9
2	73,1	74,3	75,4	75,4
...	...	...	...	...
37	72,8	73,7	74,5	74,5
38	74,4	75,1	75,9	75,9

Letak dan urutan data sudah berbeda

Catatan:

a. Membagi dataset menjadi dua bagian untuk data *training* dan *Testing* tidak harus dibagi rata, apalagi apabila dataset yang digunakan tidak terlalu banyak. Asalkan data *training* dan target *training* dengan data *Testing* dan target *Testing* tidak sama.
b. Tahun dataset AHH dapat digunakan kembali sebagai data *Testing* maupun target, asalkan letak dan urutannya sudah berbeda (antara data *training* dan data *Testing*).
c. Jumlah *input* tahun data *training* dan data *Testing* harus sama. Seperti yang disajikan pada tabel 3.4, *input* nya ada 3 dengan 1 target. Begitu pula pada tabel 3.5, *input* nya juga 3 dengan 1 target pula.

2. *Normalisasi*

Normalisasi data *training* dan data *Testing* perlu dilakukan untuk mengubah data menjadi bernilai antara angka 0 dan 1 (0,1-0,9). Hasil normalisasi ini nantinya akan diproses dan dimasukkan kedalam aplikasi Matlab. Berikut rumus normalisasi yang digunakan (Afriliansyah *et al.*, 2019; Bhawika *et al.*, 2019; Parulian *et al.*, 2019; Purba *et al.*, 2019; Siregar *et al.*, 2019; Sormin *et al.*, 2019; Wanto *et al.*, 2019):

$$x' = \frac{0.8\,(x-a)}{b-a} + 0.1 \quad (7)$$

Keterangan :

x' = *Hasil Normalisasi*
0.8 = *Nilai default normalisasi nilai optimum*
x = *Data yang akan dinormalisasi*
b = *Nilai data tertinggi*
a = *Nilai data terendah*
0.1 = *Nilai default normalisasi nilai minimum*

Sebelum dilakukan normalisasi, kita harus mengetahui nilai data tertinggi (b) dan data terendah (a) dari tabel data *training* maupun data *Testing*.

a. Normalisasi Data Training

Berdasarkan tabel 3.4 nilai data tertinggi (b) = 82,70 sedangkan nilai terendah (a) = 46,30. Berikut contoh perhitungannya berdasarkan persamaan (15):

$$x' = \frac{0.8\,(76{,}4 - 46{,}30)}{82{,}70 - 46{,}30} + 0.1 = 0{,}761538$$

Begitu seterusnya lakukan hal yang sama hingga data 38 (Vietnamese). Hasil Normalisasi data *Training* secara keseluruhan dapat dilihat pada tabel berikut.

Tabel 3.6: Hasil Normalisasi Data *Training*

Data	*Input* (Dalam Tahun)			Target
	1995-2000	2000-2005	2005-2010	
1	0,761538	0,776923	0,798901	0,798901
2	0,656044	0,689011	0,715385	0,715385
3	0,816484	0,849451	0,878022	0,878022
4	0,491209	0,541758	0,585714	0,585714
5	0,792308	0,812088	0,845055	0,845055
6	0,781319	0,803297	0,829670	0,829670

Data	*Input* (Dalam Tahun)			Target
	1995-2000	2000-2005	2005-2010	
7	0,607692	0,642857	0,673626	0,673626
8	0,640659	0,695604	0,717582	0,717582
9	0,752747	0,781319	0,809890	0,809890
10	0,526374	0,510989	0,559341	0,559341
11	0,541758	0,557143	0,572527	0,572527
12	0,774725	0,803297	0,829670	0,829670
13	0,827473	0,869231	0,893407	0,893407
14	0,427473	0,469231	0,508791	0,508791
15	0,532967	0,572527	0,601099	0,601099
16	0,776923	0,805495	0,831868	0,831868
17	0,812088	0,845055	0,873626	0,873626
18	0,851648	0,880220	0,900000	0,900000
19	0,779121	0,809890	0,836264	0,836264
20	0,396703	0,500000	0,609890	0,609890
21	0,807692	0,834066	0,851648	0,851648
22	0,467033	0,502198	0,526374	0,526374
23	0,728571	0,783516	0,840659	0,840659
24	0,684615	0,695604	0,704396	0,704396
25	0,671429	0,693407	0,708791	0,708791
26	0,576923	0,598901	0,618681	0,618681
27	0,702198	0,730769	0,759341	0,759341
28	0,429670	0,462637	0,493407	0,493407
29	0,100000	0,121978	0,185714	0,185714
30	0,801099	0,823077	0,853846	0,853846
31	0,469231	0,500000	0,526374	0,526374
32	0,803297	0,829670	0,860440	0,860440
33	0,790110	0,823077	0,867033	0,867033
34	0,601099	0,691209	0,695604	0,695604
35	0,823077	0,842857	0,864835	0,864835
36	0,634066	0,653846	0,693407	0,693407
37	0,667033	0,682418	0,702198	0,702198
38	0,686813	0,717582	0,732967	0,732967

b. Normalisasi Data Testing

Berdasarkan tabel 3.5 nilai data tertinggi (b) = 83,50 sedangkan nilai terendah (a) = 47,30. Berikut contoh perhitungannya berdasarkan persamaan (15):

$$x' = \frac{0.8\ (77{,}1 - 47{,}30)}{83{,}50 - 47{,}30} + 0.1 = 0{,}758564$$

Begitu seterusnya lakukan hal yang sama hingga data 38 (Vietnamese). Hasil Normalisasi data *Testing* secara keseluruhan dapat dilihat pada tabel berikut.

Tabel 3.7: Hasil Normalisasi Data *Testing*

Data	*Input* (Dalam Tahun)			Target
	2000-2005	2005-2010	2010-2015	
1	0,758564	0,780663	0,798343	0,798343
2	0,670166	0,696685	0,720994	0,720994
3	0,831492	0,860221	0,875691	0,875691
4	0,522099	0,566298	0,612707	0,612707
5	0,793923	0,827072	0,842541	0,842541
6	0,785083	0,811602	0,831492	0,831492
7	0,623757	0,654696	0,685635	0,685635
8	0,676796	0,698895	0,716575	0,716575
9	0,762983	0,791713	0,807182	0,807182
10	0,491160	0,539779	0,555249	0,555249
11	0,537569	0,553039	0,570718	0,570718
12	0,785083	0,811602	0,833702	0,833702
13	0,851381	0,875691	0,895580	0,895580
14	0,449171	0,488950	0,519890	0,519890
15	0,553039	0,581768	0,603867	0,603867
16	0,787293	0,813812	0,831492	0,831492
17	0,827072	0,855801	0,873481	0,873481
18	0,862431	0,882320	0,900000	0,900000
19	0,791713	0,818232	0,838122	0,838122
20	0,480110	0,590608	0,637017	0,637017
21	0,816022	0,833702	0,853591	0,853591
22	0,482320	0,506630	0,522099	0,522099
23	0,765193	0,822652	0,853591	0,853591
24	0,676796	0,685635	0,694475	0,694475
25	0,674586	0,690055	0,709945	0,709945

Data	*Input* (Dalam Tahun)			Target
	2000-2005	2005-2010	2010-2015	
26	0,579558	0,599448	0,625967	0,625967
27	0,712155	0,740884	0,765193	0,765193
28	0,442541	0,473481	0,493370	0,493370
29	0,100000	0,164088	0,210497	0,210497
30	0,804972	0,835912	0,853591	0,853591
31	0,480110	0,506630	0,524309	0,524309
32	0,811602	0,842541	0,860221	0,860221
33	0,804972	0,849171	0,871271	0,871271
34	0,672376	0,676796	0,694475	0,694475
35	0,824862	0,846961	0,860221	0,860221
36	0,634807	0,674586	0,696685	0,696685
37	0,663536	0,683425	0,701105	0,701105
38	0,698895	0,714365	0,732044	0,732044

3. Menentukan Model Arsitektur Jaringan

Langkah selanjutnya adalah menentukan model arsitektur jaringan yang akan digunakan untuk proses analisis dan membantu perhitungan untuk prediksi. Tidak ada aturan baku yang mengharuskan penggunaan model arsitektur jaringan tertentu. Biasanya model arsitektur yang digunakan dimulai dari nilai yang kecil berlanjut kenilai yang lebih besar. Pada kasus ini, model arsitektur yang digunakan ada 3, antara lain;

a. 3-5-1 (3 adalah *input*, 1 *hidden* layer dengan 5 neuron dan 1 *output*)
b. 3-10-1 (3 adalah *input*, 1 *hidden* layer dengan 10 neuron dan 1 *output*)
c. 3-5-10-1 (3 adalah *input*, 2 *hidden* layer dengan 5 neuron dan 10 neuron serta 1 *output*).

Catatan : ***Input*** (masukkan) = **3** ditentukan berdasarkan *input* tabel, nilainya tidak bisa diubah, karena sudah ditetapkan di awal pada tabel. *Hidden* layer bisa diubah, bisa menggunakan 1 atau 2 *hidden* layer. Neuron *hidden* layer juga bisa diubah nilainya sesuai keinginan untuk mencari hasil yang terbaik (pada kasus ini menggunakan **5** dan **10** neuron). Sedangkan ***output* = 1** ditentukan berdasarkan **target** tabel pada masing-masing data yang hanya menampung satu nilai.

4. *Pelatihan (Training) dan Pengujian (Testing) Data*

Berikutnya adalah melakukan pelatihan data. Pelatihan data dilakukan untuk melatih data yang digunakan agar sistem berjalan dengan konsisten dan akurat berdasarkan model pola jaringan yang digunakan. Sedangkan Pengujian data dilakukan untuk mencoba mengenali pola-pola masukan yang diujikan untuk kemudian dicocokkan dengan model hasil dari proses *training* (pelatihan). Langkah-langkah pelatihan dan pengujian data berdasarkan model arsitektur jaringan yang digunakan dapat dilakukan sebagai berikut :

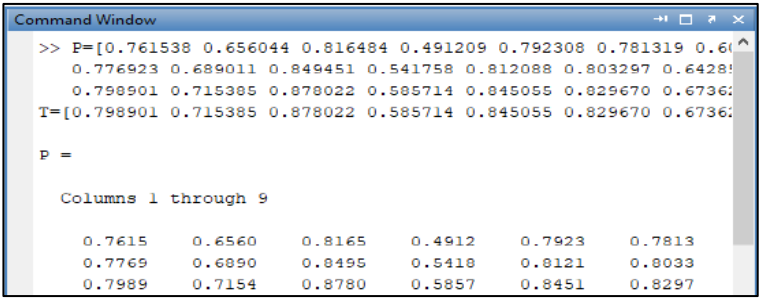
a. Pelatihan dan Pengujian dengan Model Arsitektur 3-5-1

- Buka aplikasi **Matlab** (Penulis menggunakan **Matlab 2011b**). Kemudian masukkan normalisasi data *training* berdasarkan tabel 3.6 (Hasil Normalisasi Data *Training*) secara horizontal (mendatar) seperti berikut:

```
P=[0.761538 0.656044 0.816484 0.491209 0.792308
   0.781319 0.607692 .......dst hingga 0.686813;
   0.776923 0.689011 0.849451 0.541758 0.812088
   0.803297 0.642857 .......dst hingga 0.717582;
   0.798901 0.715385 0.878022 0.585714 0.845055
   0.829670 0.673626 .......dst hingga 0.732967]
T=[0.761538 0.656044 0.816484 0.491209 0.792308
   0.781319 0.607692 .......dst hingga 0.686813]
```

→ Enter

Hasil nya akan berupa normalisasi yang sudah diubah menjadi 4 desimal dibelakang koma. Seperti terlihat pada gambar berikut:



Gambar 3.3: *Input* Data *Training* di Matlab

Keterangan:
P = *Input* data *training*
T = Target data *training*

- Kemudian ketikkan kode program berikut (berdasarkan parameter tabel 3.1):

```
>> net=newff(minmax(P),[5,1],{'tansig','logsig'},'traincgp');
```
//Perintah ini untuk membentuk jaringan dengan 1 hiden layer sebanyak 5 neuron dan 1 *output*. Fungsi aktivasi yang digunakan adalah tansig (sigmoid bipolar) dan logsig (sigmoid biner) serta Fungsi pelatihan traincgp (Polak ribiere)

```
>> net.LW{1,1};
```
//Perintah ini untuk melihat nilai bobot awal pada lapisan masukan dan lapisan tersembunyi (bilangan diambil secara acak dari komputer)

```
>> net.b{1};
```
//Perintah ini untuk melihat nilai bias pada *hidden* layer/lapisan tersembunyi (bilangan diambil secara acak dari komputer)

```
>> net.LW{2,1};
```
//Perintah ini untuk melihat nilai bobot pada *hidden* layer/lapisan tersembunyi dan *output* layer/lapisan keluaran (bilangan diambil secara acak dari komputer)

```
>> net.b{2};
```
//Perintah ini untuk melihat nilai bias pada *output* layer/lapisan keluaran (bilangan diambil secara acak dari komputer)

```
>> net.trainParam.epochs=1000;
```
//Penjelasan dapat dilihat pada tabel 3.1

```
>> net.trainParam.show = 25;
```

```
>> net.trainParam.showCommandLine = false;
```
//Penjelasan dapat dilihat pada tabel 3.1

```
>> net.trainParam.showWindow = true;
```
//Penjelasan dapat dilihat pada tabel 3.1

```
>> net.trainParam.goal = 0;
```
//Penjelasan dapat dilihat pada tabel 3.1

```
>> net.trainParam.time = inf;
```
//Penjelasan dapat dilihat pada tabel 3.1

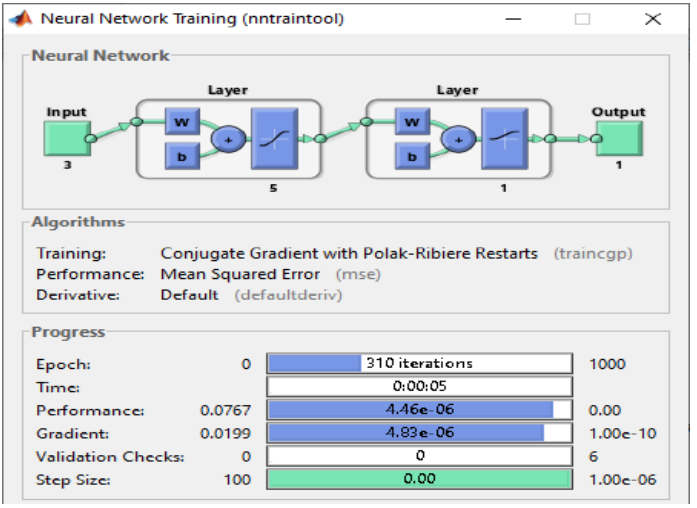
```
>> net.trainParam. min_grad= 1e-10;
```
//Penjelasan dapat dilihat pada tabel 3.1

```
>> net.trainParam.max_fail = 6;
```
//Penjelasan dapat dilihat pada tabel 3.1

```
>> net.trainParam.searchFcn = 'srchcha';
```
//Penjelasan dapat dilihat pada tabel 3.1

```
>> net=train(net,P,T)
```
//Penjelasan dapat dilihat pada tabel 3.1

Akan muncul gambar hasil pelatihan dengan algoritma *Conjugate gradient Polak-Ribiere* seperti berikut:



Gambar 3.4: Hasil Pelatihan dengan Model Arsitektur 3-5-1

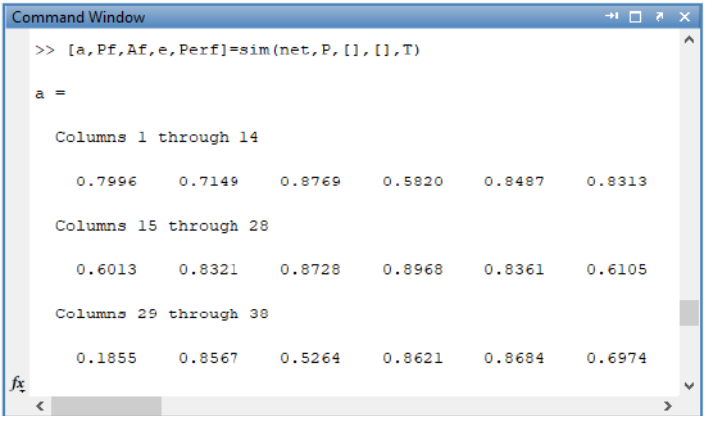
Untuk melihat tampilan *Plots*: *Performance*, *Training State* dan *regression* dapat dilakukan dengan menekan button pada gambar yang ada didalam *frame Plots*.

- Selanjutnya ketikkan kode berikut:

```
>> [a,Pf,Af,e,Perf]=sim(net,P,[],[],T)
```

//Kode ini untuk mendapatkan nilai *a* (*Output*) dan nilai *e* (*error*) data *training*.

Maka akan muncul gambar seperti berikut:



Gambar 3.5: Mendapatkan nilai a (*output*) dan nilai e (*error*) dengan Model Arsitektur 3-5-1

Kemudian pada *Ms. Excel* buat tabel seperti berikut:

Tabel 3.8: Hasil Data *Training* dengan Model 3-5-1

Data	Target	*Output*	*Error*	SSE
1	0,79890	0,79960	-0,00070	0,0000004885
2	0,71538	0,71490	0,00048	0,0000002349
3	0,87802	0,87690	0,00112	0,0000012588
4	0,58571	0,58200	0,00371	0,0000137959
5	0,84505	0,84870	-0,00365	0,0000132864
6	0,82967	0,83130	-0,00163	0,0000026558
7	0,67363	0,67490	-0,00127	0,0000016221
8	0,71758	0,71660	0,00098	0,0000009651
9	0,80989	0,80970	0,00019	0,0000000361
10	0,55934	0,55370	0,00564	0,0000318170
11	0,57253	0,57630	-0,00377	0,0000142320
12	0,82967	0,82980	-0,00013	0,0000000168
13	0,89341	0,88810	0,00531	0,0000281599
14	0,50879	0,50840	0,00039	0,0000001530
15	0,60110	0,60130	-0,00020	0,0000000404
16	0,83187	0,83210	-0,00023	0,0000000538
17	0,87363	0,87280	0,00083	0,0000006829
18	0,90000	0,89680	0,00320	0,0000102400
19	0,83626	0,83610	0,00016	0,0000000268
20	0,60989	0,61050	-0,00061	0,0000003720
21	0,85165	0,85150	0,00015	0,0000000220
22	0,52637	0,52720	-0,00083	0,0000006829
23	0,84066	0,83990	0,00076	0,0000005766
24	0,70440	0,70310	0,00130	0,0000016786
25	0,70879	0,70750	0,00129	0,0000016672
26	0,61868	0,62150	-0,00282	0,0000079450
27	0,75934	0,75820	0,00114	0,0000013011
28	0,49341	0,49570	-0,00229	0,0000052597
29	0,18571	0,18550	0,00021	0,0000000459
30	0,85385	0,85670	-0,00285	0,0000081444
31	0,52637	0,52640	-0,00003	0,0000000007
32	0,86044	0,86210	-0,00166	0,0000027571
33	0,86703	0,86840	-0,00137	0,0000018688
34	0,69560	0,69740	-0,00180	0,0000032242
35	0,86484	0,86700	-0,00216	0,0000046865
36	0,69341	0,69460	-0,00119	0,0000014242
37	0,70220	0,70200	0,00020	0,0000000391

Data	Target	*Output*	*Error*	SSE
38	0,73297	0,73010	0,00287	0,0000082199
			Jlh SSE	0,0001696823
			MSE	**0,0000044653**

Keterangan :

Data = 38 Negara
Target = Diambil dari normalisasi target data *training*
Output = Diambil dari nilai *a* hasil dari Matlab
Error = Diperoleh dari *Target – Output* atau dari nilai *e* hasil dari Matlab
SSE = *Error* ^ 2
Jlh SSE = Total nilai SSE
MSE = Jlh SSE / 38 (banyaknya data)

- Kemudian masukkan normalisasi data *Testing* berdasarkan tabel 3.7 (Hasil Normalisasi Data *Testing*) secara horizontal (mendatar) seperti berikut:

```
PP=[0.758564 0.670166 0.831492 0.522099 0.793923
    0.785083 0.623757 ........dst hingga 0.698895;
    0.780663 0.696685 0.860221 0.566298 0.827072
    0.811602 0.654696 .......dst hingga 0.714365;
    0.798343 0.720994 0.875691 0.612707 0.842541
    0.831492 0.685635 .......dst hingga 0.732044]
TT=[0.798343 0.720994 0.875691 0.612707 0.842541
    0.831492 0.685635 .......dst hingga 0.732044]
→ Enter
```

Hasil nya akan berupa normalisasi yang sudah diubah menjadi 4 desimal dibelakang koma.

Keterangan:

PP = *Input* data *Testing*
TT = Target data *Testing*

- Kemudian ketikkan kode program berikut:

```
>> [a,Pf,Af,e,Perf]=sim(net,PP,[],[],TT)
```

//Kode ini untuk mendapatkan nilai *a* (*Output*) dan nilai *e* (*error*) data *training*.

Kemudian pada *Ms. Excel* buat tabel seperti berikut:

Tabel 3.9: Hasil Akurasi Data *Testing* dengan Model 3-5-1

Data	Target	*Output*	*Error*	SSE	Akurasi
1	0,79834	0,79700	0,00134	0,0000018024	1
2	0,72099	0,72010	0,00089	0,0000008001	1
3	0,87569	0,87430	0,00139	0,0000019338	1
4	0,61271	0,60980	0,00291	0,0000084517	1
5	0,84254	0,84020	0,00234	0,0000054823	1
6	0,83149	0,83110	0,00039	0,0000001534	1
7	0,68564	0,68660	-0,00096	0,0000009305	1
8	0,71657	0,71520	0,00137	0,0000018895	1
9	0,80718	0,80460	0,00258	0,0000066684	1
10	0,55525	0,55790	-0,00265	0,0000070298	1
11	0,57072	0,57380	-0,00308	0,0000094973	1
12	0,83370	0,83380	-0,00010	0,0000000097	1
13	0,89558	0,89410	0,00148	0,0000021907	1
14	0,51989	0,51970	0,00019	0,0000000359	1
15	0,60387	0,60580	-0,00193	0,0000037349	1
16	0,83149	0,83080	0,00069	0,0000004785	1
17	0,87348	0,87250	0,00098	0,0000009617	1
18	0,90000	0,89890	0,00110	0,0000012100	1
19	0,83812	0,83800	0,00012	0,0000000148	1
20	0,63702	0,64060	-0,00358	0,0000128409	1
21	0,85359	0,85610	-0,00251	0,0000062943	1
22	0,52210	0,52570	-0,00360	0,0000129640	1
23	0,85359	0,84800	0,00559	0,0000312611	0
24	0,69448	0,69400	0,00048	0,0000002258	1
25	0,70994	0,70940	0,00054	0,0000002968	1
26	0,62597	0,62770	-0,00173	0,0000030038	1
27	0,76519	0,76320	0,00199	0,0000039735	1
28	0,49337	0,49800	-0,00463	0,0000214354	1
29	0,21050	0,20970	0,00080	0,0000006356	1
30	0,85359	0,85230	0,00129	0,0000016671	1
31	0,52431	0,52700	-0,00269	0,0000072394	1
32	0,86022	0,85890	0,00132	0,0000017450	1
33	0,87127	0,86690	0,00437	0,0000191032	0
34	0,69448	0,69500	-0,00052	0,0000002755	1
35	0,86022	0,86070	-0,00048	0,0000002294	1
36	0,69669	0,69680	-0,00011	0,0000000132	1
37	0,70110	0,70060	0,00050	0,0000002550	1

Data	Target	*Output*	*Error*	SSE	Akurasi
38	0,73204	0,73050	0,00154	0,0000023846	1
			Jlh SSE	0,0001791189	**95%**
			MSE	**0,0000047137**	

Keterangan :

Data = 38 Negara
Target = Diambil dari normalisasi target data *Testing*
Output = Diambil dari nilai *a* data *Testing* hasil Matlab
Error = Diperoleh dari *Target – Output* atau dari nilai *e* data *Testing* hasil dari Matlab
SSE = *Error* ^ 2
Jlh SSE = Total nilai SSE
MSE = Jlh SSE / 38 (banyaknya data)
Akurasi = Bernilai 1 (benar) apabila nilai *error* ≤ 0,003 (If *Error* ≤ 0,003;1;0)
0,003 = Minimum *error* yang digunakan (0,003-0,001)
95% = Diperoleh dari *Jumlah benar / Jumlah data* * *100* (36/38*100 = 94,74% = **95%** (dibulatkan))

b. Pelatihan dan Pengujian dengan Model Arsitektur 3-10-1

Untuk pelatihan dan pengujian dengan model arsitektur 3-10-1 hampir sama proses pengerjaannya dengan model arsitektur 3-5-1. Hanya saja ada sedikit perbedaan. Untuk lebih jelasnya dapat dilihat pada penjelasan berikut:

- Masukkan normalisasi data *training* berdasarkan tabel 3.6 (sama seperti pada model 3-5-1):

```
P=[0.761538 0.656044 0.816484 0.491209 0.792308
   0.781319 0.607692 .......dst hingga 0.686813;
   0.776923 0.689011 0.849451 0.541758 0.812088
   0.803297 0.642857 .......dst hingga 0.717582;
   0.798901 0.715385 0.878022 0.585714 0.845055
   0.829670 0.673626 .......dst hingga 0.732967]
T=[0.761538 0.656044 0.816484 0.491209 0.792308
   0.781319 0.607692 .......dst hingga 0.686813]
```

→ Enter

Hasil nya akan berupa normalisasi yang sudah diubah menjadi 4 desimal dibelakang koma.

- Kemudian ketikkan kode program berikut (berdasarkan parameter tabel 3.1):

```
>> net=newff(minmax(P),[10,1],{'tansig','logsig'},'traincgp');
```

//Perintah ini untuk membentuk jaringan dengan 1 hiden layer sebanyak 10 neuron dan 1 *output*. Fungsi aktivasi yang digunakan adalah tansig (sigmoid bipolar) dan logsig (sigmoid biner) serta Fungsi pelatihan traincgp (Polak ribiere)

```
>> net.LW{1,1};
```

//Perintah ini untuk melihat nilai bobot awal pada lapisan masukan dan lapisan tersembunyi (bilangan diambil secara acak dari komputer)

```
>> net.b{1};
```

//Perintah ini untuk melihat nilai bias pada *hidden* layer/lapisan tersembunyi (bilangan diambil secara acak dari komputer)

```
>> net.LW{2,1};
```

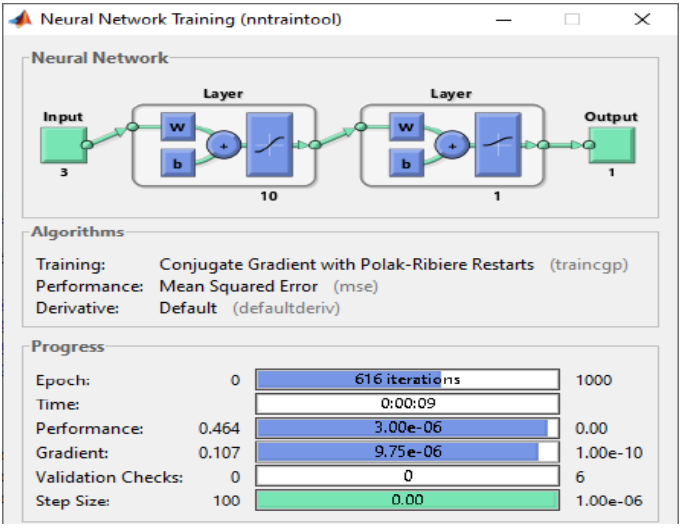
//Perintah ini untuk melihat nilai bobot pada *hidden* layer/lapisan tersembunyi dan *output* layer/lapisan keluaran (bilangan diambil secara acak dari komputer)

```
>> net.b{2};
```

//Perintah ini untuk melihat nilai bias pada *output* layer/lapisan keluaran (bilangan diambil secara acak dari komputer)

```
>> net.trainParam.epochs=1000;
>> net.trainParam.show = 25;
>> net.trainParam.showCommandLine = false;
>> net.trainParam.showWindow = true;
>> net.trainParam.goal = 0;
>> net.trainParam.time = inf;
>> net.trainParam. min_grad= 1e-10;
>> net.trainParam.max_fail = 6;
>> net.trainParam.searchFcn = 'srchcha';
>> net=train(net,P,T)
```

Akan muncul gambar hasil pelatihan dengan algoritma *Conjugate gradient Polak-Ribiere* seperti berikut:



Gambar 3.6: Hasil Pelatihan dengan Model Arsitektur 3-10-1

Untuk melihat tampilan *Plots*: *Performance*, *Training State* dan *regression* dapat dilakukan dengan menekan button pada gambar yang ada didalam frame *Plots*.

- Selanjutnya ketikkan kode berikut:

```
>> [a,Pf,Af,e,Perf]=sim(net,P,[],[],T)
```

//Kode ini untuk mendapatkan nilai *a* (*Output*) dan nilai *e* (*error*) data *training*.

Kemudian pada *Ms. Excel* buat tabel seperti berikut (nilai *a* (*output*) dimasukkan pada tabel):

Tabel 3.10: Hasil Data *Training* dengan Model 3-10-1

Data	Target	*Output*	*Error*	SSE
1	0,79890	0,79720	0,00170	0,0000028937
2	0,71538	0,71540	-0,00002	0,0000000002
3	0,87802	0,87850	-0,00048	0,0000002285
4	0,58571	0,58710	-0,00139	0,0000019202
5	0,84505	0,84540	-0,00035	0,0000001191
6	0,82967	0,83010	-0,00043	0,0000001846
7	0,67363	0,67620	-0,00257	0,0000066236
8	0,71758	0,71880	-0,00122	0,0000014825
9	0,80989	0,80940	0,00049	0,0000002402
10	0,55934	0,55560	0,00374	0,0000139925
11	0,57253	0,57040	0,00213	0,0000045261
12	0,82967	0,83050	-0,00083	0,0000006884
13	0,89341	0,89240	0,00101	0,0000010132
14	0,50879	0,51010	-0,00131	0,0000017129
15	0,60110	0,60300	-0,00190	0,0000036142
16	0,83187	0,83280	-0,00093	0,0000008684
17	0,87363	0,87450	-0,00087	0,0000007632
18	0,90000	0,89680	0,00320	0,0000102400
19	0,83626	0,83760	-0,00134	0,0000017856
20	0,60989	0,60790	0,00199	0,0000039605
21	0,85165	0,85350	-0,00185	0,0000034286
22	0,52637	0,52580	0,00057	0,0000003290
23	0,84066	0,83750	0,00316	0,0000099814
24	0,70440	0,70180	0,00260	0,0000067372
25	0,70879	0,70780	0,00099	0,0000009825
26	0,61868	0,61990	-0,00122	0,0000014852
27	0,75934	0,75730	0,00204	0,0000041643
28	0,49341	0,49570	-0,00229	0,0000052597

Data	Target	*Output*	*Error*	SSE
29	0,18571	0,18570	0,00001	0,0000000002
30	0,85385	0,85470	-0,00085	0,0000007291
31	0,52637	0,52550	0,00087	0,0000007632
32	0,86044	0,86140	-0,00096	0,0000009224
33	0,86703	0,86650	0,00053	0,0000002841
34	0,69560	0,69980	-0,00420	0,0000176031
35	0,86484	0,86580	-0,00096	0,0000009309
36	0,69341	0,69420	-0,00079	0,0000006295
37	0,70220	0,70140	0,00080	0,0000006365
38	0,73297	0,73170	0,00127	0,0000016054
			Jlh SSE	0,0001133300
			MSE	**0,0000029824**

Keterangan :
(Sama seperti keterangan pada tabel 3.4 model 3-5-1)

- Kemudian masukkan normalisasi data *Testing* berdasarkan tabel 3.7 (sama seperti pada model 3-5-1):

```
PP=[0.758564 0.670166 0.831492 0.522099 0.793923
    0.785083 0.623757 ........dst hingga 0.698895;
    0.780663 0.696685 0.860221 0.566298 0.827072
    0.811602 0.654696 .......dst hingga 0.714365;
    0.798343 0.720994 0.875691 0.612707 0.842541
    0.831492 0.685635 .......dst hingga 0.732044]
TT=[0.798343 0.720994 0.875691 0.612707 0.842541
    0.831492 0.685635 .......dst hingga 0.732044]
→ Enter
```

Hasil nya akan berupa normalisasi yang sudah diubah menjadi 4 desimal dibelakang koma.

- Kemudian ketikkan kode program berikut:

```
>> [a,Pf,Af,e,Perf]=sim(net,PP,[],[],TT)
```

//Kode ini untuk mendapatkan nilai *a* (*Output*) dan nilai *e* (*error*) data *training*.

Kemudian pada *Ms. Excel* buat tabel seperti berikut:

Tabel 3.11: Hasil Akurasi Data *Testing* dengan Model 3-10-1

Data	Target	*Output*	*Error*	SSE	Akurasi
1	0,79834	0,79720	0,00114	0,0000013054	1
2	0,72099	0,72020	0,00079	0,0000006312	1
3	0,87569	0,87660	-0,00091	0,0000008270	1
4	0,61271	0,61540	-0,00269	0,0000072513	1
5	0,84254	0,84490	-0,00236	0,0000055628	1

Data	Target	*Output*	*Error*	SSE	Akurasi
6	0,83149	0,83270	-0,00121	0,0000014600	1
7	0,68564	0,68740	-0,00176	0,0000031140	1
8	0,71657	0,71540	0,00117	0,0000013797	1
9	0,80718	0,80730	-0,00012	0,0000000138	1
10	0,55525	0,55510	0,00015	0,0000000221	1
11	0,57072	0,56870	0,00202	0,0000040733	1
12	0,83370	0,83490	-0,00120	0,0000014360	1
13	0,89558	0,89300	0,00258	0,0000066570	1
14	0,51989	0,52010	-0,00021	0,0000000443	1
15	0,60387	0,60500	-0,00113	0,0000012828	1
16	0,83149	0,83280	-0,00131	0,0000017116	1
17	0,87348	0,87460	-0,00112	0,0000012529	1
18	0,90000	0,89600	0,00400	0,0000160000	0
19	0,83812	0,83970	-0,00158	0,0000024915	1
20	0,63702	0,64130	-0,00428	0,0000183477	1
21	0,85359	0,85470	-0,00111	0,0000012295	1
22	0,52210	0,52040	0,00170	0,0000028881	1
23	0,85359	0,85590	-0,00231	0,0000053307	1
24	0,69448	0,69210	0,00238	0,0000056413	1
25	0,70994	0,70870	0,00124	0,0000015494	1
26	0,62597	0,62780	-0,00183	0,0000033604	1
27	0,76519	0,76330	0,00189	0,0000035849	1
28	0,49337	0,49520	-0,00183	0,0000033483	1
29	0,21050	0,24390	-0,03340	0,0011157445	1
30	0,85359	0,85580	-0,00221	0,0000048790	1
31	0,52431	0,52280	0,00151	0,0000022783	1
32	0,86022	0,86230	-0,00208	0,0000043223	1
33	0,87127	0,87340	-0,00213	0,0000045338	1
34	0,69448	0,69280	0,00168	0,0000028061	1
35	0,86022	0,86160	-0,00138	0,0000019017	1
36	0,69669	0,69820	-0,00151	0,0000022950	1
37	0,70110	0,70060	0,00050	0,0000002550	1
38	0,73204	0,72950	0,00254	0,0000064729	1
			Jlh SSE	0,0012472855	**97%**
			MSE	**0,0000328233**	

Keterangan :
Data = 38 Negara
Target = Diambil dari normalisasi target data *Testing*
Output = Diambil dari nilai *a* data *Testing* hasil Matlab

Error = Diperoleh dari *Target – Output* atau dari nilai *e* data *Testing* hasil dari Matlab
SSE = *Error* ^ 2
Jlh SSE = Total nilai SSE
MSE = Jlh SSE / 38 (banyaknya data)
Akurasi = Bernilai 1 (benar) apabila nilai *error* ≤ 0,003 (If *Error* ≤ 0,003;1;0)
0,003 = Minimum *error* yang digunakan (0,003-0,001)
97% = Diperoleh dari *Jumlah benar / Jumlah data * 100* (37/38*100 = 97,37% = **97%** (dibulatkan))

c. Pelatihan dan Pengujian dengan Model Arsitektur 3-5-10-1

- Masukkan normalisasi data *training* berdasarkan tabel 3.6 (Hasil Normalisasi Data *Training*) secara horizontal (mendatar) seperti berikut:

```
P=[0.761538 0.656044 0.816484 0.491209 0.792308
   0.781319 0.607692 .......dst hingga 0.686813;
   0.776923 0.689011 0.849451 0.541758 0.812088
   0.803297 0.642857 .......dst hingga 0.717582;
   0.798901 0.715385 0.878022 0.585714 0.845055
   0.829670 0.673626 .......dst hingga 0.732967]
T=[0.761538 0.656044 0.816484 0.491209 0.792308
   0.781319 0.607692 .......dst hingga 0.686813];
```

→ Enter

Hasil nya akan berupa normalisasi yang sudah diubah menjadi 4 desimal dibelakang koma.

- Kemudian ketikkan kode program berikut (berdasarkan parameter tabel 3.1):

>> net=newff(minmax(P),[5,10,1],{'tansig','logsig','tansig'},'traincgp');
//Perintah ini untuk membentuk jaringan dengan 2 hiden layer sebanyak 5 neuron dan 10 neuron serta 1 *output*. Fungsi aktivasi yang digunakan adalah tansig (sigmoid bipolar) dan logsig (sigmoid biner) serta Fungsi pelatihan traincgp (Polak ribiere)

>> net.LW{1,1};
//Perintah ini untuk melihat nilai bobot awal pada lapisan masukan dan lapisan tersembunyi (bilangan diambil secara acak dari komputer)

>> net.b{1};
//Perintah ini untuk melihat nilai bias pada *hidden* layer/lapisan tersembunyi (bilangan diambil secara acak dari komputer)

>> net.LW{2,1};
//Perintah ini untuk melihat nilai bobot pada *hidden* layer/lapisan tersembunyi dan *output* layer/lapisan keluaran (bilangan diambil secara acak dari komputer)

>> net.b{2};

//Perintah ini untuk melihat nilai bias pada *output* layer/lapisan keluaran (bilangan diambil secara acak dari komputer)

>> *net.LW{3,1};*

//Perintah ini untuk melihat nilai bobot pada *hidden* layer/lapisan tersembunyi dan *output* layer/lapisan keluaran (bilangan diambil secara acak dari komputer)

>> *net.b{3};*

//Perintah ini untuk melihat nilai bias pada *output* layer/lapisan keluaran (bilangan diambil secara acak dari komputer)

>> *net.trainParam.epochs=1000;*

//Penjelasan dapat dilihat pada tabel 3.1

>> *net.trainParam.show = 25;*

//Penjelasan dapat dilihat pada tabel 3.1

>> *net.trainParam.showCommandLine = false;*

//Penjelasan dapat dilihat pada tabel 3.1

>> *net.trainParam.showWindow = true;*

//Penjelasan dapat dilihat pada tabel 3.1

>> *net.trainParam.goal = 0;*

//Penjelasan dapat dilihat pada tabel 3.1

>> *net.trainParam.time = inf;*

//Penjelasan dapat dilihat pada tabel 3.1

>> *net.trainParam. min_grad= 1e-10;*

//Penjelasan dapat dilihat pada tabel 3.1

>> *net.trainParam.max_fail = 6;*

//Penjelasan dapat dilihat pada tabel 3.1

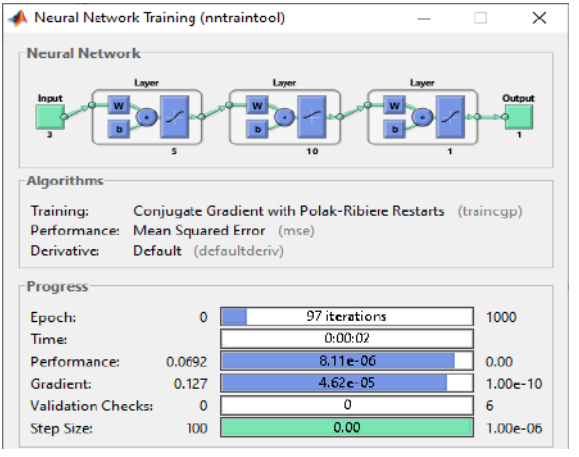
>> *net.trainParam.searchFcn = 'srchcha';*

//Penjelasan dapat dilihat pada tabel 3.1

>> *net=train(net,P,T)*

//Penjelasan dapat dilihat pada tabel 3.1

Akan muncul gambar hasil pelatihan dengan algoritma *Conjugate gradient Polak-Ribiere* seperti berikut:



Gambar 3.7: Hasil Pelatihan dengan Model Arsitektur 3-5-10-1

Untuk melihat tampilan *Plots*: *Performance*, *Training State* dan *regression* dapat dilakukan dengan menekan button pada gambar yang ada di dalam frame *Plots*.

- Selanjutnya ketikkan kode berikut:

```
>> [a,Pf,Af,e,Perf]=sim(net,P,[],[],T)
```

//Kode ini untuk mendapatkan nilai *a* (*Output*) dan nilai *e* (*error*) data *training*.

Kemudian pada *Ms. Excel* buat tabel seperti berikut:

Tabel 3.12: Hasil Data *Training* dengan Model 3-5-10-1

Data	Target	*Output*	*Error*	SSE
1	0,79890	0,79890	0,00000	0,0000000000
2	0,71538	0,71720	-0,00182	0,0000032956
3	0,87802	0,87630	0,00172	0,0000029652
4	0,58571	0,59000	-0,00429	0,0000183673
5	0,84505	0,84890	-0,00385	0,0000147844
6	0,82967	0,83150	-0,00183	0,0000033477
7	0,67363	0,67500	-0,00137	0,0000018868
8	0,71758	0,71790	-0,00032	0,0000001009
9	0,80989	0,80900	0,00089	0,0000007923
10	0,55934	0,56050	-0,00116	0,0000013441
11	0,57253	0,57160	0,00093	0,0000008602
12	0,82967	0,82970	-0,00003	0,0000000009
13	0,89341	0,88640	0,00701	0,0000490924
14	0,50879	0,50640	0,00239	0,0000057179
15	0,60110	0,60140	-0,00030	0,0000000907
16	0,83187	0,83200	-0,00013	0,0000000174
17	0,87363	0,87250	0,00113	0,0000012687
18	0,90000	0,89420	0,00580	0,0000336400
19	0,83626	0,83610	0,00016	0,0000000268
20	0,60989	0,60950	0,00039	0,0000001522
21	0,85165	0,85310	-0,00145	0,0000021073
22	0,52637	0,53010	-0,00373	0,0000138859
23	0,84066	0,84090	-0,00024	0,0000000579
24	0,70440	0,70170	0,00270	0,0000072663
25	0,70879	0,70760	0,00119	0,0000014190
26	0,61868	0,61630	0,00238	0,0000056707
27	0,75934	0,75970	-0,00036	0,0000001291
28	0,49341	0,48850	0,00491	0,0000240747
29	0,18571	0,18570	0,00001	0,0000000002

Data	Target	*Output*	*Error*	SSE
30	0,85385	0,85710	-0,00325	0,0000105875
31	0,52637	0,52950	-0,00313	0,0000097742
32	0,86044	0,86230	-0,00186	0,0000034612
33	0,86703	0,86730	-0,00027	0,0000000713
34	0,69560	0,68870	0,00690	0,0000476707
35	0,86484	0,86830	-0,00346	0,0000120051
36	0,69341	0,69870	-0,00529	0,0000280202
37	0,70220	0,70230	-0,00010	0,0000000104
38	0,73297	0,73110	0,00187	0,0000034858
			Jlh SSE	0,0003074489
			MSE	**0,0000080908**

Keterangan :
Data = 38 Negara
Target = Diambil dari normalisasi target data *training*
Output = Diambil dari nilai *a* hasil dari Matlab
Error = Diperoleh dari *Target – Output* atau dari nilai *e* hasil dari Matlab
SSE = *Error* ^ 2
Jlh SSE = Total nilai SSE
MSE = Jlh SSE / 38 (banyaknya data)

- Kemudian masukkan normalisasi data *Testing* berdasarkan tabel 3.7 (Hasil Normalisasi Data *Testing*) secara horizontal (mendatar) seperti berikut:

```
PP=[0.758564 0.670166 0.831492 0.522099 0.793923
    0.785083 0.623757 ........dst hingga 0.698895;
    0.780663 0.696685 0.860221 0.566298 0.827072
    0.811602 0.654696 .......dst hingga 0.714365;
    0.798343 0.720994 0.875691 0.612707 0.842541
    0.831492 0.685635 .......dst hingga 0.732044]
TT=[0.798343 0.720994 0.875691 0.612707 0.842541
    0.831492 0.685635 .......dst hingga 0.732044]
```

→ Enter

Hasil nya akan berupa normalisasi yang sudah diubah menjadi 4 desimal dibelakang koma.
Keterangan:
PP = *Input* data *Testing*
TT = Target data *Testing*

- Kemudian ketikkan kode program berikut:

```
>> [a,Pf,Af,e,Perf]=sim(net,PP,[],[],TT)
```

//Kode ini untuk mendapatkan nilai *a* (*Output*) dan nilai *e* (*error*) data *training*.

Kemudian pada *Ms. Excel* buat tabel seperti berikut:

Tabel 3.13: Hasil Akurasi Data *Testing* dengan Model 3-5-10-1

Data	Target	*Output*	*Error*	SSE	Akurasi
1	0,79834	0,79630	0,00204	0,0000041720	1
2	0,72099	0,72210	-0,00111	0,0000012222	1
3	0,87569	0,87530	0,00039	0,0000001526	1
4	0,61271	0,61580	-0,00309	0,0000095655	1
5	0,84254	0,84130	0,00124	0,0000015412	1
6	0,83149	0,83170	-0,00021	0,0000000434	1
7	0,68564	0,68790	-0,00226	0,0000051286	1
8	0,71657	0,71590	0,00067	0,0000004551	1
9	0,80718	0,80410	0,00308	0,0000095007	0
10	0,55525	0,55900	-0,00375	0,0000140729	1
11	0,57072	0,57020	0,00052	0,0000002686	1
12	0,83370	0,83420	-0,00050	0,0000002483	1
13	0,89558	0,89220	0,00338	0,0000114251	0
14	0,51989	0,52250	-0,00261	0,0000068147	1
15	0,60387	0,60240	0,00147	0,0000021533	1
16	0,83149	0,83150	-0,00001	0,0000000001	1
17	0,87348	0,87340	0,00008	0,0000000065	1
18	0,90000	0,89630	0,00370	0,0000136900	0
19	0,83812	0,83880	-0,00068	0,0000004603	1
20	0,63702	0,63820	-0,00118	0,0000014005	1
21	0,85359	0,85800	-0,00441	0,0000194379	1
22	0,52210	0,52550	-0,00340	0,0000115638	1
23	0,85359	0,84820	0,00539	0,0000290646	0
24	0,69448	0,69190	0,00258	0,0000066313	1
25	0,70994	0,71000	-0,00006	0,0000000031	1
26	0,62597	0,62500	0,00097	0,0000009348	1
27	0,76519	0,76400	0,00119	0,0000014241	1
28	0,49337	0,49300	0,00037	0,0000001370	1
29	0,21050	0,21000	0,00050	0,0000002472	1
30	0,85359	0,85360	-0,00001	0,0000000001	1
31	0,52431	0,52770	-0,00339	0,0000114962	1
32	0,86022	0,86030	-0,00008	0,0000000062	1
33	0,87127	0,86710	0,00417	0,0000173949	0
34	0,69448	0,69430	0,00018	0,0000000307	1

Data	Target	*Output*	*Error*	SSE	Akurasi
35	0,86022	0,86310	-0,00288	0,0000082887	1
36	0,69669	0,69680	-0,00011	0,0000000132	1
37	0,70110	0,70060	0,00050	0,0000002550	1
38	0,73204	0,73080	0,00124	0,0000015480	1
			Jlh SSE	0,0001907983	**87%**
			MSE	**0,0000050210**	

Keterangan :
Data = 38 Negara
Target = Diambil dari normalisasi target data *Testing*
Output = Diambil dari nilai *a* data *Testing* hasil Matlab
Error = Diperoleh dari *Target – Output* atau dari nilai *e* data *Testing* hasil dari Matlab
SSE = *Error* ^ 2
Jlh SSE = Total nilai SSE
MSE = Jlh SSE / 38 (banyaknya data)
Akurasi = Bernilai 1 (benar) apabila nilai *error* ≤ 0,003 (If *Error* ≤ 0,003;1;0)
0,003 = Minimum *error* yang digunakan (0,003-0,001)
87% = Diperoleh dari *Jumlah benar / Jumlah data* * *100* (33/38*100 = 86,84% = **87%** (dibulatkan))

5. Menentukan Model Arsitektur Terbaik

Berdasarkan pelatihan dan pengujian data dengan menggunakan model arsitektur 3-5-1, 3-10-1 dan 3-5-10-1 maka dapat disimpulkan model arsitektur terbaik yang disajikan pada tabel berikut:

Tabel 3.14: Penentuan Model Arsitektur Terbaik

***Conjugate gradient Polak-Ribiere* (traincgp)**				
Arsitektur	**Epoch**	**Waktu**	**MSE**	**Accuracy**
3-5-1	310	00.05	0,0000047137	95%
3-10-1	616	00.09	0,0000328233	97%
3-5-10-1	97	00.02	0,0000050210	87%

Berdasarkan tabel 3.14, dapat diambil kesimpulan bahwa model arsitektur terbaik adalah model 3-5-1 dengan tingkat akurasi 95%. Meskipun Akurasi nya 95%, dan lebih rendah dari dari model 3-10-1 yang tingkat akurasi nya 97%, tetapi model arsitektur 3-5-1 lebih kecil Mean Square *Error* (MSE) nya dibandingkan 2 model arsitektur yang

lain. Sebagaimana perlu diketahui bahwa semakin kecil MSE nya, maka semakin baik hasil untuk melakukan prediksi nantinya.

6. Melakukan Prediksi Data

Prediksi data Angka Harapan Hidup (AHH) penduduk dunia akan dilakukan menggunakan model arsitektur 3-5-1. Karena model 3-5-1 merupakan model arsitektur yang terbaik.

a. Prediksi Data AHH Penduduk Dunia (Tahun 2015-2020)

- Buka Aplikasi Matlab. Masukkan normalisasi data untuk prediksi berdasarkan tabel 3.7 ke Matlab seperti berikut:

```
P=[0.758564 0.670166 0.831492 0.522099 0.793923
   0.785083 0.623757 ........dst hingga 0.698895;
   0.780663 0.696685 0.860221 0.566298 0.827072
   0.811602 0.654696 .......dst hingga 0.714365;
   0.798343 0.720994 0.875691 0.612707 0.842541
   0.831492 0.685635 .......dst hingga 0.732044]
T=[0.798343 0.720994 0.875691 0.612707 0.842541
   0.831492 0.685635 .......dst hingga 0.732044]
```

→ Enter

Hasil nya akan berupa normalisasi yang sudah diubah menjadi 4 desimal dibelakang koma.

Keterangan:

P = *Input* data untuk prediksi

T = Target data untuk prediksi

- Kemudian ketikkan kode program berikut (berdasarkan model arsitektur terbaik 3-5-1):

```
>> net=newff(minmax(P),[5,1],{'tansig','logsig'},'traincgp');
>> net.LW{1,1};
>> net.b{1};
>> net.LW{2,1};
>> net.b{2};
>> net.trainParam.epochs=1000;
>> net.trainParam.show = 25;
>> net.trainParam.showCommandLine = false;
>> net.trainParam.showWindow = true;
>> net.trainParam.goal = 0;
>> net.trainParam.time = inf;
>> net.trainParam. min_grad= 1e-10;
>> net.trainParam.max_fail = 6;
>> net.trainParam.searchFcn = 'srchcha';
>> net=train(net,P,T)
```

- Selanjutnya ketikkan kode berikut pada Matlab:

```
>> [a,Pf,Af,e,Perf]=sim(net,P,[],[],T)
```

Akan muncul nilai *a* (*output*) yang nantinya kan dimasukkan kedalam tabel perhitungan. Secara keseluruhan perhitungan untuk mendapatkan hasil prediksi Angka Harapan Hidup (AHH) tahun 2015-2020 dapat dilihat pada tabel berikut ini:

Tabel 3.15: Prediksi AHH Tahun 2015-2020

No	Negara	Data Real	Target	Target Prediksi	Prediksi
1	United States	78,9	0,79834	0,79800	79,5
2	Saudi Arabia	75,4	0,72099	0,72110	76,5
3	Australia	82,4	0,87569	0,87550	82,5
4	Bangladesh	70,5	0,61271	0,60750	72,1
5	Netherlands	80,9	0,84254	0,84170	81,2
6	Belgium	80,4	0,83149	0,83240	80,9
7	Brazil	73,8	0,68564	0,68710	75,2
8	China	75,2	0,71657	0,71600	76,3
9	Denmark	79,3	0,80718	0,80580	79,8
10	Russian	67,9	0,55525	0,55600	70,1
11	The Philippines	68,6	0,57072	0,57070	70,7
12	Finland	80,5	0,83370	0,83510	81,0
13	Hong Kong	83,3	0,89558	0,89500	83,3
14	India	66,3	0,51989	0,51760	68,6
15	Indonesia	70,1	0,60387	0,60420	72,0
16	English	80,4	0,83149	0,83200	80,8
17	Italy	82,3	0,87348	0,87370	82,5
18	Japan	83,5	0,90000	0,89940	83,5
19	German	80,7	0,83812	0,83930	81,1
20	Cambodia	71,6	0,63702	0,64000	73,4
21	Canada	81,4	0,85359	0,85710	81,8
22	Kazakhstan	66,4	0,52210	0,52250	68,8
23	South Korea	81,4	0,85359	0,85050	81,6
24	Kuwait	74,2	0,69448	0,69420	75,5
25	Malaysia	74,9	0,70994	0,70990	76,1
26	Mexico	71,1	0,62597	0,62630	72,8
27	Egypt	77,4	0,76519	0,76450	78,2
28	Myanmar	65,1	0,49337	0,49590	67,7
29	Nigeria	52,3	0,21050	0,21040	56,6
30	Norway	81,4	0,85359	0,85370	81,7
31	Pakistan	66,5	0,52431	0,52390	68,8
32	France	81,7	0,86022	0,86030	82,0
33	Singapore	82,2	0,87127	0,86880	82,3
34	Sri Lanka	74,2	0,69448	0,69490	75,5

No	Negara	Data Real	Target	Target Prediksi	Prediksi
35	Sweden	81,7	0,86022	0,86170	82,0
36	Thailand	74,3	0,69669	0,69800	75,6
37	Venezuela	74,5	0,70110	0,70120	75,7
38	Vietnamese	75,9	0,73204	0,73100	76,9
Nilai Max		83,5			
Nilai Min		52,3			
Nilai Max – Nilai Min		31,2			

Keterangan :

Data Real = Diperoleh dari data terakhir AHH tahun 2010-2015 (berdasarkan Tabel 3.3)

Target = Normalisasi Data Real

Nilai Max = Nilai terbesar dari data real

Nilai Min = Nilai terendah dari data real

Target Prediksi = Nilai *a* (*output*) yang di peroleh dari Matlab

Prediksi = (((Target Prediksi-0,1)*('Hasil max-min'))/0,8+min)

b. Perbandingan Data Awal dengan Hasil Prediksi

Pada Tabel berikut akan disajikan data awal sebelum di lakukan prediksi dengan data hasil prediksi.

Tabel 3.16: Perbandingan Data Awal dengan Hasil Prediksi

No	Negara	Umur/Tahun				Prediksi
		1995-2000	2000-2005	2005-2010	2010-2015	2015-2020
1	United States	76,4	77,1	78,1	78,9	79,5
2	Saudi Arabia	71,6	73,1	74,3	75,4	76,5
3	Australia	78,9	80,4	81,7	82,4	82,5
4	Bangladesh	64,1	66,4	68,4	70,5	72,1
5	Netherlands	77,8	78,7	80,2	80,9	81,2
6	Belgium	77,3	78,3	79,5	80,4	80,9
7	Brazil	69,4	71,0	72,4	73,8	75,2
8	China	70,9	73,4	74,4	75,2	76,3
9	Denmark	76,0	77,3	78,6	79,3	79,8
10	Russian Federation	65,7	65,0	67,2	67,9	70,1
11	The Philippines	66,4	67,1	67,8	68,6	70,7
12	Finland	77,0	78,3	79,5	80,5	81,0
13	Hong Kong SAR	79,4	81,3	82,4	83,3	83,3
14	India	61,2	63,1	64,9	66,3	68,6

No	Negara	Umur/Tahun				Prediksi
		1995-2000	2000-2005	2005-2010	2010-2015	2015-2020
15	Indonesia	66,0	67,8	69,1	70,1	72,0
16	English	77,1	78,4	79,6	80,4	80,8
17	Italy	78,7	80,2	81,5	82,3	82,5
18	Japan	80,5	81,8	82,7	83,5	83,5
19	German	77,2	78,6	79,8	80,7	81,1
20	Cambodia	59,8	64,5	69,5	71,6	73,4
21	Canada	78,5	79,7	80,5	81,4	81,8
22	Kazakhstan	63,0	64,6	65,7	66,4	68,8
23	South Korea	74,9	77,4	80,0	81,4	81,6
24	Kuwait	72,9	73,4	73,8	74,2	75,5
25	Malaysia	72,3	73,3	74,0	74,9	76,1
26	Mexico	68,0	69,0	69,9	71,1	72,8
27	Egypt	73,7	75,0	76,3	77,4	78,2
28	Myanmar	61,3	62,8	64,2	65,1	67,7
29	Nigeria	46,3	47,3	50,2	52,3	56,6
30	Norway	78,2	79,2	80,6	81,4	81,7
31	Pakistan	63,1	64,5	65,7	66,5	68,8
32	France	78,3	79,5	80,9	81,7	82,0
33	Singapore	77,7	79,2	81,2	82,2	82,3
34	Sri Lanka	69,1	73,2	73,4	74,2	75,5
35	Sweden	79,2	80,1	81,1	81,7	82,0
36	Thailand	70,6	71,5	73,3	74,3	75,6
37	Venezuela	72,1	72,8	73,7	74,5	75,7
38	Vietnamese	73,0	74,4	75,1	75,9	76,9

Catatan :

- Untuk prediksi tahun 2020-2025 bisa menggunakan *Input* data untuk prediksi (P): tahun 2005-2010, 2010-2015 dan 2015-2020 dengan Target data untuk prediksi (T): tahun 2015-2020. Data real adalah data prediksi tahun 2015-2020. Begitu untuk seterusnya.

Bab 4

Algoritma Habb dan Penerapan

4.1 Pengenalan

Perkembangan *Computer Vision* saat ini telah di implementaswikan pada berbagai bidang (Yana Simamora, Hasan and Rizka, 2016) di mana pada perkembangan *Computer Vision* ini berhubungan erat dengan penerapan aturan atau metode. Di mana *Jaringan Saraf Tiruan* banyak dikembangkan baik itu untuk mengenalan pola, peramalan cuaca peramalam perdagangan. *Jaringan Saraf Tiruan* banyak turunan metode – metodenya diantaranya adalah Jaringan Habb, Adeline, Paerceptron juga Backpropagation(Tambunan, Yudi and Fauzi, 2019). Yang mana metode metode tersebut dapat digunakan untuk mengenali pola.

Di dalam pengenalan pola bentuk banyak metode metode yang digunakan misalnya dengan menggunakan metode perceptron(Tambunan, 2019) di dalam penelitian tersebut mengenali sebuah bentuk. Kemudian pada penelitian (Muliono and Hakim Lubis, 2018) meneliti pengenalan bentuk atau pola dari sebuah huruf dengan menggunakan jaringan Habb. Dan pada tulisan ini juga memaparkan dan memberikan pemecahan kasus kasus kecil dengan menggunakan algoritma jaringan *Habb*.

4.2 Algoritma Habb

Jaringan *Habb* atau *Hebbian* ini adalah salah satu turudan dari cabang *Artifisial Inteligen* (Kecerdasan Buatan) (G, Widagda and Suarbawa, 2018) yang mana, jaringan *Habb* ini di kembangkan dengan memodifikasi bobot dan biasnya. Jaringan *Habb* ini tidak memakai *hiden layer* yang mana *hidel layer*

banyak ditemukan pada algoritma algoritma turunan dari *Jaringan Saraf Tiruan*.

4.2.1 Langkah Langkah Penyelesaian

Untuk menyelesaikan pemasalahan dengan menggunakan algoritma Habb adalah dalam setiap putaran ulangan, bobot dan bias diubah berdasarkan perkalian semua masukan x_i terhubung langsung dengan unit keluaran *y*, maka perubahan nilai bobot dilakukan berdasarkan persamaan

$$w_i(baru) = w_i(lama) + x_i y \quad (4.1)$$

Algoritma pelatihan *Habb* dengan input *s* dan unit target *t* adalah sebagai berikut:

1. Inisialisasi semua bobot = w_i = 0, (i=1,2,3,..,..,..,n)
2. Untuk semua vertor input *s* dan unit target *t*, lakukan :
   a. Set aktivasi untuk $x_i = s_i$ (i=1,2,3,..,..,..,n)
   b. Set aktivasi unit keluaran $y = t$
   c. Perbaiki bobot dengan memenuhi persamaan

$$w_i(baru) = w_i(lama) + \Delta w \ (i = 1,2,3,..,..,..,n)$$

$$\Delta w = x_i y \quad (4.2)$$

   d. Perbaiki bias menurut persamaan *b* (baru) = *b* (lama) = *t*

Perhatikan bahwa perbaikan bias diperlakukan sama seperti bobot.

4.2.2 Contoh Penyelesaian

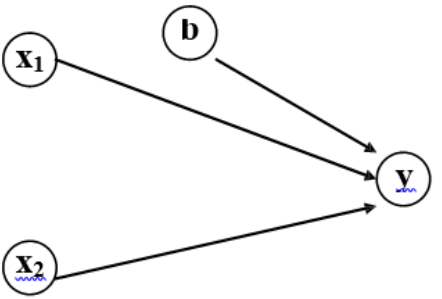
Dari langkah – langkah penyelesaian di atas kita menerapkan pada contoh :

Contoh 1 :

Kenalilah pola fungsi "DAN" dengan metode *Habb*, di mana inputan binar dan target binar.

Tabel 4.1: Fungsi AND

Masukan			**Target**
x_1	x_2	b	T
1	1	1	1
1	0	1	0
0	1	1	0
0	0	1	0



Gambar 4.1: Hubungan masukan x_1, x_2, b dan output y.

Pertama sekali semua bobot dan bias diberikan nilai = 0. Untuk data masukan dan target, perubahan bobot dihitung dari perkalian data masukan dan targetnya.

$\Delta w_1 = x_1 t \qquad \Delta w_2 = x_2 t \qquad \Delta b = 1.t$

Bobot w_1 baru = bobot w_1 lama + Δw_1 (i=1, 2, 3,…n)

Sehingga hasil perhitungan bobot menggunakan persamaan tersebut tampak pada tabel di bawah ini :

Tabel 4.2: Hasil perhitungan

Input			**Target**		**Perubahan bobot-bias**			**Bobot dan bias**		
x_1	x_2	b	T	y	Δw_1	Δw_2	Δb	w_1	w_2	B
								0	0	0
1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	0	1	1	1
0	1	1	0	0	0	0	0	1	1	1
0	0	1	0	0	0	0	0	1	1	1

Dari hasil perhitungan yang tampak pada Tabel 4.2, bobot berubah pada pasangan data pertama saja. Pada data kedua, ketiga dan keempat tidak ada perubahan bobot karena target = 0. Sehingga perubahan bobot (hasil kali dari nilai masukan dan target) juga bernilai 0.

Merujuk pada tabel di atas didapatkan nilai bobot $w_1 = 1$, $w_2 = 1$ dan $b = 1$.

$$Net = \sum_{i=1}^{2} w_i x_i + b = 1.x_1 + 1.x_2 + 1 = x_1 + x_2 + 1$$

Dengan menggunakan formula di atas kita menguji bobot tersebut pada seluruh data masukan sehingga akan diperoleh data yang tampak pada Tabel 4.3 di bawah ini :

Tabel 4.3: Hasil pengujian bobot baru

Input		Target	$net = \sum_{i=1}^{2} w_i x_i + b$	$y = f(net)\begin{cases}1\ jika\ net \geq 0\\0\ jika\ net \leq 0\end{cases}$
x_1	x_2			
1	1	1	1.1+1.1+1=3	1
1	0	0	1.1+0.1+1=2	1
0	1	0	0.1+1.1+1=2	1
0	0	0	0.1+0.1+1=1	1

Pada Tabel 4.3 di atas *f(net)* tidak sama dengan target yang dimasukkan ke dalam fungsi "DAN" sehingga dapat disimpulkan bahwa *Habb* tidak dapat Mengenali pola tersebut dengam masukan biner dan keluaran biner.

Contoh2 :

Kenalilah pola fungsi "DAN" dengan metode *Habb*, di mana inputan binar dan target bipolar yang tampak pada Tabel 4.4.

Tabel 4.4: Fungsi AND dengan output bipolar

Masukan			Target
x_1	x_2	b	T
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

Dengan menggunakan aturan dan persamaan yang sama pada perhitungan sebelumnya yaitu pada masukan biner dan keluaran biner. Pada Tabel 4.5 di bawah ini didapat hasil perhitungan masukan biner dan keluaran bipolar.

Tabel 4.5: Hasil perhitungan bobot baru

Input			**Target**		**Perubahan bobot-bias**			**Bobot dan bias**		
x_1	x_2	b	T	y	Δw_1	Δw_2	Δb	w_1	w_2	b
								0	0	0
1	1	1	1	1	1	1	1	1	1	1
1	0	1	-1	-1	-1	0	-1	0	1	0
0	1	1	-1	-1	0	-1	-1	0	0	-1
0	0	1	-1	-1	0	0	-1	0	0	-2

Sehingga hasil bobot terahir dari hasil perhitungan dangan memasukkan persamaan didapatkan nilai bobot baru w_1=0, w_2=0 dan b=-2

$$Net = \sum_{i=1}^{2} w_i x_i + b \qquad (4.3)$$

Dengan kembali menggunakan formula di atas, kemudian kita menguji pada seluruh data masukan, sehingga diperoleh data yang tampak Tabel 4.6 sebagai berikut di bwah ini :

Tabel 4.6: Hasil pengujian bobot baru terhadap *y*

Input		Target	$net = \sum_{i=1}^{2} w_i x_i + b$	$y = f(net)\begin{cases}1\ jika\ net \geq 0\\0\ jika\ net \leq\ 0\end{cases}$
x_1	x_2			
1	1	1	1.0+1.0+(-2)=-2	-1
1	0	-1	1.0+0.0+(-2)=-2	-1
0	1	-1	0.0+1.0+(-2)=-2	-1
0	0	-1	0.0+0.0+(-2)=-2	-1

Pada tabel di atas *f(net)* tidak sama dengan target yang dimasukkan ke dalam fungsi "DAN", di mana hasil f(net) hanya dapat mengenali tiga dari target yang di tentukan yaitu kedua, ketiga dan keempat, sementara yang pertama tidak dapat dikenali. Sehingga dapat disimpulkan bahwa *Habb* tidak dapat Mengenali pola tersebut dengan aturan masukan binar dan keluaran bipolar.

Contoh 3 :

Kenalilah pola fungsi "DAN" dengan metode Habb, di mana inputan bipolar dan target bipolar yang tampak pada Tabel 4.7.

Tabel 4.7: Tabel masukan dan keluaran bipolar

Masukan			Target
x_1	x_2	b	T
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Dengan menggunakan aturan dan persamaan yang sama pada perhitungan sebelumnya yaitu pada masukan binar dan keluaran binar atau masukan biner

dan keluaran bipolar. Pada Tabel 4.8 di bawah ini, terdapat hasil perhitungan masukan bipolar dan keluaran bipolar.

Tabel 4.8: Hasil perhitungan masukan dan keluaran bipolar

Input			Target		Perubahan bobot-bias			Bobot dan bias		
x_1	x_2	b	T	y	Δw_1	Δw_2	Δb	w_1	w_2	b
								0	0	0
1	1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	-1	1	-1	0	2	0
-1	1	1	-1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	-1	1	1	-1	2	2	-2

Sehingga hasil bobot terahir dari hasil perhitungan dangan memasukkan persamaan didapatkan nilai bobot baru $w_1 = 2$, $w_2 = 2$ dan $b = -2$

$$Net = \sum_{i=1}^{2} w_i x_i + b$$

Dengan kembali menggunakan formula di atas, kemudian kita menguji pada seluruh data masukan sehingga diperoleh data yang tampak pada Tabel 4.9 sebagai berikut di bawah ini :

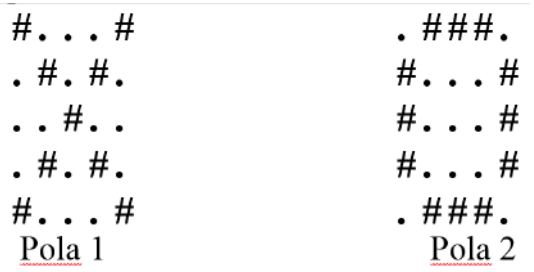
Tabel 4.9: Hasil perhitungan boot terhadap *y*

Input		Target	$net = \sum_{i=1}^{2} w_i x_i + b$	$y = f(net) \begin{cases} 1 \; jika \; net \geq 0 \\ 0 \; jika \; net \leq 0 \end{cases}$
x_1	x_2			
1	1	1	1.2+1.2+(-2)=2	1
1	-1	-1	1.2+(-1).2+(-2)=-2	-1
-1	1	-1	(-1).2+1.2+(-2)=-2	-1
-1	-1	-1	(-1).2+(-1).2+(-2)=-6	-1

Pada tabel di atas nilai *f(net)* sama dengan nilai target yang dimasukkan ke dalam fungsi "DAN". Sehingga dapat disimpulkan bahwa Habb dapat Mengenali pola tersebut dengan baik menggunakan aturan masukan bipolar dan keluaran bipolar.

Contoh 4:

Kenalilah pola gambar berikut dengan menggunakan algoritma *Habb* demana masukan dan keluaran menggunakan bipolar dan bobot awal diset pada nilai = 0



Gambar 4.2: Gambar pola 1 dan pola 2

Pada kasus kali ini kita telah masuk pada kasus yang di perhadapkan pada kehidupan sehari hari, yang akan bisa kita terapkan pada kasus kasus yang lain seperti pengenalan pola bentuk mobil, pola nomot polisi pada plat kendaraan dan lain sebagainya. Baiklah kembali ke topik pada kasus ini misal pada X kita beri nilai target 1 sedangkan bukan X kita beri target –1 itu adalah pada target. Sekarang kita masuk pada tahap *extraksi* pola gambar di mana setiap lambang "#" kita beri nilai 1 dan lambang "." kita beri nilai –1.

Nilai input untuk pola 1 dan pola 2 menjadi Tabel 4.10 sebagai berikut :

Tabel 4.10: Extaksi nilai dari pola 1 dan pola 2

Pola	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13
"X"	1	-1	-1	-1	1	-1	1	-1	1	-1	-1	-1	1
"O"	-1	1	1	1	-1	1	-1	-1	-1	1	1	-1	-1

X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	X24	X25	t
-1	-1	-1	1	-1	1	-1	1	-1	-1	-1	1	1
-1	1	1	-1	-1	-1	1	-1	1	1	1	-1	-1

Sedangkan perubahan bobot (Δwi) dan bias setelah diberikan input pola 1 dan 2 pada Tabel 4.11 sebagai berikut :

Tabel 4.11: Nilai perubahan bobot dan bias

Pola	Δw_1	Δw_2	Δw_3	Δw_4	Δw_5	Δw_6	Δw_7	Δw_8	Δw_9	Δw_{10}	Δw_{11}	Δw_{12}	Δw_{13}
"X"	1	-1	-1	-1	1	-1	1	-1	1	-1	-1	-1	1
"O"	-1	1	1	1	-1	1	-1	-1	-1	1	1	-1	-1

Δw_{14}	Δw_{15}	Δw_{16}	Δw_{17}	Δw_{18}	Δw_{19}	Δw_{20}	Δw_{21}	Δw_{22}	Δw_{23}	Δw_{24}	Δw_{25}	Δb
1	-1	-1	-1	1	-1	1	-1	1	-1	-1	-1	1
-1	1	1	1	-1	1	-1	-1	-1	1	1	-1	-1

Dan bobot akhir (wi) dan bias b dapat ditentukan dari penjumlahan kedua perubahan bobot di atas tampak pada tabel Tabel 4.12 di bawah ini :

Tabel 4.12: Perubahan bobot ahir dan bias

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}
2	-2	-2	-2	2	-2	2	0	2	-2	-2	0	2	0

w_{15}	w_{16}	w_{17}	w_{18}	w_{19}	w_{20}	w_{21}	w_{22}	w_{23}	w_{24}	w_{25}	b
-2	-2	2	0	2	-2	2	-2	-2	-2	2	0

Kemudian dengan menggunakan formula yang sama seperti perhitungan sebelumnya :

$$Net = \sum_{i=1}^{5} w_i x_i + b$$

Untuk pola 1

net = {1(2)+(-1)(-2)+(-1)(-2)+(-1)(-2)+(1)(2)}+{(-1)(-2)+1(2)+(-1)(0)+1(2)+(-1)(-2)}+{(-1)(-2)+(-1)(0)+1(2)+(-1)(0)+(-1)(-2)}+{(-1)(-2)+(1)(2)+(-1)(0)+1(2)+(-1)(-2)}+{1(2)+(-1)(-2)+(-1)(-2)+(-1)(-2)+1(2)} = 42 maka *f(net)* = 1

net = {(-1)(2)+1(-2)+1(-2)+1(-2)+(-1)(2)}+{1(-2)+(-1)(2)+(-1)(0)+(-1)(2)+1(-2)}+{1(-2)+(-1)(0)+(-1)(2)+(-1)(0)+(1)(-2)}+{1(-2)+(-1)(2)+(-1)(0)+(-1)(2)+(1)(-2)}+{(-1)(2)+(1)(-2)+(1)(-2)+1(-2)+(-1)(2)} = -42 maka *f(net)* = -1

dengan membandingkan hasil *f(net)* ke target maka didapatkan hasil yang sangat baik. *Habb* dapat mengenali pola tersebut dengan baik.

4.2.3 Kelebihan

Jika diamati dalam beberapa contoh di atas dalam penyelesaikan contoh kasus yang telah di buat bahwa algoritma *Habb* ini, jika dua neuron dihubungkan dengan masukan secara serentak akan menjadi aktif baik itu sama – sama bernilai positif atau sama – sama bernilai negatif, maka kekuatan sinapsisnya itu akan meningkat. Begitu juga dengan sebalinya apabila dua neuron sama – sama aktif tetapi tidak sinkron di mana satu bernilai positif dan satu lagi bernilai negatif, maka kekuatan akan melemah.

4.2.4 Kekurangan

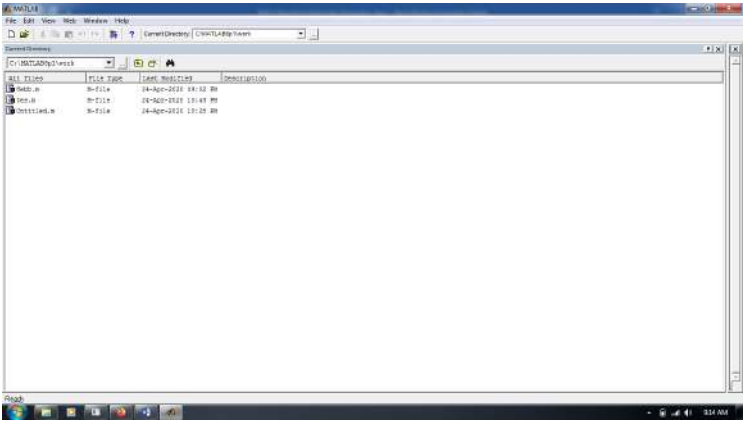
Dari beberapa contoh penyelesaian yang diberikan di atas, ada beberapa masalah yang harus ditangani dengan serius ataupun hal hal yang harus diperhatikan dalam menggunakan algoritma *Habb* ini. Di antaranya adalah dalam menentukan nilai masukan atau representasi nilai masukan juga nilai *output* untuk fungsi aktivasi yang berupa *trashoold.* Representasi yang mampu untuk diolah dalam menentukan pola adalah representasi dalam bentuk bipolar (nilai -1 atau 1) baik itu dalam bentuk masukan atau keluaran, sementara nilai dalam bentuk biner (nilai o atau 1) tidak mampu menyelesaikan persoalan dalam mengenali pola baik itu dalam nilai masukan atau keluaran.

4.3 Penerapan Algoritma Habb dengan Software

Di dalam penerapan algoritma Habb untuk mengenali pola dengan menggunakan *sofware* pendukung yaitu Matlab, kenapa menggunakan Matlab? karena ini adalah sebagai alat untuk pembuktian algoritma *Habb* yang telah kita buktikan secara matematis dapat mengenali pola, di mana masukan atau data input yang berupa data bipolar (nilai 1 dan -1). Dan ini juga tidak

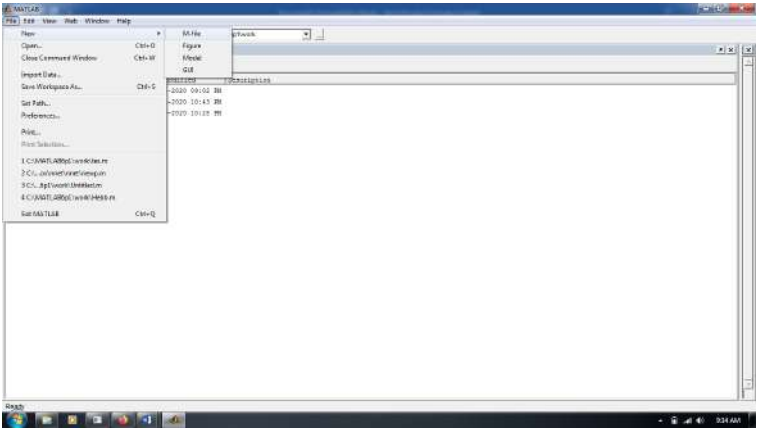
memberikan pemahaman pemahaman awal bagaimana menggunakan Matlab, karena akan memakan waktu untuk memperkenalkan bagaimana menjelaskan Matlab dari awal.

Baiklah kita akkan mencoba membuktikan perhitungan matematis kita ke dalam *sofware* pendukung yaitu Matlab. Hal pertama kita akan menjalankan sofware Matlab yang ada pada Laptop atau PC masing masing. Dengan tampilan awal yang tampak pada Gambar 4.3 sebagai berikut.



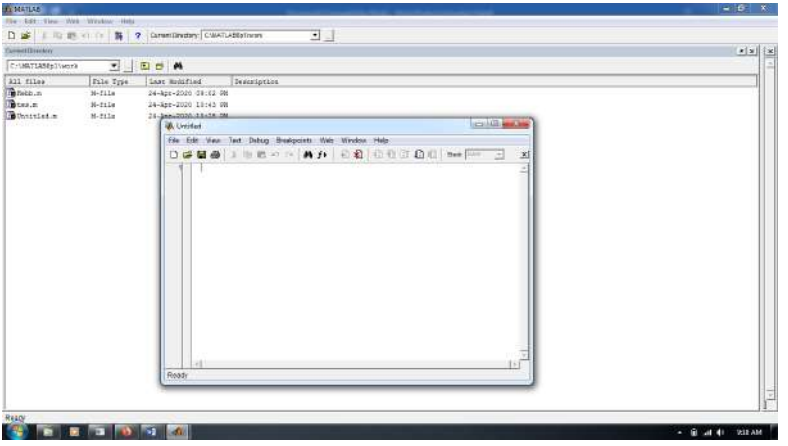
Gambar 4.3: Tampilan Matlap pada saat pertama dijalankan

Kemudian kita masuk pada menu File dan masuk ke new kemudian masuk pada menu tab M.file yang tampak pada Gambar 4.4 di bawah ini.



Gambar 4.4: Proses pembuatan lembar kerja

Setelah itu kita klik maka akan muncul lembah kerja kita sebagai lembar kerja baru yang tampak seperti Gambar 4.5 di bawah ini, di mana pada tempat inilah kita akan memasukkan koding untuk mengolah data data yang kita dapatkan sebelumnya pada topik pembahasan di atas.



Gambar 4.5: Lembar kerja yang masih kosong.

Setelah lembar kerja tersebut muncul, kita dapat memasukkan koding dan data data sesuai dengan pembahasan kita sebelumnya.

```
clc;

clear all;

close all;
```

pada koding pembuka di atas digunakan untuk membersihkan layar dan menutup aplikasi yang sedang berjalan. Di mana ini akan memberikan keleluasaan kita dalam mengolah data yang telah kita siapkan. Kemudian,

```
x = [1 1 -1 -1; 1 -1 1 -1];

t = [1 -1 -1 -1];

w = [0 0];

b = 0;
```

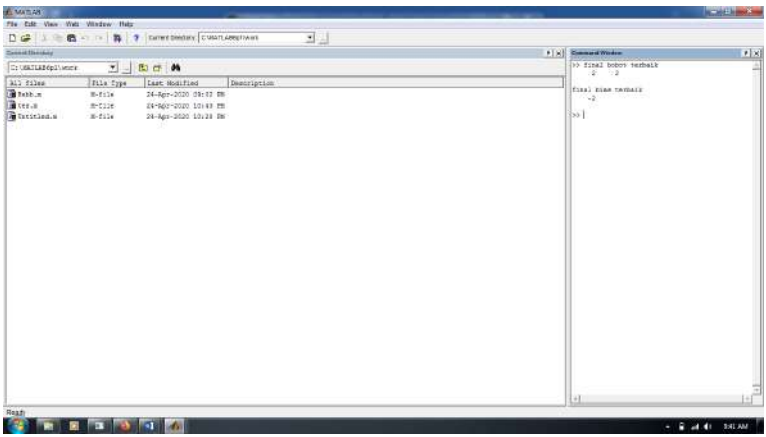
pada koding selanjutnya di atas tersebut, nilai yang terdapat pada x ini adalah nilai yang kita sebut sebagai nilai inputan yang berupa nilai bipolar, yaitu

tampak bernilai 1 dan -1, t adalah nilai target yang telah kita tentukan sebelumnya yaitu kembali menggunakan nilai bipolar sama seperti nilai pada masukan, w adalah bobot dari masukan di mana pada kasus ini kita mengesetnya dengan nilai "0", b adalah bias di mana pada kasus ini bias ini kita mengesetnya dengan nilai "0".

```
disp('bobot terbaik');

disp(w);

disp('bias terbaik');

disp(b);
```

pada *disp* ini dalam menampilkan baik itu yang ditampung oleh tipe data ataupun string yang akan ditampilkan ke layar pada saat koding di *ran* (jalankan), untuk w adalah *variable* penampung bobot setelah koding dijalankan dan menampilkanya ke dalam layar begitu juga dengan b ini adalah sebagai variable untuk penampung hasil proses dari koding yang sedang di jalan kan kemudian hasilnya akan di tempailkan kelayar.

Setelah semuanya dimasukkan baik itu dari data data yang akan dimasukkan berupa nilai inputan, bobot dan bias kemudian di compile dan tidak ada kesalahan. Hasilnya akan tampak pada Gambar 4.6 di bawah ini



Gambar 4.6: Hasil bobot terahir

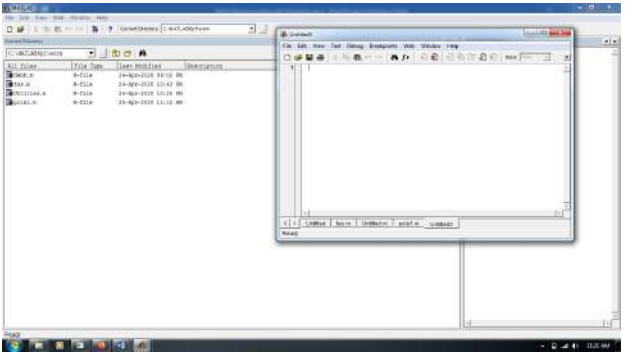
Pada tampilan consol output, setelah koding di compile dan dijalankan terdapat nilai bobot terbiak untuk masukan pertama atau x1 adalah 2 dan untuk x2

adalah 2. Kemudian pada bobot kita melihat disana itu adalah -2, kemudian kita akan membandingkan nilai hasil dari pada generate dengan menggunakan hasil software dengan menggunakan Matlab dan menggunakan perhitungan matematis yang tampak pada Tabel 4.13 di bawah ini .

Tabel 4.13: Perbandingan bobot

Bobot Hasil dengan perhitungan manual			**Bobot Hasil dengan menggunakan Matlab**		
X_1	X_2	b	X_1	X_2	b
2	2	-2	2	2	-2

Dari hasil tabel di atas kita dapatkan hasil adalah bahwasanya software Matlab dapat memberikan perhitungan dengan baik untuk mengenali pola bentuk dari fungsi AND, di mana hasil dari perhitungan manual dan hasil dati Matlab adalah sama. Kemudian pada contoh kasus no 4, di sana adalah sebanyak 25 masukan pada pola 1 dan 25 masukan pada pola 2, pada kesempatan ini kita akan mengolah data tersebut untuk di olah ke dalam Matlab tetapi ada sedikit kita modifikasi, di mana yang akan kita modifikasi adalah bagian pada targetnya namun tetap menggunakan biplar, kenapa di modifikasi karena di dalam perhitungan manual masukan sebanyak 25 pada pola 1 dan 25 pada pola 2, namun pada target dia hanya dua yaitu 1 dan -1. Oleh karena itu untuk masukan tadi pada pola 1 sebannyak 25 dan pola 2 sebanyak 25 maka kita akan mebuat targetnya itu kembali sebanyak 25. Ok baiklah kita akan memodifikasi nilai nilai tersebut. Kita akan membuat lembar kerja baru pada gambar 4.7 dan memasukkan kodingnya



Gambar 4.7: Lembar kerja baru untuk kasus ke 4

Ini adalah tampilan halaman karja yang baru dan kita akan memasukkan koding ke dalam lembar kerja yang baru tersebut.

```
x = [1 -1 -1 -1 1 -1 1 -1 1 -1 -1 -1 1 -1 -1 -1 1 -1 1 -1 1 -1 -1 -1 1;
-1 1 1 1 -1 1 -1 -1 -1 1 1 -1 -1 -1 1 1 -1 -1 -1 1 -1 1 1 1 -1];
t = [1 -1];
```

pada koding di atas kita telah memasukkan nilai pola 1 sampai pada tanda; kemudian diikuti pada pola ke 2, dan pada target telah kita set pada contoh kasis sebelumnya. Setelah dilakukan hal tersebut dan didapatkan nilai yang sama terhadap hasil perhitungan matematis secara manual.

Bab 5

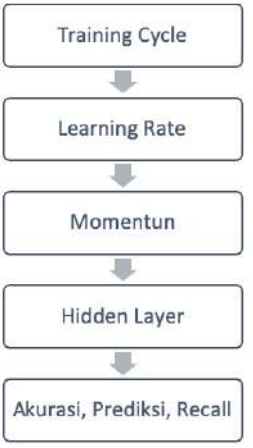
Prediksi Gaya Belajar VARK dengan Artificial Neural Network

5.1 Pendahuluan

Prediksi gaya belajar VARK dengan *Artificial Neural Network* (ANN) dibangun berdasarkan permasalahan dalam pembelajaran *online*. Salah satu permasalahan dari pembelajaran *online* adalah kebutuhan personalisasi untuk meningkatkan motivasi pembelajaran (Hasibuan and Purbo, 2018). Menurut Graf et.al salah satu bentu personalisasi adalah mengetahui gaya belajar yang dimilikinya. bahwa setiap pembelajar memiliki gaya belajar (Graf, 2009). Prediksi gaya belajar saat ini dibagi menjadi dua yakni konvensional dan otomatis, jika konvensional dengan menggunakan kuesioner maka otomatis dideteksi menggunakan ANN (Hasibuan and Nugroho, 2016).

Buku ini melakukan prediksi gaya belajar vark dengan menggunakan metode ANN. Adapun tools yang digunakan untuk mendeteksi gaya belajar dengan menggunakan *rapidminer*. *Rapidminer* merupakan tools berbasis GUI yang dapat digunakan untuk melakukan prediksi dan analisis dengan metode kecerdasan buatan yang telah disediakan(RapidMiner, 2012). Adapun dataset yang digunakan untuk mendeteksi gaya belajar dengan menggunakan dataset *prior knowledge* dan *behavior*. Kedua dataset tersebut memiliki karakteristik tersendiri. Buku ini akan menunjukkan langkah langkah dari prediksi gaya belajar dengan ANN dengan membandingkan kedua dataset tersebut.

Tahapan penggunaan ANN dengan rapidminer adalah sebagai berikut:



Gambar 5.1: Tahapan ANN

Langkah awal yang dilakukan dapat dilihat pada Tabel 5.1 yakni melakukan pengujian *training cycle*. Pengujian *training cycle* dimulai dari 200 sampai 1000. Pada *training cycle* 900 dan 1000 dihasilkan nilai akurasi tertinggi yaitu 91,48%. Maka *training cycle* yang digunakan adalah 900 dengan akurasi 91,48% dan deviasi +/-5,90.

Tabel 5.1: Pengujian *Training Cycle*

Training Cycles	***Learning Rate***	***Momentum***	**Deviasi**	**Akurasi**
200	0,1	0,1	+/-6,23%	76,33 %
400	0,1	0,1	+/-0,76%	82,58 %
600	0,1	0,1	+/-6,57%	85,04 %
700	0,1	0,1	+/-9,24%	87,31%
800	0,1	0,1	+/-9,24%	87,31 %
900	0,1	0,1	+/-5,90%	91,48 %
1000	0,1	0,1	+/-5,90%	91,48 %

Setelah melakukan pengujian *training cycle* dilanjutkan dengan pengujian learning rate dengan nilai 0,1 sampai 0,5 yang dapat dilihat pada tabel 5.2. Hasil pengujian menunjukkan bahwa cukup dengan nilai *learning rate* 0,1 telah menghasilkan nilai akurasi 91,48%.

Tabel 5.2: Pengujian *Learning Rate*

Training Cycles	Learning Rate	Momentum	Deviasi	Akurasi
900	0,1	0,1	+/-5,90%	91,48 %
900	0,2	0,1	+/-5,90%	91,48 %
900	0,3	0,1	+/-5,90%	91,48 %
900	0,4	0,1	+/-5,90%	91,48 %
900	0,5	0,1	+/-5,90%	91,48 %

Setelah melakukan pengujian *learning rate* di atas dilanjutkan pengujian *momentum.* Tabel 5.3 merupakan proses pengujian momentum untuk mendapatkan nilai akurasi yang tinggi.

Tabel 5.3*:* Pengujian Momentum

Training Cycles	Learning Rate	Momentum	Deviasi	Akurasi
900	0,1	0,1	+/-5,90%	91,48 %
900	0,1	0,2	+/-5,90%	91,48 %
900	0,1	0,3	+/-5,90%	91,48 %
900	0,1	0,4	+/-5,90%	91,48 %
900	0,1	0,5	+/-5,90%	91,48 %

Pada pengujian *momentum* di atas pengujian dimulai dengan memberikan nilai dari 0,1 hingga 0,5. Hasil pengujian menunjukkan bahwa nilai momentum 0,1 sudah memberikan hasil yang optimal dengan nilai akurasi 91,48% dan deviasi +/-5,90%. Setelah melakukan pengujian *momentum,* dilanjutkan pengujian *hidden layer* yang dapat dilihat pada tabel 5.4.

Tabel 5.4: Pengujian *Hidden Layer*

Jumlah *Hidden Layer*	**Akurasi**	**Deviasi**
1	60,98 %	+/-6,46%
2	71,97%	+/-8,55%
3	82,77%	+/-5,65%
4	87,31%	+/-9,24%
5	89,39 %	+/-6,86%
6	91,48 %	+/-5,90%
7	89,39%	+/-6,86%
8	91,48 %	+/-5,90%

Pada tabel 5.5 proses pengujian hidden layer dimulai dari nilai 1 sampai 8. Hasil akurasi tertinggi terdapat pada nilai hidden layer 6 dan 8. Pada penelitian ini nilai *hidden layer* yang digunakan adalah 6 dengan akurasi 91,48% dan deviasi +/-5,90%. Hasil akhir pengujian JST pada data PK dapat dilihat pada tabel 5.5 di bawah ini.

Tabel 5.5: Hasil Akurasi JST dengan data PK

Hidden Layer	**Training Cycles**	**Learning Rate**	**Momentum**	**Deviasi**	**Akurasi**
6	900	0,1	0,1	+/-5,90%	91,48 %

Pada tabel 5.5 di atas nilai akurasi yang diperoleh yaitu 91,48 % dengan nilai deviasi +/-5,90, nilai ini jauh lebih baik dibandingkan dengan deteksi gaya belajar dengan metode lain. Tabel 5.6 merupakan 1 pengujian dengan menggunakan AE dan MAE.

Tabel 5.6: Pengujian Absolute Error (AE) dan Median Absolute Error (MAE)

Gaya Belajar	Manusia	Sistem	AE	MAE
Visual	1	0,467	0,533	0,133
Audio	1	0,184	0,816	0,204
Read	1	0,137	0,863	0,216
Kinethestic	1	0,189	0,811	0,203

Tabel 5.6 merupakan pengujian menggunakan AE dan MAE menunjukkan bahwa nilai *error* masih besar. Hal ini dikarenakan keterbatasan data responden yang diambil hanya 14 pembelajar, sehingga pengujian AE dan MAE kurang tepat untuk konfirmasi.

Tabel 5.7*:* Pengujian dengan data PK menggunakan JST

NIS	Input PK				Output				Label	Prediksi
	KOF	KOM	AOK	IOK	V	A	K	R		
6581	3	3	2	2	0,061	0,821	0,008	0,109	Audio	Audio
6606	4	3	3	0	0,846	0,000	0,129	0,025	Visual	Visual
6623	2	3	2	0	0,017	0,904	0,002	0,077	Audio	Audio
6632	3	3	2	3	0,044	0,007	0,946	0,003	Kinethestic	Kinethestic
6553	4	3	2	3	0,899	0,024	0,076	0,000	Visual	Visual
6635	4	3	0	2	0,935	0,065	0,000	0,000	Visual	Visual
6532	3	3	2	3	0,031	0,094	0,870	0,006	Kinethestic	Kinethestic
6570	3	3	4	3	0,136	0,002	0,591	0,270	Audio	Kinethestic
6620	3	2	3	3	0,067	0,017	0,859	0,570	Visual	Kinethestic
6630	3	3	3	3	0,067	0,001	0,849	0,083	Kinethestic	Kinethestic
6538	3	3	3	2	0,097	0,184	0,031	0,687	Read	Read
6604	4	3	2	2	0,850	0,148	0,000	0,001	Visual	Visual
6572	4	3	2	4	0,185	0,002	0,813	0,000	Kinethestic	Kinethestic
6589	4	3	2	2	0,850	0,148	0,000	0,001	Visual	Visual

Pada tabel 5.7 disajikan hasil deteksi gaya belajar dengan menggunakan JST dengan menggunakan data PK. Hasil deteksi menunjukkan hanya 2 pembelajar yang tidak terkonfirmasi gaya belajarnya dari 14 pembelajar. Hal ini bermakna bahwa metode deteksi gaya belajar dengan menggunakan data PK hasilnya cukup tinggi 12/14 = 86%.

Tabel 5.8 menunjukkan pengujian *confusion matrix* menggunakan data PK untuk mendeteksi gaya belajar VARK.

Tabel 5.8*:* Confusion Matrix Data PK

LS /Prediction	**Audio**	**Visual**	**Kinesthetic**	**Read**	**Precsion**
Audio	7	1	0	0	87,50%
Visual	1	18	0	0	94,74%
Kineshetic	1	1	11	0	84,62%
Read	0	1	0	6	100,00
Recall	77,78%	90,00%	100,00%	100,00%	
Accuracy Cohen's Kappa (K)	91,48% 0,876				

Berdasarkan pengujian *confusion matrix* tabel 5.8 menggunakan data PK dihasilkan nilai akurasi diperoleh 91,48%, hasil perhitungan nilai *Cohen's Kappa* menunjukkan nilai 0,876. Hasil pengujian menunjukkan nilai akurasi sudah di atas 90% hal ditambah dengan dengan nilai *cohen's kappa* di atas 0,7 bermakna reliability tinggi.

5.2 Pengujian dengan Data Behavior

Tabel 5.9 merupakan pengujian dengan data behavior menggunakan JST, langkah awal yang dilakukan dengan pengaturan *training cycle.*

Tabel 5.9: Pengujian Training Cycle

Training Cycles	*Learning Rate*	*Momentum*	Deviasi	Akurasi
200	0,1	0,1	+/-18,95%	52,27 %
400	0,1	0,1	+/-11,94%	56,44 %
600	0,1	0,1	+/-6,71%	50,00 %
700	0,1	0,1	+/-4,83%	47,92%
800	0,1	0,1	+/-10,84%	48,11 %
900	0,1	0,1	+/-10,84%	48,11 %
1000	0,1	0,1	+/-10,84%	48,11 %

Pengujian training cycle dengan memberikan nilai dari 200 hingga 1000. Berdasarkan hasil pengujian *training cycle* hasil akurasi maksimal terdapat pada nilai training cycle 400 dengan nilai akurasi 56,44%. Pengujian selanjutnya yakni menguji nilai learning rate yang dapat dilihat pada tabel 5.10.

Tabel 5.10: Pengujian Learning Rate

Training Cycles	Learning Rate	Momentum	Deviasi	Akurasi
400	0,1	0,1	+/-11,94%	56,44 %
400	0,2	0,1	+/-14,10%	54,73 %
400	0,3	0,1	+/-13,89%	50,19 %
400	0,4	0,1	+/-11,86%	43,75 %
400	0,5	0,1	+/-7,62%	43,75 %
400	0,6	0,1	+/-7,62%	43,75 %
400	0,7	0,1	+/-5,26%	45,83 %
400	0,8	0,1	+/-8,93%	41,67 %

Tabel 5.10 melakukan pengujian dengan memberikan nilai learning rate dari 0,1 hingga 0,8, diperoleh hasil nilai akurasi tertinggi pada learning rate 0,1 dengan nilai akurasi 56,44%.

Hasil pengujian learning rate dengan *training cycle* 400 dan 0,1 ini memiliki nilai deviasi sebesar +/-11,94 %. Nilai ini cukup besar untuk pengukuran nilai deviasi yang sebaiknya di bawah +/-5%. Pengujian selanjutnya yang dilakukan setelah *learning rate* yakni pengujian nilai momentum yang dapat dilihat pada tabel 5.11.

Tabel 5.11: Pengujian Momentum

Training Cycles	Learning Rate	Momentum	Deviasi	Akurasi
400	0,1	0,1	+/-11,94%	56,44 %
400	0,1	0,2	+/-12,91%	54,17 %
400	0,1	0,3	+/-12,91%	54,17 %
400	0,1	0,4	+/-6,71%	50,00 %
400	0,1	0,5	+/-4,83%	47,92 %
400	0,1	0,6	+/-11,77%	50,19 %
400	0,1	0,7	+/-11,37%	54,73 %
400	0,1	0,8	+/-12,40%	43,94 %

Tabel 5.11 melakukan pengujian nilai momentum dengan memberikan nila 0,1 hingga 0,8. Nilai akurasi yang paling tinggi adalah 56,44% dengan deviasi nilai +/-11,94%. Selanjutnya setelah pengujian nilai momentum maka dilakukan pengujian nilai *hidden layer* yang dapat dilihat pada tabel 5.12

Tabel 5.12: Pengujian Hidden Layer

Jumlah *Hidden Layer*	**Akurasi**	**Akurasi**
1	43,75 %	+/-11,29%
2	54,55%	+/-8,11%
3	56,44%	+/-11,94%
4	56,63%	+/-15,28%
5	54,36 %	+/-14,73%
6	56,44%	+/-11,94%
7	56,63%	+/-15,28%
8	56,44%	+/-11,94%

Pada tabel 5.12 di atas dilakukan pengujian nilai *hidden layer* dari 1 sampai 8. Nilai hidden layer 3 memperoleh nilai akurasi yang tertinggi yakni 56,44%. Pada akhirnya hasil pengujian dengan menggunakan data *behavior* dapat dilihat pada tabel 5.13.

Tabel 5.13: Hasil Akurasi JST dengan data Behavior

Hidden Layer	Training Cycles	Learning Rate	Momentum	Deviasi	Akurasi
3	400	0,1	0,1	+/- 11,94%	56,44%

Pada Tabel 5.13 di atas dapat disampaikan bahwa hasil deteksi dengan data *behavior* hanya memiliki nilai akurasi paling tinggi 56,44%. Hal ini menyatakan bahwa data *behavior* yang digunakan untuk deteksi gaya belajar kurang akurat dibanding dengan menggunakan data *prior knowledge* dengan nilai akurai 91,48%.

Untuk membandingkan hasil pengujian lainnya digunakan pengujian *absolute eror* dan median *absolute error* yang dapat dilihat pada tabel 5.14.

Tabel 5.14: Pengujian Absolute Error

Gaya Belajar	Manusia	Sistem	AE	MAE
Visual	1	0,432	0,568	0,142
Audio	1	0,167	0,833	0,208
Read	1	0,133	0,867	0,217
Kinethestic	1	0,277	0,723	0,181

Pada tabel 5.14 di atas dapat dilihat bahwa nilai AE masih tinggi di atas 0,5. Hal ini dapat disebabkan bahwa variable interaksi yang di ambil hanya menghitung dimensi banyaknya kunjungan yang dilakukan pembelajar terhadap bahan ajar.

Agar hasil deteksi dapat diterima maka diperlukan konfirmasi kepada pembelajaran dengan jawaban ya atau tidak, hasil deteksi gaya belajar dapat dilihat pada tabel 5.15.

Tabel 5.15*:* Hasil Deteksi Data Behavior

NIS	Input Behavior				Output				Label	Prediksi
	V	A	R	K	V	A	K	R		
6581	4	2	2	2	0,763	0,101	0,124	0,012	Audio	Visual
6606	4	2	1	1	0,778	0,153	0,054	0,015	Visual	Visual
6623	3	4	2	2	0,215	0,488	0,122	0,175	Audio	Audio
6632	3	3	2	4	0,096	0,112	0,385	0,406	Kinethestic	Read
6553	4	3	2	3	0,749	0,090	0,153	0,008	Visual	Visual
6635	4	2	1	1	0,804	0,148	0,039	0,009	Visual	Visual
6532	4	4	3	3	0,318	0,241	0,333	0,107	Kinethestic	Kinethestic
6570	4	4	4	3	0,353	0,021	0,436	0,190	Audio	Kinethestic
6620	4	4	4	4	0,038	0,139	0,615	0,209	Visual	Kinethestic
6630	4	4	4	4	0,324	0,020	0,452	0,204	Kinethestic	Kinethestic
6538	3	3	3	2	0,028	0,126	0,732	0,114	Read	Kinethestic

6604	4	4	2	2	0,657	0,093	0,228	0,023	Visual	Visual
6572	3	4	2	4	0,075	0,122	0,405	0,398	Kinethestic	Kinethestic
6589	4	3	2	2	0,713	0,087	0,186	0,013	Visual	Visual

Hasil deteksi gaya belajar dengan menggunakan data behavior dapat dilihat pada tabel 5.15 yang menkonfirmasi 11 pembelajar, dan hanya 4 pembelajar yang tidak confirm, sehingga presentasinya adalah 9/14 =56%. Hal ini menunjukkan bahwa hasil deteksi dengan interaksi belum akurat dibanding hasil deteksi dengan menggunakan data PK.

Tabel 5.16: *Confusion Matrix Data Behavior*

LS /Prediction	**Audio**	**Visual**	**Kinesthetic**	**Read**	**Precsion**
Audio	5	2	0	0	71.43%
Visual	3	15	6	3	55.56%
Kineshetic	1	3	4	1	44.44%
Read	0	0	1	2	66.67%
Recall	55.56%	75.00%	36.36%	33.33%	
Accuracy Cohen's Kappa (K)	56.44% + 0,341				

Berdasarkan pengujian confusion matrix dengan menggunakan data behavior diperoleh nilai akurasi diperoleh 56,44% serta didukung dengan nilai *Cohen's Kappa 0,341* menunjukkan tingkat konsistensi dan *reliability* yang rendah. Hal ini bermakna metode deteksi dengan bahavior kurang akurat dibandingkan dengan menggunakan data PK.

Bab 6

Prediksi dengan Algoritma Levenberg-Marquardt

6.1 Pendahuluan

Seperti yang telah dibahas pada bab terdahulu, *backpropagation* merupakan algoritma pembelajaran dalam JST yang terawasi *(supervised learning*) yang menggunakan model *multilayer*. Pembelajaran *backpropagation* menggunakan pola penyesuaian bobot untuk mencapai nilai kesalahan minimum antara keluaran hasil prediksi dengan keluaran yang nyata. Dalam model *backpropagation* terdapat beberapa pelatihan yang bisa digunakan, yaitu algoritma *Fletcher-Reeves Update, Polak-Ribiere, PowellBeale Restarts, Scaled Conjugate Gradient, Gradient Descent dengan Momentum dan Adaptive Learning Rate, Resilent Backpropagation, BFGS, One Step Secant, Levenberg-Marquar*dt (Mustafidah, Rahmadhani and Harjono, 2019).

Algoritma Levenberg-Marquardt, yang dikembangkan secara independen oleh Kenneth Levenberg dan Donald Marquardt, memberikan solusi numerik untuk masalah meminimalkan fungsi non-linear. Cepat dan memiliki konvergensi yang stabil. Di bidang jaringan saraf tiruan, Algoritma ini cocok untuk melatih masalah kecil dan menengah. (Yu and Wilamowski, 2016). Algoritma ini merupakan pengembangan dari algoritma *error backpropagation* (EBP). Algoritma ini dibangun untuk mengatasi beberapa kekurangan yang ada pada algoritma EBP dengan memanfaatkan teknik optimisasi numerik standar yaitu menggunakan pendekatan matriks Jacobian. Tujuan dari levenberg marquardt adalah meminimalkan total error. Algoritma Levenberg–Marquardt dirancang secara khusus untuk meminimalisasi galat jumlah kuadrat. Algoritma ini

akan memperbarui bobot dengan cara memperkecil nilai galat dari selisih bobot baru dengan bobot lama. Namun, pada saat yang bersamaan, algoritma ini akan mempertahankan *step size* agar tidak menjadi besar (Udin, Kaloko and Hardianto, 2017).

Perbedaan algoritma Levenberg–Marquardt dengan *backpropagation* terletak pada proses update bobot (Hidayat, Isnanto and Nurhayati, 2013). Pada algoritma *backpropagation*. Proses update bobot pada Levenberg–Marquardt berdasarkan pendekatan Matrik Hessian. Matrik Hessian merupakan turunan kedua dari fungsi kinerja terhadap setiap komponen bobot dan bias. Penentuan matriks Hessian merupakan langkah dasar pada Levenberg–Marquardt untuk mencari bobot-bobot dan bias koneksi yang digunakan. Agar mempermudah proses komputasi, selama algoritma pelatihan berjalan, matriks Hessian diubah dengan pendekatan secara iteratif di setiap *epoch*.

Algoritma Levenberg–Marquardt adalah salah satu algoritma yang digunakan untuk meramalkan atau memperediksi hasil berikutnya berdasarkan data-data yang sudah ada sebelumnya (Yu and Wilamowski, 2016). Banyak penelitian menghasilkan simpulan bahwa algoritma Levenberg-Marquardt merupakan algoritma pelatihan yang paling optimal (Mustafidah, Budiastanto and Suwarsito, 2019).

Tabel 6.1: Perbandingan hasil pelatihan beberapa metode pelatihan JST dengan menggunakan data pelatihan yang sama (Susanti, 2014).

No	Nama/penemu metode pelatihan JST Backpropagation	Metode Pelatihan yang tersedia pada ToolboxMatlab	Jumlah Iterasi, Max 1500	MSE < 0.001
1	Scaled Conjugate Gradient	trainscg	53	8,632 x 10^{-04}
2	Fletcher-Reeves Update	traincgf	103	9,421 x 10^{-04}
3	Polak-Ribie're	traincgp	65	9,224 x 10^{-04}
4	Powel-Beale Restars	traincgb	47	8,315 x 10^{-04}
5	Broyden, Fletcher, Goldfarb, Shanno	trainbfg	28	7,299 x 10^{-04}
6	One Step Secant	trainoss	41	9,155 x 10^{-04}
7	Gradient Descent (GD) with Adaptive Learning Rate	traingda	1167	9,983 x 10^{-04}

8	GD with Momentum and Adaptive Learning Rate	traingdx	117	9,731 x 10^{-04}
9	Resilent Backpropagation	trainrp	210	9,823 x 10^{-04}
10	**Levenberg-Marquardt**	**trainlm**	**11**	**9,71 x 10^{-05}**

Keterangan: MSE, *Mean Square Error*

Pada Tabel 6.1 dapat dilihat bahwa dari kesepuluh metode pelatihan, metode pelatihan, Levenberg-Marquardt memberikan nilai MSE terkecil yaitu sebesar $9{,}71 \times 10^{-5}$ dengan epoh sebanyak 11. Epoh ini menunjukkan banyaknya iterasi atau langkah yang diperlukan oleh jaringan dalam melaksanakan pelatihan hingga dipenuhi kriteria tertentu untuk berhenti (Mustafidah, Budiastanto and Suwarsito, 2019).Ini berarti bahwa metode ini dapat mencapai fungsi tujuannnya lebih cepat dibandingkan dengan metode lainnya (Susanti, 2014).

6.2 Tahapan Pelatihan dengan Algoritma Lavenberg-Marquardt

Penerapan algoritma Levenberg-Marquardt untuk pelatihan jaringan syaraf tiruan, harus menyelesaikan dua masalah yaitu menghitung matriks Jacobian, dan bagaimana mengatur proses pelatihan iteratif untuk memperbarui bobot. Algoritma Levenberg-Marquardt memecahkan permasalahan yang ada di kedua metode gradient descent dan metode Gauss Newton untuk pelatihan neural-netwok, dengan kombinasi dari dua algoritma maka algoritma ini dianggap sebagai salah satu algoritma pelatihan yang paling efisien (Sylvia Jane Annatje Sunarauw, 2018).

Bobot-bobot awal dilatih dengan melalui tahap maju untuk mendapatkan error keluaran yang selanjutnya error ini digunakan dengan tahap mundur untuk memperoleh nilai bobot yang sesuai agar dapat memperkecil nilai error sehingga target keluaran yang dikehendaki tercapai. Tujuan dari model ini adalah untuk mendapatkan keseimbangan antara kemampuan jaringan mengenali pola yang digunakan selama proses pelatihan berlangsung serta kemampuan jaringan memberikan respon yang benar terhadap pola masukan yang berbeda dengan pola masukan selama pelatihan.

Algoritma *steepest descent*, juga dikenal sebagai algoritma *error backpropagation* (EBP) adalah algoritma orde pertama, dan algoritma *Gauss Newton* adalah orde kedua yang merupakan pengembangan dari algoritma Newton, di mana pada algoritma Newton digunakan matriks Hessian (H) sebagai derivatif orde kedua total fungsi galat yang merupakan proses perhitungan yang sangat rumit. Oleh karena itu, untuk menyederhanakan proses perhitungan pada algoritma *Gauss Newton* diperkenalkan matriks Jacobian (J)(Yu and Wilamowski, 2016)

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N^2} \end{bmatrix}$$

$$J = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \cdots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & \cdots & \frac{\partial e_{1,2}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & \cdots & \frac{\partial e_{1,M}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{P,1}}{\partial w_1} & \frac{\partial e_{P,1}}{\partial w_2} & \cdots & \frac{\partial e_{P,1}}{\partial w_N} \\ \frac{\partial e_{P,2}}{\partial w_1} & \frac{\partial e_{P,2}}{\partial w_2} & \cdots & \frac{\partial e_{P,2}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{P,M}}{\partial w_1} & \frac{\partial e_{P,M}}{\partial w_2} & \cdots & \frac{\partial e_{P,M}}{\partial w_N} \end{bmatrix}$$

Keterangan:

$\partial_{e1,1}$: turunan parsial pertama error jaringan

∂_{w_1} : turunan parsial turunan pertama bobot jaringan

Algoritma Levenberg-Marquardt memperkenalkan pendekatan lain untuk matriks Hessian dengan persamaan

$$H = J^T J + \mu I$$

Keterangan:

J^T : sebuah transpose dari matrik jacobian

J : matrik jacobian yang berisikan turunan pertama dari error jaringan terhadap bobot dan bias jaringan.

μ : selalu positif, yang disebut koefisien kombinasi

I : matriks identitas

Sedangkan gradien dapat dihitung dengan :

$$g = J \,.e$$

Keterangan:

e : vector yang menyatakan semua error pada output jaringan.

Perubahan pembobot (Δw) dapat dihitung dengan :

$$\Delta w = [J^T J + \mu I\,]^{-1} * J^T e$$

Keterangan:

e : konstanta learning

I : sebuah matrik identitas.

Langkah Pelatihan Levenberg–Marquardt dapat dijabarkan sebagai berikut (Zulfadli, 2019):

1. Inisialiasasikan bobot dan bias awal dengan bilangan acak kecil, target minimal, dan maksimum epoch. Target biasanya dihitung menggunakan *Mean of Squared Error* (MSE)
2. Tentukan parameter yang dibutuhkan, di antaranya:

a. Inisialisasi epoch = 0

b. Parameter Levenberg-Marquardt ($\mu > 0$). Parameter harus besar dari nol.

c. Parameter faktor Tau (τ), parameter ini yang berguna untuk dibagikan atau dikalikan dengan parameter Marquardt.

3. Perhitungan *Feedforward* untuk setiap unit input (x_i, = 1, 2, 3, . . . , n) menerima sinyal masukan dan diteruskan ke seluruh unit pada lapisan tersembunyi. Masing-masing unit lapisan tersembunyi (z_j, j = 1, 2,) menjumlahkan sinyal-sinyal input berbobot (v_{ij}) dan bias ($b1_j$).

$$z_{in_j} = b1_j + \sum_{i=1}^{n} x_i v_{ij}$$

Untuk menghitung sinyal output maka gunakan fungsi aktivasi.

$$z_j = f\,(z_{in_j})$$

Selanjutnya sinyal tersebut dikirimkan ke seluruh unit pada lapisan atasnya.

Keterangan:

z_{in_j} : Sinyal input yang masuk ke *hidden layer*

$b1_j$: Bobot awal dari bias *hidden layer*

x_i : Input ke i = 1, 2, 3, . . , n

v_{ij} : Bobot awal dari input layer ke *hidden layer*

z_j : Fungsi aktivasi pada *hidden layer*

4. Setiap unit lapisan output (Yk, k = 1, 2, 3, . . , m) menjumlahkan sinyal sinyal input berbobot (Wjk) dan bias ($b2k$).

$$y_{in_k} = b2_k + \sum_{i=1}^{n} z_j w_{jk}$$

Untuk menghitung sinyal output maka gunakan fungsi aktivasi.

$$y_k = f\,(y_{in_k})$$

Selanjutnya sinyal tersebut dikirimkan ke seluruh unit pada lapisan atasnya.

Keterangan:

y_{in_k} : Sinyal input yang masuk ke lapisan output

$b2_k$: Bobot awal dari bias ke output layer

z_j : Output dari setiap neuron hidden layer

w_{jk} : Bobot awal dari hidden layer ke output layer

y_k : Fungsi aktivasi pada output layer

5. Hitung error dan MSE.

   Rumus menghitung *error*.

$$e_r = t_r - y_r$$

$$e = [t_r - y_1 t_2 - y_2 \dots \dots t_r - y_r]^T$$

   Keterangan,

   r = *input* ke -r

   t_r = Target keluaran yang diharapkan

   e_r = Kesalahan pada unit keluaran (output)

   y_r = Keluaran *actual*

   Rumus menghitung MSE:

$$MSE = \frac{\sum_{i=1}^{n} er^2}{n}$$

6. Untuk tiap unit lapisan output, hitung *error neuron* (y_k, = 1,2,3,...,m)

$$\delta 2_k = (t_r - y_r)\, f'(y_{in_k})$$

$$\varphi 2_{jk} = \delta 2_k z_j$$

   $\beta 2_k = \delta 2_k$

   Kemudian hitung koreksi bobot untuk memperbaiki nilai w_{jk}

$$\Delta w_{jk} = \varphi 2_{jk}$$

   Untuk memperbaiki nilai $\beta 2_k$, maka hitung koreksi bias.

$$\Delta b2_k = \beta 2_k$$

Keterangan:

$\delta 2_k$: Error neuron utk tiap lapisan keluaran (*output*)

$\varphi 2_{jk}$: Menghitung bobot pada *output*

$\beta 2_k$: Menghitung bias pada *output*

Δw_{jk} : Koreksi bobot untuk memperbaiki nilai w_{jk}

$\Delta b 2_k$: Nilai koreksi bias untuk memperbaiki $\beta 2_k$

7. Error neuron dihitung untuk setiap unit lapisan tersembunyi (z_j, = 1,2,3,...p)

$$\delta_{in_j} = \sum_{k=1}^{m} \delta 2_k w_{jk}$$

$$\delta 1_j = \delta_{in_j} f'(z_{in_j}) 1 - (z_{in_j})$$

$$\varphi 1_{ij} = \delta 1_j x_j$$

$$\beta 1_j = \delta 1_j$$

Selanjutnya menghitung koreksi bobot (koreksi bobot berguna untuk memperbaiki nilai v_{ij} nantinya), yaitu:

$$\Delta v_{ij} = \varphi 1_{ij}$$

koreksi bias juga dihitung (nantinya digunakan untuk memperbaiki $b1_j$):

$$\Delta b 1_j = \beta 1_j$$

Keterangan:

δ_{in_j} : Error neuron untuk tiap lapisan tersembunyi (*hidden layer*)

$\delta 1_j$: Error neuron untuk tiap *hidden layer*

$\varphi 1_{ij}$: Menghitung bobot pada *hidden layer*

$\beta 1_j$: Menghitung bias pada *hidden layer*

Δv_{ij} : Koreksi bobot untuk memperbaiki nilai v_{ij}

$\Delta b 1_j$: Koreksi bias untuk memperbaiki $\beta 1_j$

8. Bentuk matriks Jacobian $J(x)$. X yaitu matriks yang berisikan nilai koreksi bobot dan bias dari seluruh jaringan.

$$J = [\varphi 1_{11} \dots \varphi 1_{np} \beta_{11} \dots \beta 1_p \varphi 2_{11} \dots \varphi 2_{pm} \beta 2_1 \dots \beta 2_m]$$

9. Menghitung bobot baru

$$W_{baru} = W_{lama} - [J^T J + \mu I]^{-1} J^T e$$

Keterangan:

J : Matriks Jacobian

J^T : Transpos Matriks Jacobian

I : Matriks Identitas

e : Nilai *Error*

10. Hitung MSE Jika MSE baru <= MSE lama, maka

    a. $\mu = \mu \tau$

    b. $epoch = epoch + 1$

    c. Kembali ke langkah 3

Jika MSE baru > MSE lama, maka

    a. $\mu = \mu * \tau$

    b. Kembali ke langkah 9

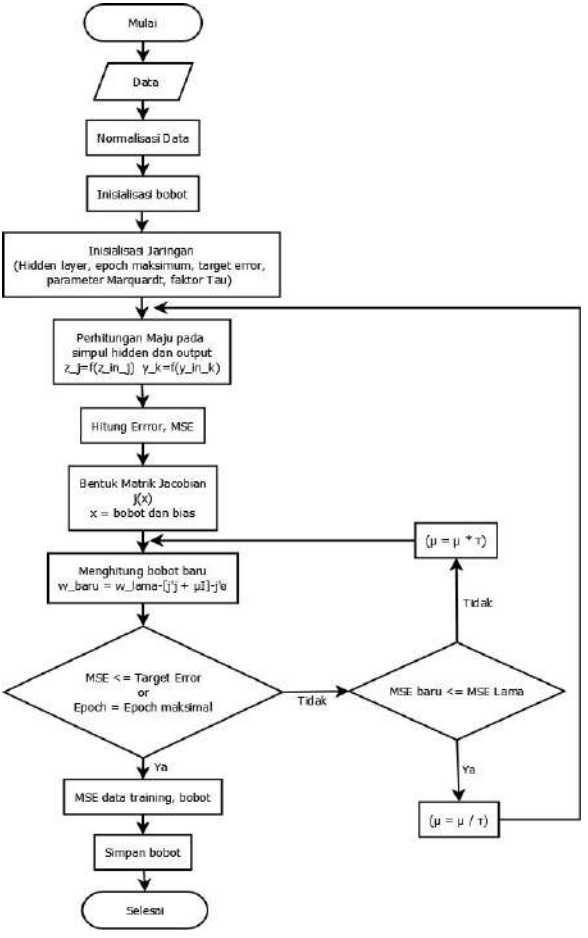
Keterangan:

μ : Parameter Marquardt

τ : Parameter faktor Tau

11. Proses pelatihan berhenti jika error = target error ataupun *epoch* ≥ *epoch* maksimal.

Implementasi algoritma Levenberg- Marquardt untuk pelatihan *neural network* adalah seperti diagram alir pada Gambar 6.1.



Gambar 6.1: Pelatihan JST Levenberg-Marquardt

6.3 Proses Pelatihan dengan Algoritma Lavenberg-Marquardt

Ada banyak penelitian Jaringan Syaraf Tiruan dengan menggunakan algoritma Levenberg-Marquardt untuk memprediksi atau meramalkan, di antaranya (Hidayat, Isnanto and Nurhayati, 2013) berkesimpulan Arsitektur JST ini yang

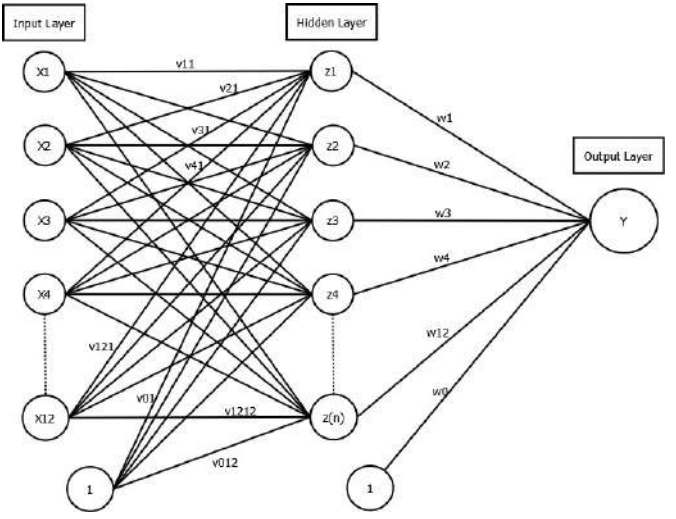
paling tepat dalam memrediksi harga logam mulia emas dengan hasil akurasi yang memiliki selisih terendah antara data latih dengan data uji, diperoleh data latih dengan akurasi 99,760% dan akurasi data uji 98,849%, kombinasi *neuron* 10-30 dengan laju pembelajaran sebesar 0,00001 dan galat sebesar 0,00001.Penelitian (Merdekawati and Ismail, 2019) berpendapat algoritma Levenberg–Marquardt memiliki performance yang baik dalam memprediksi curah hujan kota Jakarta.

Seperti yang telah dijelaskan di atas bahwa perbedaan algoritma Levenberg–Marquardt dengan Backpropagation terletak pada proses update bobot, maka pada sub bab ini hanya akan menguraikan contoh serta tahapan perhitungan pelatihan menggunakan algoritma Levenberg–Marquardt, sedangkan cara pengujian data target dan bobot hasil dari pelatihan, dilakukan dengan menggunakan algoritma *backpropagation* yang telah dijelaskan di bab sebelumnya. Sebagai contoh, perhitungan pelatihan menggunakan algoritma Levenberg Maquardt ini menggunakan data acak sebanyak 168 (seratus enam puluh delapan) dengan asumsi data latih yang digunakan 90%, sehingga data yang akan dilatih berjumlah 151.

Pada tabel 6.2, data tersebut telah dinormalisasi dan menggunakan arsitektur jaringan syaraf tiruan Levenberg Maquardt pada gambar 6.2. Data latih yang telah dilatih dengan algoritma LM nantinya akan dijadikan acuan untuk mengetahui pola prediksi data yang muncul dimasa yang akan datang.

Tabel 6.2 Contoh Data Hasil Normalisasi

No	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	Target
1	0,396	0,283	0,194	0,562	0,540	0,314	0,498	0,366	0,250	0,262	0,462	0,216	0,272
2	0,283	0,194	0,562	0,439	0,437	0,272	0,349	0,324	0,238	0,386	0,322	0,272	0,349
3	0,194	0,562	0,540	0,253	0,549	0,349	0,467	0,536	0,342	0,295	0,434	0,349	0,467
4	0,562	0,540	0,314	0,321	0,316	0,467	0,378	0,435	0.402	0,328	0,554	0,467	0,378
…	…	…	…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…
150	0,244	0,313	0,210	0,221	0,463	0,127	0,171	0,442	0,332	0.451	0,527	0,127	0,171
151	0,313	0,210	0,081	0,539	0,341	0,171	0,273	0,263	0,143	0,248	0,341	0,171	0,273



Gambar 6.2: Arsitektur Jaringan LM Prediksi (Zulfadli, 2019)

Proses pelatihan dengan algoritma Levenberg-Marquardt dalam kasus di atas dapat diuraikan sebagai berikut:

Tahap I : Memasukkan Data Pelatihan

Data yang digunakan adalah data ke 1 yang sudah dinormalisasi pada table 6.2

Data ke-1 = (X1 = 0.396, X2 = 0.283, X3 = 0.194, X4 = 0.562, X5 = 0.540, X6 = 0.314, X7 = 0.498, X8 = 0.366, X9 = 0.250, X10 = 0.262, X11 = 0.462, X12 = 0.216, Target = 0.272)

Tahap II : Tentukan Parameter

Parameter yang dibutuhkan, diantaranya:

- Jumlah neuron pada lapisan tersembunyi = 12
- Nilai parameter Marquardt (η) = 0.1
- Faktor tau (τ) =10
- Maksimum epoch = 1
- Target error = 0.5

Tahap III : Tentukan Bobot dan Bias

Setelah parameter ditentukan, langkah selanjutnya inisialisasi bobot awal setiap neuron dan bias pada lapisan masukan dan lapisan tersembunyi secara acak. Inisialisasi bobot dan bias dapat dilihat pada Tabel 6.3 dan 6.4 berikut:

Tabel 6.3: Bobot dan Bias *Input Layer* ke *Hidden Layer*

	X1	**X2**	**X3**	**X4**	**---**	**X12**	**Bias**
Z1	0,100	0,400	0,100	0,400	---	0,100	0,200
Z2	0,200	0,300	0,200	0,300	---	0,200	0,300
Z3	0,300	0,200	0,300	0,200	---	0,300	0,400
Z4	0,400	0,100	0,400	0,100	---	0,400	0,400
Z5	0,400	0,100	0,400	0,100	---	0,400	0,300
Z6	0,300	0,200	0,300	0,200	---	0,300	0,200
Z7	0,200	0,300	0,200	0,300	---	0,200	0,100
Z8	0,100	0,400	0,100	0,400	---	0,100	0,100
Z9	0,100	0,400	0,100	0,400	---	0,100	0,200
Z10	0,400	0,200	0,300	0,100	---	0,400	0,300
Z11	0,300	0,200	0,200	0,200	---	0,300	0,200
Z12	0,200	0,300	0,100	0,300	---	0,200	0,400

Tabel 6.4 Bobot dan Bias *Hidden Layer* ke *Output Layer*

	X1	X2	X3	X4	---	X12	Bias
Y1	0,200	0,300	0,400	0,400		0,100	0,100

Tahap IV : Tahap *Feedforward* Levenberg–Marquardt

1. Data pertama akan dilakukan proses hitung maju menggunakan Persamaan :

$$Z_{in_j} = \mathrm{b}b1_j + \sum_{i=1}^{n} x_i v_{ij}$$

Pada proses ini langkah pertama akan dihitung sinyal input yang merupakan bobot dan bias dengan cara menjumlahkan semua sinyal yang masuk pada lapisan tersembunyi.

Z_{in_1} = 0,200 + 0,396 ∗ 0,100 + 0,283 ∗ 0,400 + 0,194 ∗ 0,100 + 0,562 ∗ 0,400 + 0,540 ∗ 0,100 + 0,314 ∗ 0,400 + 0,498 ∗ 0,100 + 0,366 ∗ 0,400 + 0,250 ∗ 0,100 + 0,262 ∗ 0,400 + 0,462 ∗ 0,100 + 0,216 ∗ 0,100 = **1,171**

Dengan cara yang sama dicari Z_{in_2} sampai dengan $Z_{in_{12}}$ maka, di dapat hasil pada table 6.5

Tabel 6.5: Penjumlahan Sinyal Input dan Bobot Pada Hidden Layer

Sinyal Input dan Bobot	Jumlah
$z_{in}(1)$	1,171
$z_{in}(2)$	1,348
$z_{in}(3)$	1,525
$z_{in}(4)$	1,602
$z_{in}(5)$	1,502
$z_{in}(6)$	1,325
$z_{in}(7)$	1,148
$z_{in}(8)$	1,071
$z_{in}(9)$	1,171
$z_{in}(10)$	1,465
$z_{in}(11)$	1,259
$z_{in}(12)$	1,382

Selanjutnya akan dihitung keluaran dari setiap neuron lapisan tersembunyi dengan fungsi aktivasi.

$$z_1 = \frac{1}{1 + e^{-1,171}} = \mathbf{0,763}$$

Tabel 6.6: Nilai Pada *Hidden Layer* Setelah Diaktifasi

Keluaran Setiap Neuron	**Nilai**
$z(1)$	0,763
$z(2)$	0,794
$z(3)$	0,821

Keluaran Setiap Neuron	Nilai
$z(4)$	0,832
$z(5)$	0,818
$z(6)$	0,790
$z(7)$	0,759
$z(8)$	0,745
$z(9)$	0,763
$z(10)$	0,812
$z(11)$	0,779
$z(12)$	0,799

2. Setelah diperoleh sinyal keluaran dari lapisan tersembunyi, kemudian sinyal tersebut dikirimkan kesemua unit pada lapisan keluaran. Penjumlahan sinyal-sinyal dari lapisan tersembunyi masuk ke lapisan keluaran

   y_{in} = 0,100 + 0,200 ∗ 0,763 + 0,300 ∗ 0,794 + 0,400 ∗ 0,821 + 0,400∗ 0,832 + 0,300 ∗ 0,818 + 0,200 ∗ 0,790 + 0,100 ∗ 0,759 + 0,100 ∗ 0,745 + 0,200 ∗ 0,763 + 0,400 ∗ 0,812 + 0,300 ∗ 0,779 + 0,100 ∗ 0,799 = **2,497**

   Selanjutnya hitung keluaran dari neuron lapisan keluaran dengan fungsi aktivasi sigmoid biner.

$$y = \frac{1}{1 + e^{-2,497}} = \mathbf{0,924}$$

3. Jika sudah mendapat nilai dari lapisan keluaran maka dapat dihitung error dan MSE.

   $Error$ = 0,272 − 0,924 = −**0,52**

   $Jumlah\ Quadrat\ Error$ = (− 0,652) 2= **0,26**

   Setelah mendapatkan MSE maka dibandingkan dengan target error. MSE tersebut masih lebih besar daripada target error yang ditetapkan di awal proses training sehingga dilanjutkan tahap pembentukan matriks *Jacobian*.

Tahap V : Tahap Backforward

Setelah tahapan *feedforward*, selanjutnya masuk tahapan backforward. Untuk mendapatkan perubahan bobot, buat matriks *Jacobian*. Matriks Jacobian ini akan digunakan untuk mendapatkan bobot baru. Tahapan *backforward* terdiri dari beberapa tahapan sebagai berikut:

1. Untuk tiap unit lapisan output menerima pola target untuk menghitung *error*:

$$\delta 2_k = 0.652 \times 0{,}924 \times (1 - 0.924) = \mathbf{-0,046}$$

Hitung $\varphi 2_{jk}$

$$\varphi 2_{11} = -0{,}046 * 0{,}763 = \mathbf{-0,035}$$

$$\varphi 2_{12} = -0{,}046 * 0{,}794 = \mathbf{-0,036}$$

Tabel 6.7: Bobot pada *Output*

Bobot pada Output	Jumlah
$\varphi 2_{11}$	-0,035
$\varphi 2_{12}$	-0,036
$\varphi 2_{13}$	-0,038
$\varphi 2_{14}$	-0,038
$\varphi 2_{15}$	-0,037
$\varphi 2_{16}$	-0,036
$\varphi 2_{17}$	-0,035
$\varphi 2_{18}$	-0,034
$\varphi 2_{19}$	-0,035
$\varphi 2_{110}$	-0,037
$\varphi 2_{111}$	-0,036
$\varphi 2_{112}$	-0,037

Selanjutnya nilai bias pada output ($\beta 2k$):

$$\beta 2_{k=} \mathbf{-0,046}$$

Setelah didapat error-nya, hitung nilai koreksi bobot yang nantinya akan digunakan untuk memperbaiki nilai bobot antara lapisan tersembunyi dan lapisan output.

$$\Delta W_{jk} = \varphi 2_{jk}$$

$$\Delta W_{11} = \mathbf{-0,035}$$

Lakukan perhitungan seperti di atas untuk mencari nilai ΔW_{12} sampai ΔW_{11}. Untuk melihat nilai ΔW_{jk} dapat dilihat pada Tabel 6.8 berikut:

Tabel 6.8: Koreksi bobot untuk memperbaiki nilai $\boldsymbol{w_{jk}}$

Koreksi Bobot	Nilai
ΔW_{11}	-0,035
ΔW_{12}	-0,036
ΔW_{13}	-0,038
ΔW_{14}	-0,038
ΔW_{15}	-0,037
ΔW_{16}	-0,036
ΔW_{17}	-0,035
ΔW_{18}	-0,034
ΔW_{19}	-0,035
ΔW_{20}	-0,037
ΔW_{21}	-0,036
ΔW_{22}	-0,037

Untuk memperbaiki nilai $\beta 2_k$, maka hitung koreksi bias

$$\Delta b2_k = \mathbf{-0,046}$$

2. Kemudian setiap unit lapisan tersembunyi menjumlahkan sinyal-sinyal *input* dari lapisan *output*. *Error neuron* dihitung untuk setiap unit lapisan tersembunyi (z_j, = 1,2,3,...p)

   Hitung δin_j

$$\delta in_1 = -0{,}046 \times 0{,}200 = \mathbf{-0,009}$$

$$\delta in_2 = -0{,}046 \times 0{,}300 = \mathbf{-0,013}$$

Lakukan perhitungan seperti di atas untuk mencari nilai δin_3 sampai δin_{12}. Untuk melihat nilai δin_1 sampai δin_{12} dapat dilihat pada Tabel 6.9 berikut:

Tabel 6.9: Hasil Penjumlahan Sinyal-Sinyal Input dari Lapisan Output

Sinyal Input dari Lapisan Output	Jumlah
δin_1	-0,009
δin_2	-0,013
δin_3	-0,018
δin_4	-0,018
δin_5	-0,014
δin_6	-0,009
δin_7	-0,005
δin_8	-0,005
δin_9	-0,009
δin_{10}	-0,018
δin_{11}	-0,014
δin_{12}	-0,005

3. Kalikan nilai tersebut dengan fungsi aktivasinya (*sigmoid biner*) untuk menghitung informasi error pada lapisan tersembunyi. Hitung $\delta 1j$:

$$\delta 1_1 = -0{,}009 * \left(\frac{1}{1+\mathrm{e}^{-1{,}171}}\right) \mathrm{x} \left(1 - \frac{1}{1+\mathrm{e}^{-1{,}171}}\right)$$

$$= -\,0{,}009 * (0{,}763) \times (1-0{,}763) = \mathbf{-0{,}0016}$$

Lakukan perhitungan seperti di atas untuk mencari nilai $\delta 1_2$ sampai $\delta 1_{12}$.

Tabel 6.10: Hasil Informasi Error pada Lapisan Tersembunyi

Informasi Error	Hasil
$\delta 1_1$	-0,0016
$\delta 1_2$	-0,0022
$\delta 1_3$	-0,0027

Informasi Error	Hasil
$\delta 1_4$	-0,0025
$\delta 1_5$	-0,0020
$\delta 1_6$	-0,0015
$\delta 1_7$	-0,0008
$\delta 1_8$	-0,0009
$\delta 1_9$	-0,0016
$\delta 1_{10}$	-0,0028
$\delta 1_{11}$	-0,0024
$\delta 1_{12}$	-0,0007

4. Kemudian hitung koreksi bobot antara lapisan *input* dan lapisan tersembunyi yang nantinya akan digunakan untuk memperbaiki nilai bobot dan bias antara lapisan input dan lapisan tersembunyi tersebut. Menghitung hasil perbaikan bobot $\varphi 1_{ij}$:

$$\Delta v_{11} = 0{,}0016 \times 0{,}396 = \mathbf{-0{,}00065}$$

$$\Delta v_{12} = 0{,}0016 \times 0{,}283 = \mathbf{-0{,}00089}$$

Lakukan perhitungan seperti di atas untuk mencari nilai ΔV_{13} sampai ΔV_{1212} Untuk melihat nilai ΔV_{11} sampai ΔV_{112} dapat dilihat pada Tabel 6.11 berikut:

Tabel 6.11: Koreksi Bobot Antara Lapisan *Input* dan Tersembunyi

X1		**X2**		**…**		**X4**	
ΔV_{11}	-0,00065	ΔV_{21}	-0,00046	…	…	ΔV_{121}	-0,00035
ΔV_{12}	-0,00089	ΔV_{22}	-0,00063	…	…	ΔV_{122}	-0,00048
ΔV_{13}	-0,00106	ΔV_{23}	-0,00076	…	…	ΔV_{123}	-0,00058
ΔV_{14}	-0,00101	ΔV_{24}	-0,00072	…	…	ΔV_{124}	-0,00055
ΔV_{15}	-0,00081	ΔV_{25}	-0,00058	…	…	ΔV_{125}	-0,00044
ΔV_{16}	-0,00060	ΔV_{26}	-0,0004	…	…	ΔV_{126}	-0,00032
ΔV_{17}	-0,00033	ΔV_{27}	-0,00023	…	…	ΔV_{127}	-0,00018
ΔV_{18}	-0,00034	ΔV_{28}	-0,00024	…	…	ΔV_{128}	-0,00018
ΔV_{19}	-0,00065	ΔV_{29}	-0,00046	…	…	ΔV_{129}	-0,00035

X1		X2		…		X4	
ΔV_{110}	-0,00110	ΔV_{210}	-0,00079	…	…	ΔV_{1210}	-0,00060
ΔV_{111}	-0,00093	ΔV_{211}	-0,00067	…	…	ΔV_{1211}	-0,00051
ΔV_{112}	-0,00091	ΔV_{212}	-0,00020	…	…	ΔV_{1212}	-0,00015

Kemudian lakukan perhitungan untuk koreksi bias antara lapisan input ke lapisan tersembunyi. Hitung $\Delta b1_j$:

$$\Delta b1_1 = -\mathbf{0{,}016}$$

$$\Delta b1_2 = -\mathbf{0{,}022}$$

Lakukan perhitungan seperti di atas untuk mencari nilai $\Delta b1_3$ sampai $\Delta b1_{12}$. Untuk melihat nilai $\Delta b1_1$ sampai $\Delta b1_{12}$ dapat dilihat pada Tabel 6.12 berikut:

Tabel 6.12: Hasil Koreksi Bias Antara Lapisan Input dan Tersembunyi

Koreksi Bias Input Hidden	Hasil
$\Delta b1_1$	-0,0016
$\Delta b1_2$	-0,0022
$\Delta b1_3$	-0,0027
$\Delta b1_4$	-0,0025
$\Delta b1_5$	-0,0020
$\Delta b1_6$	-0,0015
$\Delta b1_7$	-0,0008
$\Delta b1_8$	-0,0009
$\Delta b1_9$	-0,0016
$\Delta b1_{10}$	-0,0028
$\Delta b1_{11}$	-0,0024
$\Delta b1_{12}$	-0,0007

5. Bentuk matriks *Jacobian* J(x) yaitu matriks yang berisikan nilai koreksi bobot dan bias dari seluruh jaringan. Matriks Jacobian ini akan digunakan untuk mendapatkan bobot baru. Ukuran matriks Jacobian yang terbentuk adalah 1x169. Yaitu matriks dengan dimensi 1 baris dan 169 kolom.

$$J = [-0{,}00065\ -0{,}00046\ -0{,}00032\ \ldots\ -0{,}04584\ ]_{1x169}$$

a. Langkah selanjutnya setelah mendapat matriks *Jacobian* adalah menghitung perubahan bobot

   Pertama, buat *transpose* dari matriks *Jacobian*.

$$j_T = \begin{bmatrix} -0{,}00065 \\ -0{,}00046 \\ \ldots \\ -0{,}04584 \end{bmatrix}_{169x1}$$

b. Setelah itu transpose dari matriks Jacobian dikalikan dengan matriks Jacobian menghasilkan matriks berikut:

$$j_T j = \begin{bmatrix} 4{,}303x10^{-7} & 3{,}078x10^{-7} & \ldots & 3x10^{-5} \\ 3{,}078x10^{-7} & 2{,}202x10^{-7} & \ldots & 2{,}151x10^{-5} \\ \ldots & \ldots & \ldots & \ldots \\ 3{,}007x10^{-5} & 2{,}151x10^{-5} & \ldots & 0{,}002 \end{bmatrix}_{169x169}$$

c. Setelah itu buat matriks identitas sesuai ordo hasil kali transpose matriks Jacobian dengan matriks Jacobian. Matriks identitas sebagai berikut:

$$I = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \\ 0 & 0 & \ldots & 1 \end{bmatrix}_{169x169}$$

   Kemudian matriks identitas dikalikan dengan parameter Marquardt yang sudah diinisialisasi sebelumnya. Nilai parameter Marquardt diinisialisasikan 0,1, dikalikan dengan matriks identitas menjadi matriks berikut ini:

$$\mu * I \begin{bmatrix} 0{,}1 & 0 & \ldots & 0 \\ 0 & 0{,}1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \\ 0 & 0 & \ldots & 0{,}1 \end{bmatrix}_{169x169} =$$

d. Lalu tambahkan hasil tersebut dengan perkalian *transpose* Jacobian dan Jacobian, hasilnya sebagai berikut:

$$J^T J + \mu\, I =$$

$$\begin{bmatrix} 0{,}1000004 & 3{,}078 \times 10^{-7} & \dots & 3 \times 10^{-5} \\ 3{,}078 \times 10^{-7} & 0{,}1000002 & \dots & 2{,}151x10^{-5} \\ \dots\dots\dots & \dots\dots\dots & \dots\dots\dots & \dots\dots\dots \\ 3{,}007x10^{-5} & 2{,}151x10^{-5} & \dots & 0{,}102 \end{bmatrix}_{169x169}$$

e. Setelah mendapat hasil penjumlahan, maka dilakukan perhitungan invers, menghasilkan matriks berikut ini:

$$[\mathrm{J}^{\mathrm{T}}\mathrm{J} + \mu\,\mathrm{I}]^{-} =$$

$$\begin{bmatrix} 9{,}999 & -2{,}61 \times 10^{-5} & \dots & -0{,}0025 \\ -2{,}61 \times 10^{-5} & 9{,}999 & \dots & -0{,}0018 \\ \dots\dots\dots & \dots\dots\dots & \dots\dots\dots & \dots\dots\dots \\ -0{,}0025 & -0{,}0018 & \dots & 9{,}8218 \end{bmatrix}_{169x169}$$

f. Hasil perhitungan invers kemudian dikalikan dengan *transpose* matriks *Jacobian*, hasilnya sebagai berikut:

$$j^T j + \mu\,\mathrm{I}]^{-} j^T = \begin{bmatrix} -0{,}0055 \\ -0{,}0039 \\ \dots \\ -0{,}3885 \end{bmatrix}_{169x1}$$

g. Hasil perkalian pada tahap ke e kemudian dikalikan dengan nilai error sehingga mendapatkan perubahan bobot seperti berikut ini:

$$j^T j + \mu\,\mathrm{I}]^{-} j^T e = \begin{bmatrix} 0{,}0036 \\ 0{,}0025 \\ \dots \\ 0{,}2535 \end{bmatrix}_{169x1}$$

Hasil matriks di atas diuraikan pada Tabel 6.13 dan Tabel 6.14 berikut:

Tabel 6.13: Koreksi Bobot dan Bias *Input Layer* ke *Hidden Layer*

	1	**2**	**…**	**12**	**Bias**
ΔV1	0,0036	0,0026	…	0,0020	0,0092
ΔV2	0,0049	0,0035	…	0,0027	0,0124
ΔV3	0,0059	0,0042	…	0,0032	0,0149
ΔV4	0,0056	0,0040	…	0,0031	0,0142
ΔV5	0,0045	0,0032	…	0,0025	0,0113

	1	2	…	12	Bias
ΔV6	0,0033	0,0024	…	0,0018	0,0084
ΔV7	0,0018	0,0013	…	0,0010	0,0046
ΔV8	0,0019	0,0014	…	0,0010	0,0048
ΔV9	0,0036	0,0026	…	0,0020	0,0092
ΔV10	0,0061	0,0044	…	0,0033	0,0155
ΔV11	0,0052	0,0037	…	0,0028	0,0131
ΔV12	0,0016	0,0012	…	0,0009	0,0041

Tabel 6.14: Koreksi Bobot dan Bias *Hidden Layer* ke *Output Layer*

	1	2	…	12	Bias
Δ*w*	0,1935	0,2012	…	0,2026	0,2535

6. Untuk mendapatkan bobot baru, bobot lama dikurangi dengan perubahan bobot yang diperoleh dari tahap ke 5.g. (ΔV1= 0,0036 dibulatkan menjadi 0,004 dan seterusnya) Bobot dan bias baru dapat dilihat pada Tabel 6.15 dan 6.16 berikut:

$$V_{11baru} = 0{,}100 - 0{,}004 = \mathbf{0{,}96}$$

$$V_{12baru} = 0{,}400 - 0{,}003 = \mathbf{0{,}97}$$

Tabel 6.15: Bobot dan Bias Baru Input Layer ke *Hidden Layer*

	X_1	X_2	…	X_{12}	Bias
Z_1	0,096	0,397	…	0,098	0,191
Z_2	0,195	0,296	…	0,197	0,288
Z_3	0,294	0,196	…	0,297	0,385
Z_4	0,394	0,096	…	0,397	0,386
Z_5	0,396	0,097	…	0,398	0,289
Z_6	0,297	0,198	…	0,298	0,192
Z_7	0,198	0,299	…	0,199	0,095
Z_8	0,098	0,399	…	0,099	0,095
Z_9	0,096	0,397	…	0,098	0,191

	X_1	X_2	…	X_{12}	Bias
Z_{10}	0,394	0,196	…	0,397	0,285
Z_{11}	0,295	0,196	…	0,297	0,187
Z_{12}	0,198	0,299	…	0,199	0,396

Begitu juga mengenai nilai bobot dan bias pada *hidden layer* ke *output layer* dapat dilihat pada Tabel 6.16 berikut:.

$$W_1 = 0{,}200 - 0{,}194 = \mathbf{0{,}06}$$

Tabel 6.16: Bobot dan Bias Baru Hidden Layer ke Output Layer

	Z_1	Z_2	…	Z_{12}	Bias
Y	0,006	0,099	…	-0,103	-0,153

7. Setelah memperoleh bobot baru maka bobot baru tersebut akan menjadi bobot lama untuk perhitungan selanjutnya. Lalu lakukan hitung maju dengan menggunakan data selanjutnya. Perhitungan disini hanya menampilkan perhitungan data pertama pada *epoch* pertama serta perhitungan MSE pada *epoch* pertama saja.
8. Menghitung MSE.

   Jika sudah mendapat nilai dari lapisan keluaran maka dapat dihitung error dan MSE. Hitung error dan MSE

   $$Error = 0{,}272 - 0{,}924 = \mathbf{-0{,}52}$$

   $$MSE = (-0{,}652)^2 / 1 = \mathbf{0{,}26}$$

9. Setelah pengecekan MSE, lakukan pengecekan apakah MSE baru ≤ MSE lama. Jika ya, lakukan kembali perhitungan tahap 1 hingga tahap 9 dengan parameter Marquardt dibagi dengan faktor Tau.. Jika tidak, lakukan kembali perhitungan tahap 7 hingga tahap 9 dengan parameter Marquardt dikali dengan faktor Tau.
10. Proses pelatihan berhenti jika error = target error ataupun epoch ≥ epoch maksimal sehingga bobot akhir diperoleh pada tahap ini. Data yang dihitung pada perhitungan manual ini hanya data pertama saja.

Pengujian yang terdiri dari analisa algoritma dan pengujian terhadap algoritma pelatihan, dapat dilakukan mengikuti tahapan algoritma *backpropagation* yang telah dijelaskan di bab sebelumnya.

Bab 7
Prediksi dengan Quantum Neural Network

7.1 Pendahuluan

Quantum Neural Network (QNN) merupakan jaringan saraf klasik digunakan untuk menghubungkan neuron dan memilih nilai input, bobot dan nilai output pada setiap lapisan yang ada sehingga fungsinya dapat diminimalkan dengan range nilai antara 0 dan 1 yang pada algoritma quantum dikonversi ke dalam bilangan quantum bit(Gultom, 2017). Setiap nilai (input, bobot, output) yang terdapat pada neuron disetiap lapisan diaktivasi dengan menggunakan fungsi sigmoid atau fungsi softmax. Umumnya Algoritma ini juga dapat digunakan untuk pengenalan pola, Klasifikasi data dan Prediksi dibidang Ekonomi seperti Prediksi harga saham dan bidang lainnya(Kopczyk, 2018). Pada umumnya, Implementasi quantum neural network yang merupakan bagian pembelajaran machine learning dengan menggunakan konsep Quantum Neural Network sehingga pembelajaran dapat pengujian dapat lebih cepat. Pada pembahasan kali ini, penulis akan Memprediksi Hasil Pertandingan Sepak Bola dengan Menggunakan Algoritma Quantum Neural Network.

7.2 Menentukan Arsitektur Jaringan

Dalam menentukan arsitektur pada quantum neural network konsep yang digunakan adalah menggunakan pendekatan terhadap jaringan saraf klasik. Sebelum menentukan artitektur, Ada beberapa poin yang perlu diperhatikan sehingga hasil yang diperoleh dapat lebih efisien(Lubis, 2017).

Untuk menentukan jumlah neuron pada arsitektur yang digunakan penulis menggunakan aturan "rule of thumb" sebagai berikut (Lipantri Mashur Gultom, 2015):

1. Jumlah neuron pada lapisan tersembunyi harus di antara jumlah neuron pada lapisan masukan dan lapisan keluaran.
2. Jumlah neuron pada lapisan tersembunyi harus 2/3 jumlah neuron lapisan masukan, ditambah jumlah neuron lapisan keluaran.
3. Jumlah neuron pada lapisan tersembunyi harus kurang dari dua kali jumlah neuron pada lapisan masukan.

Berdasarkan aturan di atas, kita bisa menentukan neuron pada arsitektur yang tetapkan. Langkah yang dilakukan adalah sebagai berikut:

1. Menetapkan parameter sebagai variable input

Pada pembahasan ini, variable yang digunakan pada kasus prediksi hasil pertandingan sepakbola adalah sebagai berikut (Lubis and Parlina, 2019):

Tabel 7.1: Parameter Prediksi pada Prediksi Pertandingan Sepakbola

No	Parameter Prediksi	Simbol
1	Rata-rata Point 5 Pertandingan Kandang	x1
2	Rata-rata Point 5 Pertandingan Tandang	x2
3	Jumlah Point Tim Home	x3
4	Jumlah Point Tim Away	x4
5	Poin pada 5 Laga Terakhir Tandang	x5
6	point pada 5 Laga Terakhir Kandang	x6
7	Jumlah Pertandingan	x7
8	Jumlah Minggu	x8

Jumlah variable pada kasus prediksi hasil pertandingan sepakbola adalah 8 kriteria yaitu: x1, x2, x3, x4, x5, x6, x7 dan x8.

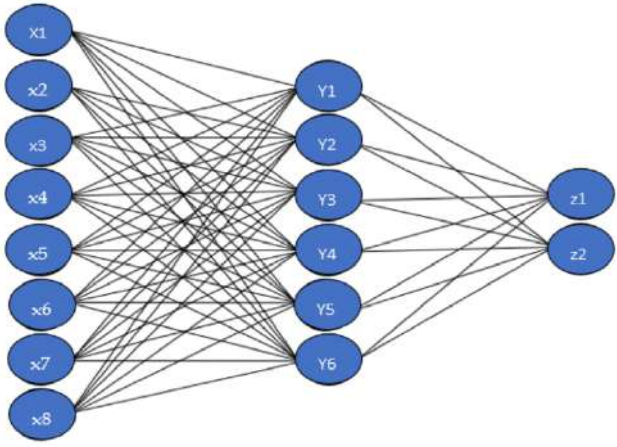
2. Menentukan neuron pada hidden layer

Dasar menentukan hidden layer adalah dengan menggunakan aturan "rules of thumb" yaitu jumlah neuron harus di antara variable input dan variable output. Jumlah neuron pada hidden layer adalah 6 buah neuron dengan nama y1, y2, y3, y4, y5 dan y6.

3. Menetapkan jumlah neuron pada output layer

Pada kasus ini, jumlah variable output ditentukan sebanyak 2 variabel dengan nama variable z1 dan z2.

Dapat disimpulkan bahwa arsitektur yang digunakan pada kasus ini adalah Arsitektur 8-6-2. Berikut prakiraan arsitektur yang digunakan pada kasus prediksi hasil pertandingan sepakbola:



Gambar 7.1: Arsitektur 8-6-2 Quantum Neural Network

Catatan:

Pada arsitektur di atas merupakan arsitektur yang digunakan pada permasalahan ini, untuk selanjutnya dapat dikembangkan dengan aturan yang ada. Berikut ini adalah dataset yang digunakan untuk memprediksi hasil pertandingan sepakbola dengan Quantum Neural Network

Tabel 7.2: Dataset Prediksi Pada Quantum Neural Network

No.	Parameter Dataset									
	Input								Output	
	x1	x2	x3	x4	x5	x6	x7	x8	z1	z2
1	6.4	0.2	1.9	0.4	3.4	3	38	38	0	0
2	6.4	9.4	1.9	0.4	3.4	9.4	38	38	0	2
3	3.2	9.2	0.4	1.9	0.2	12	38	38	0	2
4	0.6	9.2	1.1	1.9	3.4	12	38	38	0	2
5	6.2	9.2	1.3	1.9	1	12	38	38	2	0
6	3.2	9.2	1.1	1.9	3.2	12	38	38	1	2
7	9.2	9.2	1.2	1.9	0.4	12	38	38	0	2
8	6.4	9.2	2.1	1.9	3.8	12	38	38	2	5
9	6.4	6	1.6	1.3	3.6	1	38	38	0	0
10	6.4	0.6	1.6	1.1	3.6	3.6	38	38	2	2
11	6.4	9.4	1.6	0.4	3.6	9.4	38	38	0	3
12	12	12	1.7	1.7	6	3.2	38	38	0	1
13	0.4	7	1.0	1.7	6.4	6.4	38	38	0	0
14	0	6.2	0.9	1.6	6.2	9.4	38	38	4	5
15	0	7	0.9	1.7	6.2	6.4	38	38	0	0
16	6.4	3.8	1.7	1.2	12	0.4	38	38	0	3
17	12	6.2	1.7	2.1	0.6	3.6	38	38	2	2

No.	Parameter Dataset									
	Input								Output	
	x1	x2	x3	x4	x5	x6	x7	x8	z1	z2
18	6.2	6.2	1.1	2.1	6.4	9.4	38	38	2	3

1. Menentukan Nilai Input, Bobot, Output dan Learning rate

Dalam menentukan nilai input, bobot dan output dengan model quantum neural network menggunakan pendekatan komputasi quantum dengan tujuan memberikan konsep yang lebih efisien dari jaringan saraf klasik.

a. Menentukan nilai input

Berdasarkan kriteria yang digunakan, nilai input pada dataset prediksi memiliki nilai antara 0 sampai 1. Proses selanjutnya adalah melakukan proses komputasi dengan menggunakan konsep transformasi data menggunakan rumus sebagai berikut:

$$X^1 = \frac{(X-Xmin)(b-\alpha)}{(Xmax-Xmin)} + \alpha \quad \text{..... 1}$$

Contoh perhitungan transformasi pada variable x1 pada tabel 7.1

Tabel 7.3: Contoh Perhitungan Transformasi Data

X	:	6,4
Xmin	:	0
Xmax	:	12
b	:	0,9
α	:	01
Hasil	:	0,5

Dari contoh perhitungan transformasi pada tabel 7.3, berikut ini adalah tabel hasil transformasi secara keseluruhan pada dataset yang digunakan:

Tabel 7.4: Dataset Setelah Transformasi Data

X1	X2	X3	X4	X5	X6	X7	X8
0,5	0,1	0,82	0,1	0,3	0,3	0	0
0,5	0,7	0,82	0,1	0,3	0,7	0	0
0,3	0,7	0,22	0,8	0,1	0,9	0	0
0,1	0,7	0,5	0,8	0,3	0,9	0	0
0,5	0,7	0,58	0,8	0,2	0,9	0	0
0,3	0,7	0,5	0,8	0,3	0,9	0	0
0,7	0,7	0,54	0,8	0,1	0,9	0	0
0,5	0,7	0,9	0,8	0,3	0,9	0	0
0,5	0,5	0,7	0,5	0,3	0,1	0	0
0,5	0,1	0,7	0,4	0,3	0,3	0	0
0,5	0,7	0,7	0,1	0,3	0,7	0	0
0,9	0,9	0,74	0,7	0,5	0,3	0	0
0,1	0,6	0,1	0,7	0,5	0,5	0	0
0,1	0,5	0,42	0,7	0,5	0,7	0	0
0,1	0,6	0,42	0,7	0,5	0,5	0	0
0,5	0,3	0,74	0,5	0,9	0,1	0	0
0,9	0,5	0,74	0,9	0,1	0,3	0	0
0,5	0,5	0,5	0,9	0,5	0,7	0	0

b. Menentukan nilai bobot

Konsep dalam menentukan bobot tetap menggunakan bilangan biner (antara 0 dan 1), tetapi perlu diperjelas bahwa konsep yang paling penting alah melakukan konversi ke bentuk quantum bit dalam bentuk matriks dengan ordo 8X6X2. Pendekatan lainnya pun sama yaitu menentukan nilai quantum bit (qubit) ditentukan secara random.

Sebagai contoh, pada arsitektur 8-6-2 diketahui bobot pada input layer adalah:

$x_1.w_1=0{,}5.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$ $X_2.w_1=$ $0{,}1.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$ $x3.w1 = 0{,}8.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} x4.w1 = 0{,}1.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + x5.w1 = 0{,}3.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} x6.w1 = 0{,}3.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} x7.w1 = 0\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}.x8.w1 = 0.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$

cara di atas digunakan untuk menghitung nilai dari input layer (x_1, x_2, x_3, x_4, x_5, x_6, x_7 dan x_8 ke hidden layer dengan jumlah arsitektur sebanyak 6 neuron (w_1, w_2, w_3, w_4, w_5, w_6) dan contoh perhitungan di atas adalah perhitungan nilai pada satu hidden saja yaitu (w_1). Hasil dari perhitungan input layer ke hidden diaktivasi dengan menggunakan sigmoid function dengan persamaan berikut:

Selanjutnya, hasil pada hidden layer kita notasikan kedalam karakter v untuk selanjutya menghitung nilai output. Cara yag dilakukan adalah seperti dibawah ini:

untuk mementukan nilai bobot pada layer output adalah:

$v_1.y_1+v_2.y_1+v_3.y_1+v_1.w_4+y_1.w_5+v_1.y_6=0{,}015.\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + 0{,}109.\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + 0{,}109.\begin{bmatrix}0 & 1\\ 0 & 1\end{bmatrix} +=0{,}015.\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix} + 0{,}002.\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix} + 0{,}119.\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}$ dan untuk hidden ke output kedua adalah:

$v_1.y_2+v_2.y_2+v_3.y_2+v_4.y_2+v_5.y2+v_6.y_2=0{,}015.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}+0{,}015.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + 0{,}109\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + 0{,}109.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + 0{,}015\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + 0{,}002.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + 0{,}119.\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$

a. Menentukan nilai learning rate
   NIlai learning rate ditentukan secara random yang berkisar antara 0 dan 1.

b. Menentukan nilai output

Dalam menentukan hasil, konsep nya adalah nilai output harus mendekati target yang ditentukan atau Y=T. artinya, jika hasil sudah mendekati target (nilai target

minimal=0 dan maksimal=1) maka pengujian selesai karena sudah mencapai konvergensi. Jika belum mencapai target atau belum mendekati, maka perlu dilakukan perubahan nilai bobot dengan ketentuan sebagai berikut:

Output pada kasus ini adalah dengan dua variabel, yang merupakan perhitungan dari hidden layer dengan 6 buah neuron yang terhubung dengan layer output dengan 2 variabel hasil. Kita notasikan nilai output dengan variabel Y (Seperti pada penjelasan sebelumnya).

Berikut ini data target yang akan menjadi perbandingan dengan nilai output:

Tabel 7.5: Nilai Target pada Pengujian

No	Skor	Target
1	0	0-0,10
2	1	0,11-0,26
3	2	0,27-0,42
4	3	0,43-0,58
5	4	0,59-0,65
6	5	0,66-0,90

Pembelajaran dan Pengujian dengan Algoritma Quantum Neural Network

Dalam menentukan data pelatihan dan pengujian, ketentuan ini harus diperhatikan oleh peneliti, yaitu dataset pelatihan dengan pengujian presentasinya adalah 60 :40 atau dengan kata lain, presentasi terbanyak adalah pada data pembelajaran.

Untuk pembagiannya, pada pengujian ini jumlah dataset terdiri dari 18 record. Dengan demikian, jumlah dataset pembelajaran sebanyak 10 baris dan 8 record untuk dataset pengujian. Berikut pembagian dataset untuk pelatihan dan pengujian.

2. Dataset Pembelajaran

Tabel 7.6: Dataset Pembelajaran

No	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
1	1	0	0,82	0	0	0	0	0	0,1	0,1
2	1	1	0,82	0	0	1	0	0	0,1	0,4

No	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
3	0	1	0,22	1	0	1	0	0	0,1	0,4
4	0	1	0,5	1	0	1	0	0	0,1	0,4
5	1	1	0,58	1	0	1	0	0	0,2	0,1
6	0	1	0,5	1	0	1	0	0	0,2	0,4
7	1	1	0,54	1	0	1	0	0	0,1	0,4
8	1	1	0,9	1	0	1	0	0	0,2	0,9
9	1	0	0,7	1	0	0	0	0	0,1	0,1
10	1	0	0,7	0	0	0	0	0	0,2	0,4

Hasil dari dataset pembelajaran dengan Quantum Neural Networks (QNN) adalah sebagai berikut:

Tabel 7.7: Dataset Pembelajaran

NO	v1	v2	v3	v4	v5	v6	y1	y2
1	0,015	0,109	0,109	0,015	0,002	0,119	0,324	0,405
2	0,002	0,043	0,043	0,002	0,000	0,119	0,397	0,447
3	0,002	0,047	0,047	0,002	0,000	0,231	0,340	0,417
4	0,001	0,036	0,036	0,001	0,000	0,401	0,279	0,383
5	0,001	0,024	0,024	0,001	0,000	0,119	0,416	0,458
6	0,001	0,029	0,029	0,001	0,000	0,231	0,358	0,427
7	0,001	0,024	0,024	0,001	0,000	0,057	0,447	0,473
8	0,000	0,016	0,016	0,000	0,000	0,119	0,424	0,462
9	0,005	0,069	0,069	0,005	0,000	0,119	0,369	0,432
10	0,010	0,091	0,091	0,010	0,001	0,119	0,344	0,418

Pada dataset pembelajaran yang terdiri dari 10 baris data yang ditentukan 60% dari 18 baris dataset. Pada hasil pembelajaran yang dilakukan dengan menghubungkan neuron pada hidden layer dengan jumlah 6 titik ke layer output dengan jumlah variabel sebanyak 2 (Y_1 dan Y_2) dan hasil output yang diperoleh setelah diaktivasi dengan menggunakan fungsi sigmoid.

$$S(x)=\frac{1}{\exp\ ^x} \quad \text{.....} 2$$

Hasil pada dataset pembelajaran adalah setelah nilai hidden atau output di aktivasi dengan persamaan 2. Dengan dataset pembelajaran tersebut kemudian menyesuaikan dengan data target yang telah ditentukan pada tabel 7.4 untuk dilakukan perubahan nilai bobot apabila selisih antara nilai output dengan nilai target belum konvergensi.

Berikut perhitungan Mean Square Error (MSE) pada dataset pembelajaran:

Tabel 7.8: Mean Square Error (MSE) Pada Dataset Pembelajaran

Y_1	Y_2
-0,222	-0,060
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000

Pada tabel 7.8 selisih antara nilai target dan output yang dataset pembelajaran merupakan nilai error dengan menghitung selisih nilai target dengan variabel output (T-Y), kemudian selisih keduanya dibagi dengan banyaknya dataset

pembelajaran sehingga menghasilkan MSE (Mean Square Error) pada dataset pembelajaran. Rata-rata error pada y1=-0,002, dan y2=-0,006

a. Dataset Pengujian

Tabel 7.9: Dataset Pengujian

No	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	Y_1	Y_2
11	1	1	0,7	0	0	1	0	0	0,1	0,6
12	1	1	0,74	1	0	0	0	0	0,1	0,3
13	0	1	0,1	1	1	1	0	0	0,1	0,1
14	0	1	0,42	1	1	1	0	0	0,4	0,9
15	0	1	0,42	1	1	1	0	0	0,1	0,1
16	1	0	0,74	0	1	0	0	0	0,1	0,6
17	1	1	0,74	1	0	0	0	0	0,2	0,4
18	1	1	0,5	1	1	1	0	0	0,2	0,6

Hasil dari dataset pengujian dengan quantum nerural networks (QNN) adalah sebagai berikut:

Tabel 7.10: Dataset Pelatihan

No	**v_1**	**v_2**	**v_3**	**v_4**	**v_5**	**v_6**	**y_1**	**y_2**
11	0,001	0,024	0,024	0,001	0,000	0,018	0,466	0,483
12	0,001	0,023	0,023	0,001	0,000	0,018	0,467	0,483
13	0,000	0,016	0,016	0,000	0,000	0,500	0,256	0,370
14	0,000	0,012	0,012	0,000	0,000	0,500	0,260	0,372
15	0,000	0,012	0,012	0,000	0,000	0,500	0,260	0,372
16	0,004	0,061	0,061	0,004	0,000	0,018	0,427	0,462
17	0,001	0,023	0,023	0,001	0,000	0,018	0,467	0,483
18	0,000	0,004	0,004	0,000	0,000	0,018	0,487	0,493

Pada dataset pelatihan, jumlah baris data presentasi nya adalah 40% dari banyak dataset dengan jumlah 8 baris data. Pada proses pembelajaran dengan quantum neural networks hasil atau output yang diperoleh dengan menghubungkan neuron pada hidden layer dengan jumlah 6 titik ke layer output dengan jumlah variabel sebanyak 2 dan hasil output yang diperoleh sudah diaktivasi dengan

menggunakan *sigmoid function*. Dengan dataset pelatihan tersebut kemudian menyesuaikan dengan data target yang telah ditentukan pada tabel 7.5 untuk dilakukan perubahan nilai bobot apabila selisih antara nilai output dengan nilai target belum konvergensi.

Berikut perhitungan MSE pada dataset pelatihan:

Tabel 7.11: Mean Square Error (MSE) Pada Dataset Pelatihan

Y_1	Y_2
-0,046	0,012
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000
0,000	0,000

Pada tabel 7.11 selisih antara nilai target dan output yang dataset pelatihan merupakan nilai error dengan menghitung selisih nilai target dengan output (t-y), kemudian selisih dibagi dengan banyaknya dataset pelatihan sehingga menghasilkan MSE (Mean Square Error) pada dataset pelatihan. Rata-rata error pada y_1=-0,006, dan y_2=0,002.

Untuk formula perubahan bobot, gunakan persamaan di bawah ini:

$$W_{baru}=W_{lama} + \alpha\,(Y-t).X \quad \text{..... 2}$$

Catatan:

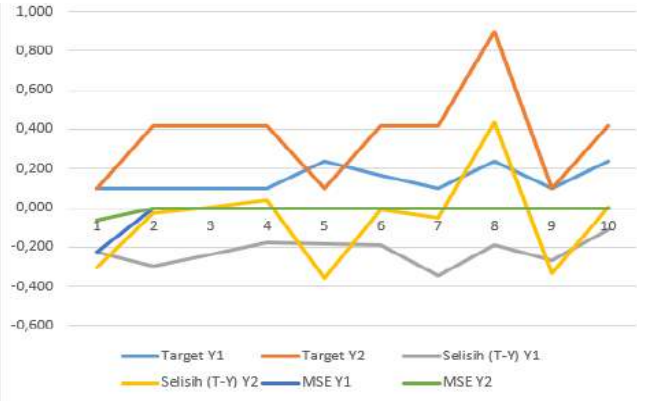
Persamaan 2 di atas digunakan apabila nilai output tidak sama atau belum mendekati nilai target. Simbol α merupakan learning rate dengan nilai antara 0,1 sampai 0,9.

7.3 Hasil dan Evaluasi

Hasil dari pembelajaran dan pelatihan dengan menggunakan Quantum Neural Networks ditampilkan grafik sebagai berikut:

a. Grafik Dataset Pembelajaran

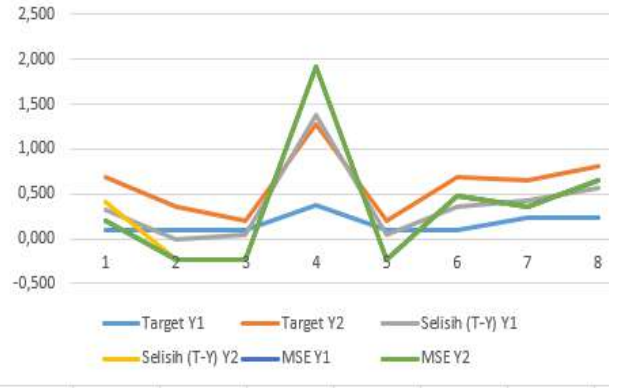
Grafik dataset pembelajaran digunakan untuk menampilkan nilai output, nilai selisih atau nilai error dan rata-rata nilai error dari dataset pembelajaran.



Gambar 7.2: Grafik Hasil Pembelajaran pada Quantum Neural Networks

Pada Grafik pembelajaran hasil yang diperoleh adalah nilai error antara output dengan nilai target pada y_1 dan y_2 lebih kecil dari 0 (Y<=0). Dengan nilai error y1=-0,022 dan y2=-0,006.

b. Grafik Dataset Pelatihan



Gambar 7.3: Grafik Hasil Pelatihan pada Quantum Neural Networks

Pada Grafik pelatihan dengan banyak data 8 record, hasil yang diperoleh adalah nilai error antara output dengan nilai target pada y1 dan y2 lebih kecil dari 0 (Y<=0). Dengan nilai error y1=-0,027 dan y2=-0,000.

Bab 8

Penerapan Quantum Perceptron Dalam Klasifikasi Data

8.1 Komputasi Quantum

Komputasi quantum adalah teknologi yang muncul memanfaatkan efek dari kuatum untuk mencapai signifikan dalam beberapa case eksponensial dan peningkatan algoritma dibandingkan algoritma sebelumnya (Wiebe, Kapoor and Svore, 2016). Beberapa tahun belakang banyak para peneliti melakukan penelitian di bidang mesin pembelajaran menggunakan komputasi quantum (Amin et al., 2018). Komputasi kuantum dan jaringan saraf adalah bidang penelitian multidisiplin. Komputasi kuantum menggunakan dari konsep-konsep dari fisika, matematika, dan computer sains (da Silva, Ludermir and de Oliveira, 2016).

Fenomena partikel pada mekanika quantum sangat menginspirasi teori komputasi quantum. Fenomena di mana sebuah partikel dalam mekanika quantum dapat memiliki dua keadaan sekaligus disebut dengan superposisi. Superposisi dari sebuah pertikel ini jika ditransformasikan kedalam komputasi dapat berupa bit di mana nilai dari bit hanya terdiri dari 0 atau 1. Nilai bit komputasi quantum dapat berupa 0 atau 1 atau kombinasi dari kedua nilai. Pada komputasi quantum dikenal quantum bit (qubit) yang merupakan basis bilangan terkecil. Qubit memiliki dua keadaan yang disimbolkan dengan keadaan |0> dan keadaan |1> sedangkan tanda "| >" dikenal dengan notasi Dirac's. Notasi "< |" disebut ket dan "| >" disebut bra (Kopczyk, 2018).

Elemen dasar komputasi kuantum adalah qubit, vektor satuan kompleks dalam ruang Hilbert 2 dimensi H2. Berlabel | 0i dan | 1i, kedua status kuantum menyatakan satu bit informasi: | 0i sesuai dengan bit 0 komputer klasik, dan | 1i ke bit 1. Lebih formal, diberi dasar {| 0i, | 1i}, qubit dasar diberikan dalam notasi Dirac dan dijelaskan oleh fungsi gelombang (Seow, Behrman and Steck, 2015).

8.2 Quantum Neural Network

Quantum Neural Network (QNN) merupakan salah satu bentuk dari jaringan saraf tiruan yang menggunakan konsep komputasi quantum (Daskin, 2018), (Ying, Ying and Feng, 2017). Dalam metode ini bentuk arsitektur jaringan masih menggunakan jaringan saraf klasik akan tetapi dari penentuan input, bobot, algoritma pembelajaran dan target sudah menggunakan pendekatan komputasi quantum (Farhi and Neven, 2018). Tujuan utama dari penggabungan dua metode ini ialah untuk memberikan konsep yang lebih efesien dari jaringan saraf klasik.

Dalam jaringan saraf klasik, kombinasi linear dari input parameter dengan bobot berbeda dimasukkan ke dalam beberapa neuron. Output dari setiap neuron ditentukan oleh aktivasi fungsi seperti yang berikut ini:

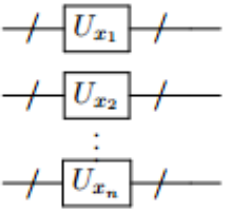
$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Fungsi aktivasi nonlinier seperti hiperbolik dan sigmoid fungsi lebih umum digunakan untuk membuat output neuron yang lebih halus, misalnya perubahan kecil dalam berat badan apa pun menyebabkan perubahan kecil pada output. Itu juga telah diperdebatkan bahwa fungsi aktivasi berkala dapat meningkatkan fungsi umum kinerja jaringan saraf dalam aplikasi spesifik(Daskin, 2019).

Mari kita asumsikan bahwa parameter input xj diharapkan dilihat dengan k jumlah bobot berbeda $\{W_{j1}, \ldots, W_{jk}, \}$ pada jaringan. Untuk setiap input, kami akan membuat yang berikut ini operator untuk mewakili perilaku input dari parameter X_j,.:

$$U_{x_j} = \begin{pmatrix} e^{iw_{j1}x_j} & & & \\ & e^{iw_{j2}x_j} & & \\ & & \ddots & \\ & & & e^{iw_{jk}x_j} \end{pmatrix}$$

Untuk setiap input Uxj, kita perlu menggunakan nomor log2k qubits. Oleh karena itu, parameter n-input mengarah ke n jumlah U_{xj},. dan membutuhkan jumlah qubit log2kn secara total, misalnya:



Gambar 8.1: Quantum Circuit

8.3 Quantum Perceptron

Komputasi kuantum adalah teknologi yang muncul yang memanfaatkan efek kuantum untuk mencapai signifikan, dan dalam beberapa case eksponensial, peningkatan algoritma lebih dari rekan-rekan klasik mereka. Semakin pentingnya mesin pembelajaran dalam beberapa tahun terakhir menyebabkan sejumlah studi yang menyelidiki kemampuan komputer kuantum untuk machine learning.

Sementara sejumlah *speed up* kuantum penting telah ditemukan, sebagian besar *speed up* ini disebabkan oleh mengganti sub rutin klasik dengan algoritma kuantum yang setara meskipun lebih cepat. Potensi sebenarnya dari kuantum. Oleh karena itu, algoritma mungkin tetap kurang dieksploitasi karena algoritma kuantum telah dikonfigurasikan untuk mengikuti hal yang sama metodologi di balik metode pembelajaran mesin tradisional. Di sini kami mempertimbangkan pendekatan alternatif, kami merancang algoritma pembelajaran mesin baru yang disesuaikan dengan speed up yang dapat diberikan komputer kuantum.

Kami menggambarkan pendekatan kami dengan fokus pada pelatihan perceptron. Perceptron adalah blok bangunan mendasar untuk berbagai model pembelajaran mesin termasuk jaringan saraf dan mesin vektor pendukung. Tidak seperti banyak lainnya algoritma pembelajaran mesin, batas yang ketat dikenal untuk kompleksitas komputasi dan statistik tradisional pelatihan perceptron. Karenanya, kami dapat menunjukkan dengan ketat peningkatan

kinerja yang berbeda baik menggunakan komputer kuantum untuk meningkatkan pelatihan perceptron tradisional atau dari merancang bentuk baru perceptron pelatihan yang sejalan dengan kemampuan komputer kuantum. Kami menyediakan dua pendekatan kuantum untuk pelatihan perceptron. Pendekatan pertama berfokus pada aspek komputasi masalah dan metode yang diusulkan secara kuadratik mengurangi skala kompleksitas pelatihan ke jumlah vektor pelatihan. Algoritma kedua berfokus pada efisiensi statistic (Wiebe, Kapoor and Svore, 2016).

Quantum Perceptron merupakan bagian dari metode jaringan saraf tiruan kuantum (Quantum Neural Network) yang menggabungkan komputasi kuantum dengan algoritma perceptron pada jaringan saraf tiruan (Daskin, 2018). Konsep dasar dari metode Quantum Perceptron ini yaitu menerapkan sifat atom dalam mekanika kuantum yang disebut dengan quantum bit (qubit) pada komputasi kuantum. Qubit memiliki keadaan yang berbeda-beda pada satu waktu tertentu serta memiliki nilai probabilitas yang berbeda-beda pula (superposisi).

8.4 Penerapan Quantum Perceptron Dalam Kalsifikasi Data

Ada beberapa masalah yang dapat diselesaikan jaringan saraf tiruan yaitu : klasifikasi (Gonçalves, 2016), (Rebentrost, Bromley and Weedbrook, 2018), prediksi (Killoran et al., 2018), pengenalan pola (Türkpençe, Akıncı and Serhat, 2018), (Torrontegui and Ripoll, 2018) dan optimasi (Seow, Behrman and Steck, 2015), (Schuld and Sinayskiy, 2014), (Ohzeki et al., 2018).

8.4.1 Data Yang Digunakan

Data yang digunakan adalah dataset dari UCI Machine learning repository (http://archive.ics.uci.edu/ml). Repository ini dataset yang berkaitan dengan beberapa penelitian seperti Machine learning, soft computing dan data mining. Data yang diguanakan dataset lenses dan balance scale. Dataset akan dibagi menjadi dua bagian yaitu dataset untuk pembelajaran dan dataset untuk pengujian (Gultom, 2017). Tabel 8.1 Berikut ini struktur dataset yang digunakan.

Tabel 8.1: Struktur dataset

Nama	Jumlah Atribut	Jumlah Kelas	Jumlah Data	Keterangan
Lenses	*4 atribut :* *1) Age of the patient: young, prepresbyopic, presbyopic* *2) Spectacle prescription : myope, hypermetrope* *3) Astigmatic: no, yes* *4) Tear production rate: reduced, normal*	*3 kelas :* *1) The patient should be fitted with hard contact lenses* *2) Tehe patient should be fitted with soft contact lenses,* *3) Tehe patient should not be fitted with contact lenses*	24	Data klasifikasi penggunaan lensa kontak

8.4.2 Tahapan Analisis

Pada tahapan ini dibagi menjadi beberapa tahap, yaitu :

1. Menenetukan arsitektur jaringan

Tahapan ini adalah menentukan arsitektur jaringan quantum perceptron.

2. Bobot dan learning rate

Dalam tahap ini dilakukan pembentukan entanglement dari qubit kemudian dilakukan pengukuran dari entanglement.

3. Pembelajaran algoritma quantum perceptron

Tahap ini merupakan tahap pembelajaran dari dataset yang telah ditransformasikan ke dalam qubit dengan entanglement dan fitur kunci yang ada.

4. Pengujian quantum perceptron

Tahap ini dilakukan untuk mengukur persentase tingkat keberhasilan proses pengenalan pola dari dataset yang ada

5. Evaluasi

Tahap ini dilakukan pengukuran konvergensi dari hasil tahap pembelajaran dan pengujian yang sebeleumnya.

8.4.3 Preprocessing Data

Dataset klasifikasi penggunaan lensa kontak memeiliki beberapa atribut sebagai berikut :

1. Age of the patient : young(1), pre-presbyopic (2), presbyopic(3)
2. Spectacle prescription : myope (1), hypermetrope(2)
3. Atigmatic : no(1), yes(2)
4. Tear production rate : reduced (1), normal (2)

Klasifikasi kelas terdiri dari :

1. Hard contact lenses
2. Soft cobtact lenses
3. No contact lenses

Dataset ini terdiri dari 24 sampel dengan distribusi kelas yaitu : 4 sampel pada kelas *hard contact lenses*, 5 sampel pada *soft contact lenses* dan 15 sampel pada *no contact lenses*. Tabel 8.2 berikut ini dataset lengkap klasifikasi penggunaan lensa kontak :

Tabel 8.2: Dataset klasifikasi penggunaan lensa kontak

No	*Age of the patient*	*Spectacle prescription*	*Astigmatic*	*Tear production rate*	*Kelas*
1	1	1	2	2	1
2	1	2	2	2	1
3	2	1	2	2	1
4	3	1	2	2	1
5	1	1	1	2	2
6	1	2	1	2	2
7	2	1	1	2	2
8	2	2	1	2	2
9	3	2	1	2	2
10	1	1	1	1	3
11	1	1	2	1	3
12	1	2	1	1	3
13	1	2	2	1	3
14	2	1	1	1	3
15	2	1	2	1	3
16	2	2	1	1	3
17	2	2	2	1	3
18	2	2	2	2	3
19	3	1	1	1	3
20	3	1	1	1	3

No	*Age of the patient*	*Spectacle prescription*	*Astigmatic*	*Tear production rate*	*Kelas*
21	3	1	2	1	3
22	3	2	1	1	3
23	3	2	2	1	3
24	3	2	2	2	3

Sebelum dilakukan pengujian, dataset klasifikasi lensa kontak tersebut dikodekan ke dalam bentuk binari sebagai berikut:

1. Pengkodean ke dalam binari untuk *atribut age of the patient*

Tabel 8.3: Kode atribut age of the patient

Kode	**Nama**
01	*Young*
10	*Pre-presbyopic*
11	*Presbyopic*

2. Pengkodean ke dalam binari untuk atribut *spectacle prescription*:

Tabel 8.4: Kode atribut *spectacle prescription*

Kode	**Nama**
0	Myope
1	Hypermetrope

3. Pengkodean ke dalam binari untuk *atribut astigmatic*

Tabel 8.5: Kode atribut astigmatic

Kode	Nama
0	No
1	Yes

4. Pengkodean ke dalam binari untuk atribut tear production rate.

Tabel 8.6: Kode atribut tear production rate

Kode	**Nama**
0	Reduced
1	Normal

5. Pengkodean kedalam binari untuk klasifiksi kelas :

Tabel 8.7: Kode klasifikasi kelas

Kode	**Nama**
01	Hard contact lenses
10	Soft contact lenses
11	No contact lenses

Kemudian dataset di rubah kedalam kode binari seperti pada tabel 8.8 berikut ini :

Tabel 8.8: Dataset klasifikasi lensa kontak dalam kode binari

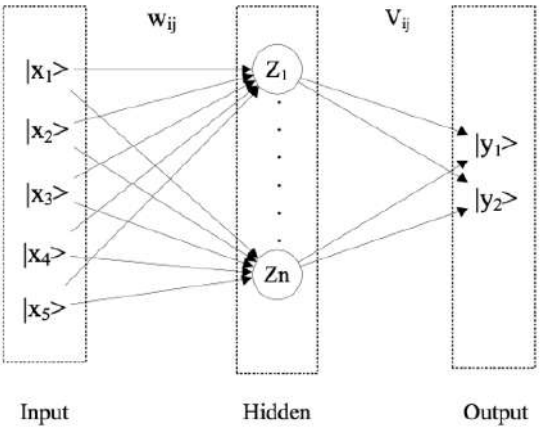
No	**Age of the patient**	**Spectacle prescription**	**Astigmatic**	**Tear production rate**	**Kelas**
1	01	0	1	1	01
2	01	1	1	1	01

No	Age of the patient	Spectacle prescription	Astigmatic	Tear production rate	Kelas
3	10	0	1	1	01
4	11	0	1	1	01
5	01	0	0	1	10
6	01	1	0	1	10
7	10	0	0	1	10
8	10	1	0	1	10
9	11	1	0	1	10
10	01	0	0	0	11
11	01	0	1	0	11
12	01	1	0	0	11
13	01	1	1	0	11
14	10	0	0	0	11
15	10	0	1	0	11
16	10	1	0	0	11
17	10	1	1	0	11
18	10	1	1	1	11
19	11	0	0	0	11
20	11	0	0	1	11
21	11	0	1	0	11
22	11	1	0	0	11

No	Age of the patient	Spectacle prescription	Astigmatic	Tear production rate	Kelas
23	11	1	1	0	11
24	11	1	1	1	11

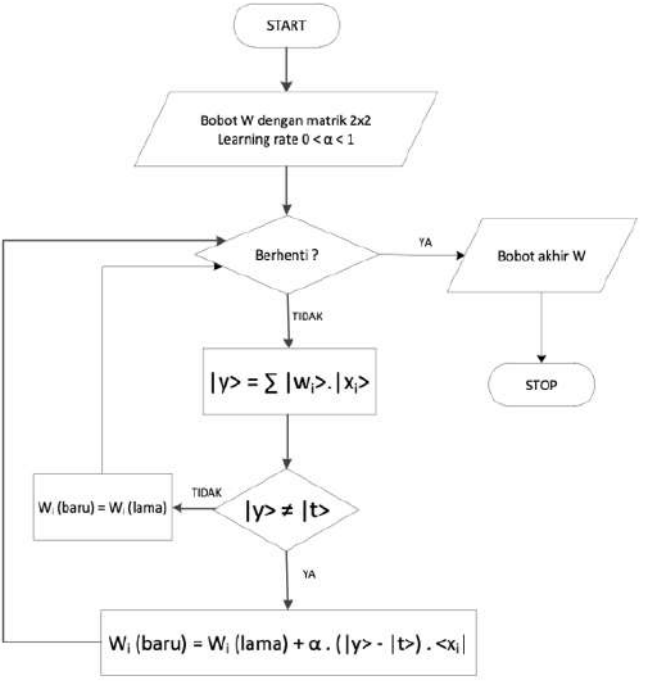
Misalkan diambil sampel dari dataset nomor 1 yaitu kode binari 0101101 maka kode tersebut memiliki arti yaitu : atribut age of patient = young, spectacle prescription = Myope, astigmatic, = yes, tear production rate = yes dan kelas soft contact lenses.

Bentuk arsitektur jaringan yang digunakan yaitu 5-N-2 dapat dilihat pada gambar 8.2. 5 input node mewakili pengkodean atribut age of the patient, spectacle prescription, astigmatic dan tear production rate. Sedangkan 2 output node mewakili pengkodean kelas hard contact lenses, soft contact lenses dan no contact lenses. Sedangkan jumlah hidden node N menggunakan 'rule of thumb'.



Gambar 8.2: Arsitektur jaringan klasifikasi penggunaan lensa kontak

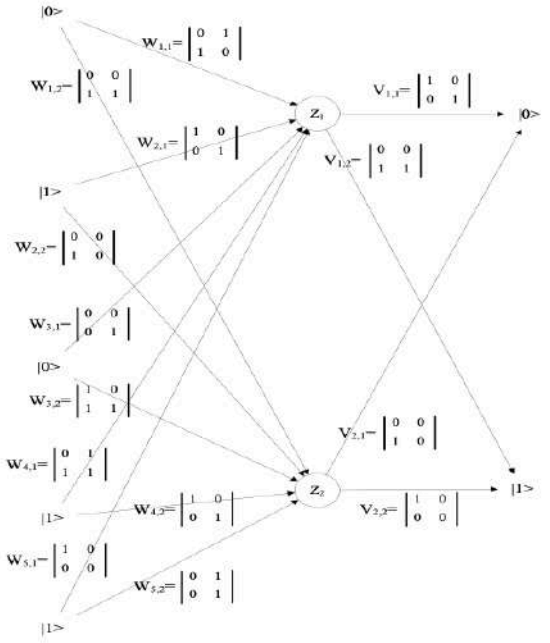
Pada pengujian ditentukan jumlah hidden node, kategori tersebut berdasarkan 'rule of thumb'. Sedangkan learning rate menggunakan 0,1. Algoritma pembelajaran quantum perceptron dilihat pada gambar 8.3 :



Gambar 8.3: Flowchart algoritma pembelajaran perceptron quantum

Pembelajaran Quntum Perceptron

Berikut ini akan dipaparkan tahapan pembelajaran perceptron dengan sebuah contoh yaitu menggunakan arsitektur 5-2-2 dan dataset penggunaan lensa kontak. Pertama sekali bobot w dan v diberi nilai random {0,1} sebagai berikut $W1,1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $W1,2 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$, $W2,1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $W2,2 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$, $W3,1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$, $W3,2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, $W4,1 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$, $W4,2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $W5,1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $W5,2 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$, $V1,1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $V1,2 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$, $V2,1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$, $V2,2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, sedangkan nilai learning rate yang dicoba yaitu 0,1. Kemudian proses pembelajaran dimulai dari data ke-1 dengan dataset nomor 1 yaitu 01011 sebagai input dan 01 sebagai output yang diharapkan. Untuk lebih lengkapnya dapat dilihat pada gambar 8.4.



Gambar 8.4: Arsitektur 5-2-2 dengan nilai input bobot dan output

Selanjutnya mencari output pada hidden Z1 dan Z2 sebagai berikut.

1) Output $Z_1 = W_{1,1}.|X_1> + W_{2,1}.|X_2> + W_{3,1}.|X_3> + W_{4,1}.|X_4> + W_{5,1}.|X_5> = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}.|0> + \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}.|1> + \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}.|0> + \begin{bmatrix}0 & 1\\1 & 1\end{bmatrix}.|1> + \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}.|1> = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}.\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}.\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}.\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0 & 1\\1 & 1\end{bmatrix}.\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}.\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}1\\3\end{bmatrix}$

2) Output $Z_2 = W_{1,2}.|X_1> + W_{2,2}.|X_2> + W_{3,2}.|X_3> + W_{4,2}.|X_4> + W_{5,2}.|X_5> = \begin{bmatrix}0 & 0\\1 & 1\end{bmatrix}.|0> + \begin{bmatrix}0 & 0\\1 & 0\end{bmatrix}.|1> + \begin{bmatrix}1 & 0\\1 & 1\end{bmatrix}.|0> + \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}.|1> +$

$$\begin{bmatrix}0 & 1\\ 0 & 1\end{bmatrix}.|1> = \begin{bmatrix}0 & 0\\ 1 & 1\end{bmatrix}.\begin{bmatrix}1\\ 0\end{bmatrix} + \begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix}.\begin{bmatrix}0\\ 1\end{bmatrix} + \begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}.\begin{bmatrix}1\\ 0\end{bmatrix} + \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}.\begin{bmatrix}0\\ 1\end{bmatrix}$$
$$+ \begin{bmatrix}0 & 1\\ 0 & 1\end{bmatrix}.\begin{bmatrix}0\\ 1\end{bmatrix} = \begin{bmatrix}0\\ 1\end{bmatrix} + \begin{bmatrix}0\\ 0\end{bmatrix} + \begin{bmatrix}1\\ 1\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix} + \begin{bmatrix}1\\ 1\end{bmatrix} = \begin{bmatrix}2\\ 1\end{bmatrix}$$

3) Output $Y_1 = V_{1,1}.|Z_1> + V_{2,1}.|Z_2> = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}.\begin{bmatrix}1\\ 3\end{bmatrix} + \begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix}.\begin{bmatrix}2\\ 1\end{bmatrix} = \begin{bmatrix}1\\ 3\end{bmatrix}$
$+ \begin{bmatrix}0\\ 2\end{bmatrix} = \begin{bmatrix}1\\ 5\end{bmatrix}$

4) Output $Y_2 = V_{1,2}.|Z_1> + V_{2,2}.|Z_2> = \begin{bmatrix}0 & 0\\ 1 & 1\end{bmatrix}.\begin{bmatrix}1\\ 3\end{bmatrix} + \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}.\begin{bmatrix}2\\ 1\end{bmatrix} = \begin{bmatrix}0\\ 4\end{bmatrix}$
$+ \begin{bmatrix}2\\ 0\end{bmatrix} = \begin{bmatrix}2\\ 4\end{bmatrix}$

Kemudian output sementara Y_1 dan Y_2 dibandingkan dengan output yang diharapkan dari $Y_1 = |0> = \begin{bmatrix}1\\ 0\end{bmatrix}$ dan $Y_2 = |1> = \begin{bmatrix}0\\ 1\end{bmatrix}$ di mana $\begin{bmatrix}1\\ 0\end{bmatrix} \neq \begin{bmatrix}1\\ 5\end{bmatrix}$ dan $\begin{bmatrix}0\\ 1\end{bmatrix} \neq \begin{bmatrix}2\\ 4\end{bmatrix}$ maka dilakukan perubahan setiap bobot $W_{i,j}$ $V_{i,j}$ dari $|X_1>$ sampai $|X_5>$ dan perhitungan nilai *error*. Pertama dilakukan perubahan bobot untuk $W_{1,1}$, $W_{2,1}$, $W_{3,1}$, $W_{4,1}$, $W_{5,1}$, $V_{1,1}$, $V_{2,1}$ pada $Y_1 \neq T_1$ sebagai berikut.

1) Bobot $W_{1,1}$ baru = $W_{1,1}$ lama + $\alpha.(|Y_1> - |T_1>).<X_1| = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} +$
$0{,}1 \,.\, \left(\begin{bmatrix}1\\ 0\end{bmatrix} - \begin{bmatrix}1\\ 5\end{bmatrix}\right) . <0| = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + 0{,}1 \,.\, \begin{bmatrix}0\\ -5\end{bmatrix}.\begin{bmatrix}1 & 0\end{bmatrix} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$
$+ 0{,}1.\begin{bmatrix}0 & 0\\ -5 & 0\end{bmatrix} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + \begin{bmatrix}0 & 0\\ -0{,}5 & 0\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0{,}5 & 0\end{bmatrix}$

2) Bobot $W_{2,1}$ baru = $W_{2,1}$ lama + $\alpha.(|Y_1> - |T_1>).<X_2| = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} +$
$0{,}1 \,.\, \left(\begin{bmatrix}1\\ 0\end{bmatrix} - \begin{bmatrix}1\\ 5\end{bmatrix}\right) . <1| = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + 0{,}1 \,.\, \begin{bmatrix}0\\ -5\end{bmatrix}.\begin{bmatrix}0 & 1\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$
$+ 0{,}1.\begin{bmatrix}0 & 0\\ 0 & -5\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & -0.5\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 0{,}5\end{bmatrix}$

3) Bobot $W_{3,1}$ baru = $W_{3,1}$ lama + $\alpha.(|Y_1\rangle - |T_1\rangle).\langle X_3|$ = $\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$ + $0{,}1 \cdot \left(\begin{bmatrix}1\\0\end{bmatrix} - \begin{bmatrix}1\\5\end{bmatrix}\right) \cdot \langle 0|$ = $\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}0\\-5\end{bmatrix} \cdot [1 \quad 0] = \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$ $+ 0{,}1 \cdot \begin{bmatrix}0 & 0\\-5 & 0\end{bmatrix} = \begin{bmatrix}0 & 0\\0 & 1\end{bmatrix} + \begin{bmatrix}0 & 0\\-0{,}5 & 0\end{bmatrix} = \begin{bmatrix}0 & 0\\-0{,}5 & 1\end{bmatrix}$

4) Bobot $W_{4,1}$ baru = $W_{4,1}$ lama + $\alpha.(|Y_1\rangle - |T_1\rangle).\langle X_4|$ = $\begin{bmatrix}0 & 1\\1 & 1\end{bmatrix}$ + $0{,}1 \cdot \left(\begin{bmatrix}1\\0\end{bmatrix} - \begin{bmatrix}1\\5\end{bmatrix}\right) \cdot \langle 1|$ = $\begin{bmatrix}0 & 1\\1 & 1\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}0\\-5\end{bmatrix} \cdot [0 \quad 1] = \begin{bmatrix}0 & 1\\1 & 1\end{bmatrix}$ $+ 0{,}1 \cdot \begin{bmatrix}0 & 0\\0 & -5\end{bmatrix} = \begin{bmatrix}0 & 1\\1 & 1\end{bmatrix} + \begin{bmatrix}0 & 0\\0 & -0{,}5\end{bmatrix} = \begin{bmatrix}0 & 1\\1 & 0{,}5\end{bmatrix}$.

5) Bobot $W_{5,1}$ baru = $W_{5,1}$ lama + $\alpha.(|Y_1\rangle - |T_1\rangle).\langle X_5|$ = $\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$ + $0{,}1 \cdot \left(\begin{bmatrix}1\\0\end{bmatrix} - \begin{bmatrix}1\\5\end{bmatrix}\right) \cdot \langle 1|$ = $\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}0\\-5\end{bmatrix} \cdot [0 \quad 1] = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$ $+ 0{,}1 \cdot \begin{bmatrix}0 & 0\\0 & -5\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0\\0 & -0{,}5\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & -0{,}5\end{bmatrix}$

6) Bobot $V_{1,1}$ baru = $V_{1,1}$ lama + $\alpha.(|Y_1\rangle - |T_1\rangle).\langle Z_1|$ = $\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$ + $0{,}1 \cdot \left(\begin{bmatrix}1\\0\end{bmatrix} - \begin{bmatrix}1\\5\end{bmatrix}\right) \cdot [1 \quad 3] = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}0\\-5\end{bmatrix} \cdot [1 \quad 3] =$ $\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}0 & 0\\-5 & -15\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0\\-0{,}5 & -1{,}5\end{bmatrix} =$ $\begin{bmatrix}1 & 0\\-0{,}5 & -1{,}5\end{bmatrix}$

7) Bobot $V_{2,1}$ baru = $V_{2,1}$ lama + $\alpha.(|Y_1\rangle - |T_1\rangle).\langle Z_2|$ = $\begin{bmatrix}0 & 0\\1 & 0\end{bmatrix}$ + $0{,}1 \cdot \left(\begin{bmatrix}1\\0\end{bmatrix} - \begin{bmatrix}1\\5\end{bmatrix}\right) \cdot [2 \quad 1] = \begin{bmatrix}0 & 0\\1 & 0\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}0\\-5\end{bmatrix} \cdot [2 \quad 1] =$ $\begin{bmatrix}0 & 0\\1 & 0\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}0 & 0\\-10 & -5\end{bmatrix} = \begin{bmatrix}0 & 0\\1 & 0\end{bmatrix} + \begin{bmatrix}0 & 0\\-1 & -0{,}5\end{bmatrix} =$ $\begin{bmatrix}0 & 0\\0 & -0{,}5\end{bmatrix}$

Kemudian dilakukan perubahan bobot untuk $W_{1,2}$, $W_{2,2}$, $W_{3,2}$, $W_{4,2}$, $W_{5,2}$, $V_{1,2}$, $V_{2,2}$ pada $Y_2 \neq T_2$ sebagai berikut.

1) Bobot $W_{1,2}$ baru = $W_{1,2}$ lama + $\alpha.(|Y_2> - |T_2>).<X_1| = \begin{bmatrix}0 & 0\\1 & 1\end{bmatrix} +$

$0{,}1 \cdot \left(\begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right) . <0| = \begin{bmatrix}0 & 0\\1 & 1\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}-2\\-3\end{bmatrix} . [1 \quad 0] = \begin{bmatrix}0 & 0\\1 & 1\end{bmatrix}$

$+ 0{,}1 . \begin{bmatrix}-2 & 0\\-3 & 0\end{bmatrix} = \begin{bmatrix}0 & 0\\1 & 1\end{bmatrix} + \begin{bmatrix}-0{,}2 & 0\\-0{,}3 & 0\end{bmatrix} = \begin{bmatrix}-0{,}2 & 0\\0{,}7 & 1\end{bmatrix}$

2) Bobot $W_{2,2}$ baru = $W_{2,2}$ lama + $\alpha.(|Y_2> - |T_2>).<X_2| = \begin{bmatrix}0 & 0\\1 & 0\end{bmatrix} +$

$0{,}1 \cdot \left(\begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right) . <1| = \begin{bmatrix}0 & 0\\1 & 0\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}-2\\-3\end{bmatrix} . [0 \quad 1] =$

$\begin{bmatrix}0 & 0\\1 & 0\end{bmatrix} + 0{,}1 . \begin{bmatrix}0 & -2\\0 & -3\end{bmatrix} = \begin{bmatrix}0 & 0\\1 & 0\end{bmatrix} + \begin{bmatrix}0 & -0{,}2\\0 & -0.3\end{bmatrix} = \begin{bmatrix}0 & -0{,}2\\1 & -0{,}3\end{bmatrix}$

3) Bobot $W_{3,2}$ baru = $W_{3,2}$ lama + $\alpha.(|Y_2> - |T_2>).<X_3| = \begin{bmatrix}1 & 0\\1 & 1\end{bmatrix} +$

$0{,}1 \cdot \left(\begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right) . <0| = \begin{bmatrix}1 & 0\\1 & 1\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}-2\\-3\end{bmatrix} . [1 \quad 0] = \begin{bmatrix}1 & 0\\1 & 1\end{bmatrix}$

$+ 0{,}1 . \begin{bmatrix}-2 & 0\\-3 & 0\end{bmatrix} = \begin{bmatrix}1 & 0\\1 & 1\end{bmatrix} + \begin{bmatrix}-0{,}2 & 0\\-0{,}3 & 0\end{bmatrix} = \begin{bmatrix}0{,}8 & 0\\0{,}7 & 1\end{bmatrix}$

4) Bobot $W_{4,2}$ baru = $W_{4,2}$ lama + $\alpha.(|Y_2> - |T_2>).<X_4| = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} +$

$0{,}1 \cdot \left(\begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right) . <1| = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}-2\\-3\end{bmatrix} . [0 \quad 1] = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$

$+ 0{,}1 . \begin{bmatrix}0 & -2\\0 & -3\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} + \begin{bmatrix}0 & -0{,}2\\0 & -0{,}3\end{bmatrix} = \begin{bmatrix}1 & -0{,}2\\0 & 0{,}7\end{bmatrix}$

5) Bobot $W_{5,2}$ baru = $W_{5,2}$ lama + $\alpha.(|Y_2> - |T_2>).<X_5| = \begin{bmatrix}0 & 1\\0 & 1\end{bmatrix} +$

$0{,}1 \cdot \left(\begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right) . <1| = \begin{bmatrix}0 & 1\\0 & 1\end{bmatrix} + 0{,}1 \cdot \begin{bmatrix}-2\\-3\end{bmatrix} . [0 \quad 1] = \begin{bmatrix}0 & 1\\0 & 1\end{bmatrix}$

$+ 0{,}1 . \begin{bmatrix}0 & -2\\0 & -3\end{bmatrix} = \begin{bmatrix}0 & 1\\0 & 1\end{bmatrix} + \begin{bmatrix}0 & -0{,}2\\0 & -0{,}3\end{bmatrix} = \begin{bmatrix}0 & 0{,}8\\0 & 0{,}7\end{bmatrix}$

6) Bobot $V_{1,2}$ baru = $V_{1,2}$ lama + α.(|Y_2> - |T_2>).<Z_1| = $\begin{bmatrix}0 & 0\\1 & 1\end{bmatrix}$ + 0,1 . $\left(\begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right)$. $[1 \quad 3]$ = $\begin{bmatrix}0 & 0\\1 & 1\end{bmatrix}$ + 0,1 . $\begin{bmatrix}-2\\-3\end{bmatrix}$. $[1 \quad 3]$ = $\begin{bmatrix}0 & 0\\1 & 1\end{bmatrix}$ + 0,1. $\begin{bmatrix}-2 & -6\\-3 & -9\end{bmatrix}$ = $\begin{bmatrix}0 & 0\\1 & 1\end{bmatrix}$ + $\begin{bmatrix}-0,2 & -0,6\\-0,3 & -0,9\end{bmatrix}$ = $\begin{bmatrix}-0,2 & -0,6\\0,7 & 0,1\end{bmatrix}$

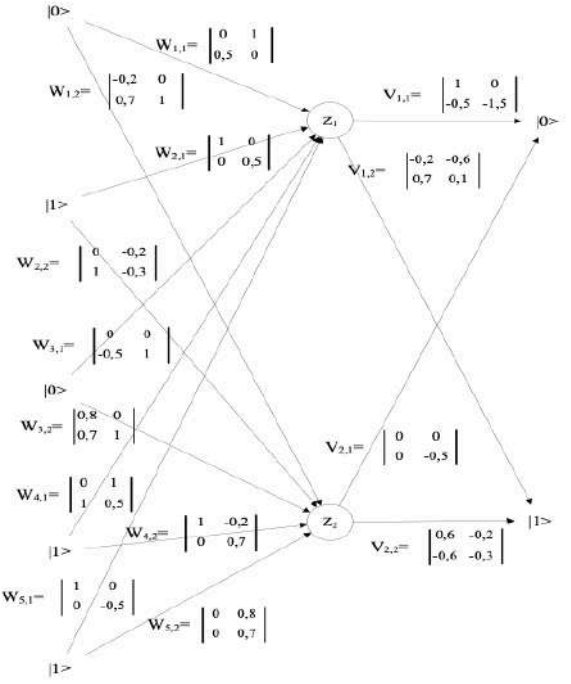
7) Bobot $V_{2,2}$ baru = $V_{2,2}$ lama + α.(|Y_2> - |T_2>).<Z_2| = $\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$ + 0,1 . $\left(\begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}2\\4\end{bmatrix}\right)$. $[2 \quad 1]$ = $\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$ + 0,1 . $\begin{bmatrix}-2\\-3\end{bmatrix}$. $[2 \quad 1]$ = $\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$ + 0,1. $\begin{bmatrix}-4 & -2\\-6 & -3\end{bmatrix}$ = $\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$ + $\begin{bmatrix}-0,4 & -0,2\\-0,6 & -0,3\end{bmatrix}$ = $\begin{bmatrix}0,6 & -0,2\\-0,6 & -0,3\end{bmatrix}$

Kemudian untuk perhitungan nilai *error* sesuai dengan tahapan berikut ini :

1) Ambil nilai output sementara Y_1 = $\begin{bmatrix}1\\5\end{bmatrix}$ dengan persamaan 2.1 menjadi 1|0> + 5|1> lalu pilih pasangan *qubit* |0> atau |1> dengan mencocokkan nilai *qubit* target dari Y_1. Karena nilai *qubit* target Y_1 = |0> maka yang dipilih adalah nilai 1 dari pasangan *qubit* |0> dari 1|0> + 5|1>.
2) Ambil nilai output sementara Y_2 = $\begin{bmatrix}2\\4\end{bmatrix}$ dengan persamaan 2.1 menjadi 2|0> + 4|1> lalu pilih pasangan *qubit* |0> atau |1> dengan mencocokkan nilai *qubit* target dari Y_2. Karena nilai *qubit* target Y_2 = |1> maka yang dipilih adalah nilai 4 dari pasangan *qubit* |1> dari 2|0> + 4|1>.
3) Kemudian nilai 1 dan 4 dari langkah sebelumnya digunakan untuk perhitungan nilai error dengan persamaan 2.4 menjadi Error ke-1 = $\frac{1}{2}$ $((1-0)^2 + (4-1)^2)$ = $\frac{1}{2}$ $(1+9) = 5$

Setelah perubahan bobot W dan V selesai, Kemudian proses pembelajaran dilanjutkan untuk data ke-2 dengan dataset nomor 2 yaitu 01111 sebagai input

dan 01 sebagai output yang diharapkan. Untuk lebih lengkapnya dapat dilihat pada gambar 8.5.



Gambar 8.5: Data ke-2 dengan bobot baru

Berikut ini output pada hidden Z_1 dan Z_2 pada iterasi ke-2.

1) Output $Z_1 = W_{1,1}.|X_1> + W_{2,1}.|X_2> + W_{3,1}.|X_3> + W_{4,1}.|X_4> + W_{5,1}.|X_5> = \begin{bmatrix} 0 & 1 \\ 0{,}5 & 0 \end{bmatrix}.|0> + \begin{bmatrix} 1 & 0 \\ 0 & 0{,}5 \end{bmatrix}.|1> + \begin{bmatrix} 0 & 0 \\ -0{,}5 & 1 \end{bmatrix}.|1> + \begin{bmatrix} 0 & 1 \\ 1 & 0{,}5 \end{bmatrix}.|1> + \begin{bmatrix} 1 & 0 \\ 0 & -0{,}5 \end{bmatrix}.|1> = \begin{bmatrix} 0 & 1 \\ 0{,}5 & 0 \end{bmatrix}.\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0{,}5 \end{bmatrix}.\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -0{,}5 & 1 \end{bmatrix}.\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0{,}5 \end{bmatrix}.\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & -0{,}5 \end{bmatrix}.\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0{,}5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0{,}5 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0{,}5 \end{bmatrix} + \begin{bmatrix} 0 \\ -0{,}5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

2) Output $Z_2 = W_{1,2}.|X_1> + W_{2,2}.|X_2> + W_{3,2}.|X_3> + W_{4,2}.|X_4> + W_{5,2}.|X_5> = \begin{bmatrix} -0,2 & 0 \\ 0,7 & 1 \end{bmatrix}.|0> + \begin{bmatrix} 0 & -0,2 \\ 1 & -0,3 \end{bmatrix}.|1> + \begin{bmatrix} 0,8 & 0 \\ 0,7 & 1 \end{bmatrix}.|1> + \begin{bmatrix} 1 & -0,2 \\ 0 & 0,7 \end{bmatrix}.|1> + \begin{bmatrix} 0 & 0,8 \\ 0 & 0,7 \end{bmatrix}.|1> = \begin{bmatrix} -0,2 & 0 \\ 0,7 & 1 \end{bmatrix}.\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & -0,2 \\ 1 & -0,3 \end{bmatrix}.\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0,8 & 0 \\ 0,7 & 1 \end{bmatrix}.\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 & -0,2 \\ 0 & 0,7 \end{bmatrix}.\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 & 0,8 \\ 0 & 0,7 \end{bmatrix}.\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0,2 \\ 0,7 \end{bmatrix} + \begin{bmatrix} -0,2 \\ -0,3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -0,2 \\ 0,7 \end{bmatrix} + \begin{bmatrix} 0,8 \\ 0,7 \end{bmatrix} = \begin{bmatrix} 0,2 \\ 2,8 \end{bmatrix}$

3) Output $Y_1 = V_{1,1}.|Z_1> + V_{2,1}.|Z_2> = \begin{bmatrix} 1 & 0 \\ -0,5 & -1,5 \end{bmatrix}.\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -0,5 \end{bmatrix}.\begin{bmatrix} 0,2 \\ 2,8 \end{bmatrix} = \begin{bmatrix} 1 \\ -3,5 \end{bmatrix} + \begin{bmatrix} 0 \\ -1,4 \end{bmatrix} = \begin{bmatrix} 1 \\ -4,9 \end{bmatrix}$

4) Output $Y_2 = V_{1,2}.|Z_1> + V_{2,2}.|Z_2> = \begin{bmatrix} -0,2 & -0,6 \\ 0,7 & 0,1 \end{bmatrix}.\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0,6 & -0,2 \\ -0,6 & -0,3 \end{bmatrix}.\begin{bmatrix} 0,2 \\ 2,8 \end{bmatrix} = \begin{bmatrix} -1,4 \\ 0,9 \end{bmatrix} + \begin{bmatrix} -0,44 \\ -0,96 \end{bmatrix} = \begin{bmatrix} -1,84 \\ -0,06 \end{bmatrix}$

Output sementara Y_1 dan Y_2 pada data ke-2 ini dibandingkan dengan output yang diharapkan dari $Y_1 = |0> = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ dan $Y_2 = |1> = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ di mana $\begin{bmatrix} 1 \\ 0 \end{bmatrix} \neq \begin{bmatrix} 1 \\ -4,9 \end{bmatrix}$ dan $\begin{bmatrix} 0 \\ 1 \end{bmatrix} \neq \begin{bmatrix} -1,84 \\ -0,06 \end{bmatrix}$ sehingga dilakukan perubahan dari setiap bobot dengan langkah yang sama dengan langkah sebelumnya. Perubahan bobot ini dilakukan terus menerus sampai nilai output yang diharapkan (Y) sama dengan nilai output sementara(T). Dengan kata lain nilai bobot akan terus menerus berubah sampai kondisi Y = T terpenuhi atau nilai error sama dengan 0.

Namun dalam proses pembelajaran untuk mencapai kondisi Y = T akan memerlukan iterasi yang cukup lama karena mengunakan konsep trial and error. Dataset yang digunakan memiliki 24 sampel sehingga paling sedikit melakukan iterasi sebanyak 24 kali akan tetapi tidak ada jaminan pada iterasi ke-24 akan menghasilkan kondisi Y = T atau error = 0. Seandainya pada iterasi ke-24 tidak dihasilkan kondisi yang diinginkan maka iterasi kembali ke iterasi pertama akan tetapi pada keadaan ini sudah dihitung satu siklus atau disebut dengan satu epoch.

Satu siklus (epoch) dalam proses pembelajaran memiliki 24 iterasi di mana semakin banyak jumlah epoch maka semakin lama waktu yang dibutuhkan dalam proses pembelajaran. Sedangkan kecepatan proses pembelajaran bergantung pada nilai learning rate α dan hasil akhir dari proses pembelajaran ini adalah nilai akhir dari bobot W dan V yang sudah memenuhi kondisi Y = T atau error = 0. Keseluruhan proses ini, mulai dari penghitungan jumlah iterasi, jumlah epoch, nilai learning rate, nilai bobot akhir W dan V akan bergantung pada kondisi Y = T atau disebut dengan konvergensi.

Bab 9

Fuzzy Neural Network, (Application System Control)

9.1 Pendahuluan

Logika fuzzy pertama klai dikembangkan oleh Zadeh di pertengahan tahun 1960 untuk mewakili nilai ynag tidak pasti, namun efektif untuk menggambarkan prilaku sistem yang kompleks tidak jelas atau tidak mudah dianalisis secara matematis. Variabel fuzzy diproses dengan menggunakan sistem yang disebut dengan logika fuzzy hal ini melibatkan fuzzyfikasi, inferensi fuzzy dan dufizzyfikasi. (Lakhmi dan Martin, 1998). Logika fuzzy adalah salah satu cara yang tepat untuk memetakan suatu ruang input ke dalam suatu ruang output (Kusumadewi, 2004:1). Logika fuzzy berbeda dengan logika digital biasa, di mana logika digital biasa hanya mengenal dua keadaan yaitu: Ya dan Tidak atau ON dan OFF atau High dan Low atau "1" dan "0". Sedangkan Logika Fuzzy meniru cara berpikir manusia dengan menggunakan konsep sifat kesamaran suatu nilai. Dengan himpunan fuzzy, suatu objek dapat menjadi anggota dari banyak himpunan dengan derajat keanggotaan yang berbeda dalam masing-masing himpunan (Wulandari, 2010). Penggunaan logika fuzzy juga sangat tepat digunakan untuk mendapatkan nilai secara pasti dari input yang diterima berupa bahasa dan mengubah menjadi angka dengan mengubah menjadi nilai keanggotaan dalam himpunan Fuzzy.(Jyh et al, 1997). Jaringan syaraf tiruan Artificial Neural Network (ANN) merupakan salah satu jaringan yang dirancang untuk menyerupai sistem kerja otak manusia yang bertujuan untuk melaksanakan tugas tertentu. (Haykin, 2009).

JST merupakan sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Secara sederhana, JST adalah sebuat alat

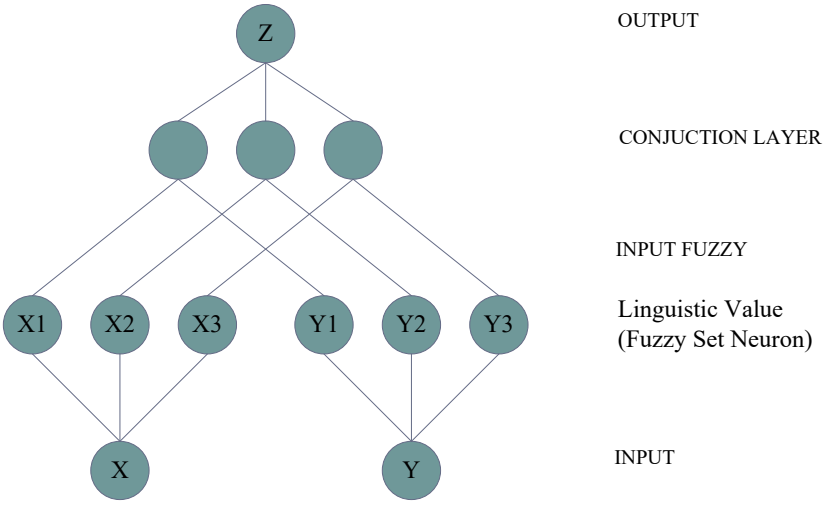
pemodelan data statistik non-linear. JST dapat digunakan untuk memodelkan hubungan yang kompleks antara input dan output untuk menemukan pola-pola pada data (Sarosa, M., Ahmad, A.S., dan Riyanto, B. Noer., 2007).

9.2 Fuzzy Neural Network

Fuzzy *neural network (FNN)* adalah salah satu jaringan saraf tiruan dengan basis aturan fuzzy (*fuzzy rule base*), *NN* adalah suatu ilmu *Soft Computing* yang diambil dari kemampuan otak manusia yang mampu memberikan stimulasi, melakukan proses dan memberikan *output*.(Jain, L.C., dan Martin, N.M., 1998). Dalam implementasi aturan fuzzy pada jaringan syaraf tiruan dibutuhkan berupa level aturan-aturan fuzzy *rules* yang dikelompokan berdasarkan lapisan-lapisan. Neuron pada umumnya terletak pada satu bagian lapisan yang sama akan memiliki kondisi yang sama. Pada setiap bagian lapisan dengan kondisi yang sama, neuron akan mempunyai suatu fungsi aktivasi yang sama. Apabila neuron dalam satu lapisan yang tersembunyi, akan dihubungkan dengan neuron pada lapisan output maka setiap neuron pada lapisan tersembunyi juga harus dihubungkan dengan setiap bagian lapisan outputnya (Kusumadewi, S. 2003: 212-214). Neural netwok dibangun dari sekian banyak node atau unit yang dihubungkan secara langsung pada link. Link dari unit yang satu ke lainya digunakan untuk melakukan propagasi aktifasi dari unit pertama ke unit lainnya. Setiap link yang digunakan memiliki bobot numerik untuk menentukan kekuatan dari sebuah sebuah konektivitas (Li, R.P., Mukaidono, M., dan Turksen, I.B., 2002).

Fuzzy Neural Network (FNN) adalah suatu model *soft computing* yang telah dilatih dengan menggunakan jaringan syaraf, namun pada struktur jaringannya di interprestasikan dengan sekelompok aturan-aturan fuzzy. pada FNN, parameter-parameter yang dimiliki oleh neuron dengan bobot penghubung yang biasanya akan di sajikan secara numeris, dapat digantikan dengan parameter fuzzy. Pada FNN, unsur-unsur terkait pada jaringan syaraf, tidak lagi menggunakan bentuk klasik seperti pada jaringan syaraf pada umumnya, namun telah diganti dengan pendekatan logika fuzzy. penggunaan logika fuzzy ini tentu saja digunakan untuk mengantisipasi adanya ketidakpastian yang terjadi pada unsur-unsur jaringan syaraf tersebut (Jain, L.C., dan Martin, N.M., 1998). Salah satu bentuk dari model fuzzy neural network dengan menggunakan lapisan

input, input fuzzy, conjuction layer dan output yang berupa single tons yang merupakan bobot konektivitas pada lapisan terakhir (Kasabov, N.K., 1996).



Gambar 9.1: Struktur Fuzzy Neural Network empat lapisan (Kasabov, N.K.,1996)

formula yang digunakan untuk menentukan besarnya keluaran atau *output* z sebagai berikut :

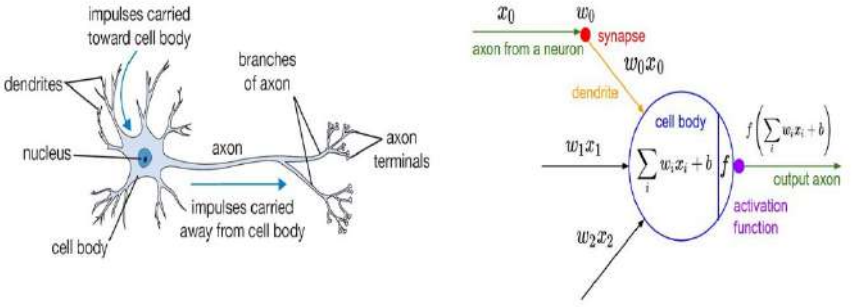
$$z = \sum_{i=l}^{m} \mu i \omega i \; / \; \sum_{i=l}^{m} \mu i$$

di mana Z adalah suatu level aktivasi dari unit *output*, μ_i dan ω_i berturut-turut adalah fungsi keanggotaan dan jalur terbobot unit *i* pada lapisan antara *output* fuzzy ke *output*.

a. Konsep Neural Network

Konsep dasar *neural network* dimulai dari otak manusia, di mana otak memuat sekitar 10^{11} neuron. Neuron ini berfungi memproses setiap informasi yang masuk. Satu lapisan neuron memiliki hanya 1 akson, dan minimal 1 dendrit. Pada setiap sel saraf yang terhubung dengan saraf lain, jumlahnya mencapai 10^4 sinapsis. Masing-maisng sel saraf yang saling berinteraksi satu sama lain akan menghasilkan kemampuan tertentu pada sistem kerja otak manusia (Kusemadewi, S. Dan Sri, H, 2010).

Neural network (NN) adalah suatu model yang terinspirasi bagaimana neuron dalam oatak manusia bekerja. Setiap neuron pada otak manusia saling berhubungan dan informasi mengalir dari setiap neuron tersebut. Gambar di bawah ini adalah suatu ilustrasi neuron dengan model matematisnya (Fausset, 1994:6).



Gambar 9.2: Struktur Neuron sel pada (Fausset, 1994:6)

Jaringan otak manusia tidak kurang dari 10^{13} neuron yang masing-masing terhubung oleh sekitar 10^{15} dendrit. Neuron memiliki berbagai komponen utama sebagai berikut. (Fausett, 1994:4).

1. Dendrit (*Dendrites*) berfungsi sebagai saluran penyampai sinyal atau informasi dari satu neuron ke neuron lainnya.
2. Akson (*Axon*) difungsikan untuk mengirimkan impuls-impuls ke sel sel saraf lainnya
3. Badan sel (Soma), berfungsi sebagai tempat pengolahan informasi

b. Struktur Neural Network

Proses tahapan struktur neuron pada otak manusia, yang dijelaskan dia atas, adalah konsep dasar dari pembangunan *neural network* buatan (*Artificial Neural Network*) yang terbentuk. Dasar dari *Artificial Neural Network* (ANN) merupakan sebuah model matematik yang berupa kumpulan unit yang terhubung secara parallel yang bentuknya menyerupai jaringan saraf pada ptak manusia. (Warren McCulloch dan Walter Pitts, 1943). Pada FNN, unsur utama pada jaringan saraf, tidak lagi menggunakan bentuk klasik seperti pada jaringan saraf pada umumnya, namun telah diganti dengan pendekatan logika fuzzy.

penggunaan logika fuzzy ini tentu saja digunakan untuk mengantisipasi adanya ketidakpastian yang terjadi pada unsur-unsur saraf tersebut

c. Neuron dengan persamaan fuzzy

Neuron fuzzy dengan *input* dan satu *output* dibentuk dengan relasi *input-output* dalam bentuk IF-THEN (Lin, 1996).

$$\text{IF } X_1 \text{ AND } ... \text{ AND } X_n \text{ THEN } Y$$

Dengan X1, ... Xn adalah input dan Y adalah *output.* Neuron fuzzy dapat dideskripsikan sebagai relasi R, sebagai contoh :

$$R = X_1 \times X_2 ... \times X_n \times Y$$

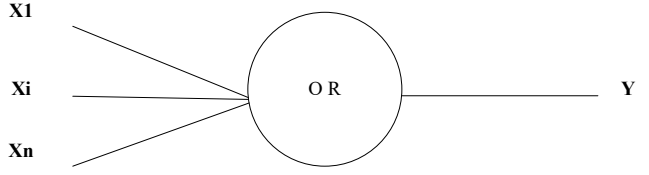
Atau

$$R = f(X_1, X_2 ... X_n, Y)$$

Dengan f(.) adalah fungsi implikasi. Sehingga dengan menggunakan aturan komposisi inferensi, *output* neuron fuzzy dapat diberikan sebagai :

$$Y_i = X_1 \, o \, (X_2 \, o (... o (X_n \, o \, R_i) ...)$$

dengan o mempresentasikan aturan komposisi inferensi, seperti max t-norm, (Lin, 1996)



Gambar 9.3: Arsitektur neuron fuzzy menggunakan persamaan fuzzy (Hanafi, M, 2011)

Contoh :

Misalkan terdapat aturan :

$$Rr : \text{IF } X \text{ is } A_i \text{ and } Y \text{ is } B_j \text{ THEN } Z \text{ is } C_p$$

Dengan A_i, B_j dan C_p adalah himpunan bagian fuzzy yang di representaiskan sebagai himpunan fuzzy segitiga dengan pusat masing-masing di a_i, b_j dan c_p.

Sebagai contoh, misalkan terdapat 2 buah aturan sebagai berikut :

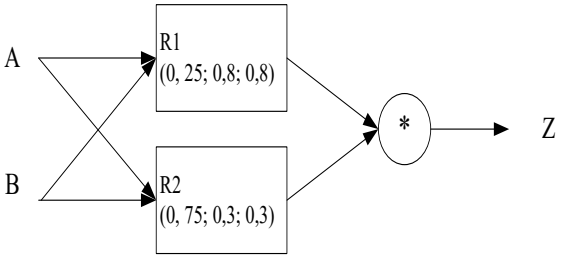
R_1 : IF x is RENDAH an y is NAIK THEN z is BANYAK

R_3 : IF x is TINGGI and y is TURUN THEN z is SEDIKIT

Pada variabel X dengan nilai pusat untuk RENDAH = 0,25 ; dan TINGGI= 0,75

Pada variabel Y dengan nilai pusat untuk TURUN = 0,3; dan NAIK= 0,7

Pada variabel Z dengan nilai pusat untuk SEDIKIT = 0,2; dan BANYAK= 0,8

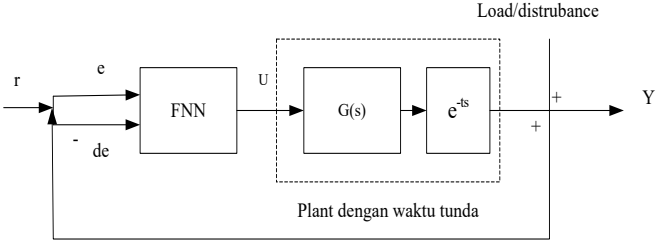


Gambar 9.4: Contoh Neuron Fuzzy (Hanafi, M, 2011)

Apabila diberikan data baru X = A' dan B = Y' maka kita dapat mengevaluasi aturannya. Node yang berisikan (*) mengkombinasikan *input* dengan opertaor * untuk mendapatkan *output* Z.

9.3 Pembuatan Model Sistem Kontrol

Struktur pada sistem kontrol dengan waktu tunda ditunjukan pada gambar 9.5. signal kontrol *u* merupakan signal dihasilkan oleh pengontrol FNN yang digunakan untuk mengontrol G (s) dengan waktu tunda (e^{-ts}).

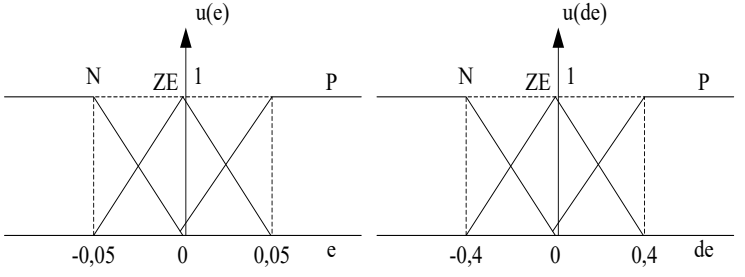


Gambar 9.5: Struktur sistem kontrol FNN pada plant dengan waktu tunda (Hanafi, M, 2011)

Simbol dari G(*s*) adalah transfer *function* dari *plant* dan e^{-ts} merupakan unsur waktu tunda yang terjadi pada plant. Sedangkan masukan *r* merupakan setpoint atau *input* sistem, signal informasi yang diterima oleh pengontrol FNN merupakan signal *error* (*e*) dan perubahan *error* (*de*) yang terjadi antara *output* (*y*) sistem dengan *input* sistem (Hanafi, M., 2011).

9.4 Menentukan Fungsi Keanggotaan Fuzzy

Untuk mendapatkan besaran linguistik dari signal masukan yang berupa signal *script* dilakukan dengan proses fuzzifikasi. Proses ini digunakan untuk memetakan suatu nilai *input* yang dilewatkan (*e* dan *de*) ke dalam derajat keanggotaan fuzzy. fungsi keanggotaan yang digunakan pada kasus ini adalah fungsi segtiga yang memiliki tiga variabel linguistik yaitu negatif (N), *about zero* (*Z*) dan positif (*P*). Berdasarkan dari hasil analisa respon yang ditunjukan pada sistem kontrol dengan waktu tunda yang kemudian dilakukan penyesuaian dengan coba-coba (*trial and error*) sampai diperoleh pengontrol yang baik, maka didapatkan fungsi keanggotaan segitiga untuk masukan *e* dan *de* seperti terlihat pada gambar di bawah ini



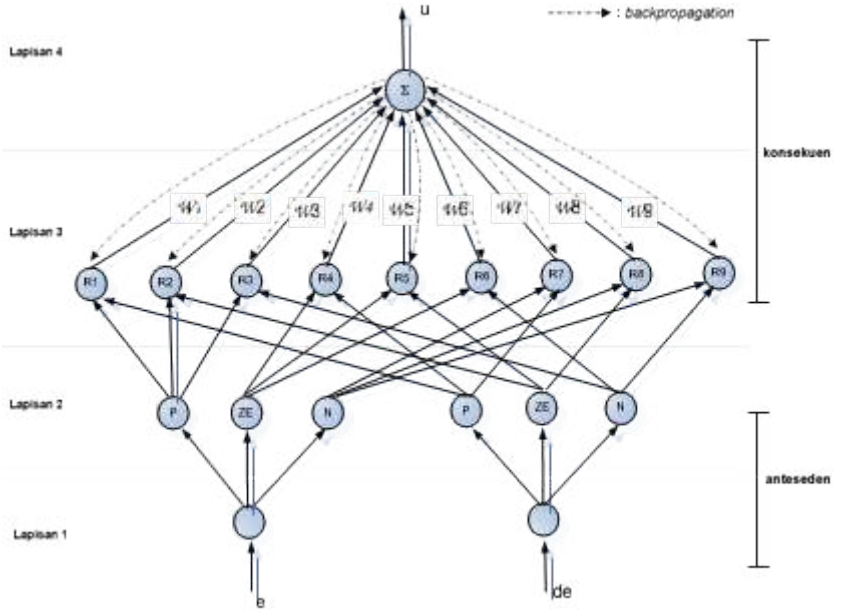
Gambar 9.6: Fungsi keanggotaan dan variabel linguistik untuk *e* dan *de* (Hanafi, M, 2011)

Selanjutnya dari tiga variabel linguistik fuzzy yang masing-masing untuk *e* dan *de*, disusun aturan-aturan fuzzy (*fuzzy rule base*). Aturan- aturan fuzzy disusun untuk mengakomodasi semua kemungkinan masukan nilai linguistik *e* dan *de*,

dan dengan *conjugtion* AND semua pasangan variabel linguistik antara masukan *e* dan *de* dikombinasikan sehingga diperoleh 9 aturan fuzzy.

9.5 Perancangan Struktur Jaringan FNN

Struktur jaringan saraf berbasis fuzzy atau fuzzy *neural network* sebagai *controller* yang dirancang pada penelitian ini ditunjukan pada gambar di bawah ini.



Gambar 9.7: Struktur Fuzzy Neural Network (Hanafi, M, 2011)

Pada lapisan pertama struktur FNN merupakan suatu lapisan input yang berfungsi melewatkan variabel masukan. *Output* dari lapisan pertama ini adalah $y_{l,i}^{(1)}$ = e dan $y_{2,j}^{(1)}$ = de. Proses pada lapisan yang kedua dilakukan suatu proses fuzzyfikasi dengan menempatkan nilai input yang dilewatkan (e dan de) ke dalam derajat keanggotaan fuzzy seperti terlihat pada gambar 1.5. keluaran dari prose lapisan yang kedua ini adalah $v_{1,i}^{(2)} = \mu(v_{1,i}^{(1)})$ dan $v_{2,j}^{(2)} = \mu\,(v_{1,j}^{(1)})$. Pada proses lapisan yang ketiga menunjukan suatu lapisan *conjuction*, digunakan

sebanyak 9 neuron dalam merealisasikan 9 aturan (*rule*) fuzzy. Dengan Menggunakan salah satu penghubung yaitu AND untuk besaran fuzzy dari masukan *e* dan *de*, maka fungsi aktivasi yang digunakan pada lapisannya adalah $\mu_{i,j}$ = min ($\mu(e_i)$, $\mu(de_j)$) atau $\mu_{i,j}$ = min ($\mu(y_{1,i}^{(2)})$,$\mu(y_{2,j}^{(2)})$). Keluarannya dapat di rumuskan dengan $y_{i,j}^{(3)} = \mu_{i,j}$. Lapisan keempat merupakan lapisan *output*. Pada lapisan ini dilakukan pembobotan dengan w_i pada setiap masukan. Keluaran dari lapisan ini dihitung dengan persamaan :

$$y^{(4)} = \frac{\sum_{i,j=1}^{n} y_{i,j}^{(3)} w_i}{\sum_{i,j=1}^{n} y_{i,j}^{(3)}} = \frac{\sum_{i,j=1}^{n} \mu_{i,j}.w_i}{\sum_{i,j=1}^{n} \mu_{i,j}} \text{ dengan n = 9 rules, maka: } u = \frac{\sum_{i,j=1}^{9} \mu_{i,j}.w_i}{\sum_{i,j=1}^{9} \mu_{i,j}}$$

9.6 Pelatihan FNN

Proses pelatihan yang dilakukan pada FNN ini menggunakan algoritma *backpropagation*. Proses dari pelatiahan ini adalah hanya untuk melatih bobot koneksi pada lapisan ke 4, pada lapisan yang lain diberikan nilai bobot koneksi diset dengan nilai satu. Jika *error* antara keluaran FNN yang sebenarnya (u(t)) dengan keluaran yang diinginkan (ud(t)), ditentukan dengan rumus $E = \frac{1}{2}(ud(t)) - u(t))^2$ dan untuk modifkasi bobot koneksi jaringan digunakan persamaan 2.3, di mana $\delta_k = (t_k - y_k)f(y\_in_k)$, sehingga modifikasi nilai bobot koneksi pada lapisan ke 4 penggunaan jaringan FNN yang dirancang ini dilakukan dengan persamaan :

$$w_{i4}(t+1) = w_{i4}(t) + a(-\frac{\delta E}{\delta w_{i4}}) + \beta(w_{i4}(t) - w_{i4}(t-1)), \text{dengan}$$

$$\frac{\partial E}{\partial w_{i4}} = \frac{\partial E}{\delta u}\frac{\partial u}{\partial w_{i4}} = (ud(t) - u(t))\frac{\mu_i}{\sum_{i=1}^{9} \mu_i}, \text{ dengan } \alpha \text{ learning factor dan } \beta =$$

momentum *factor*.

9.7 Pengujian Sistem

Pengujian sistem dilakukan dengan simulasi menggunakan *software* matlab. Model proses atau plant yang digunakan dalam pengujian ini adalah *Heat Exchanger*. Heat Exchanger merupakan contoh proses atau plant yang mengandung *deadline*. Fungsi alih *Heat Exchanger* dapat dinyatakan dalam bentuk sistem orde satu (Santoso, 2003).

$$\frac{T(s)}{F_n(s)} = \frac{K_{pex}}{\tau_{ex}s+1} e^{-ts} = G_{ex}(s)$$

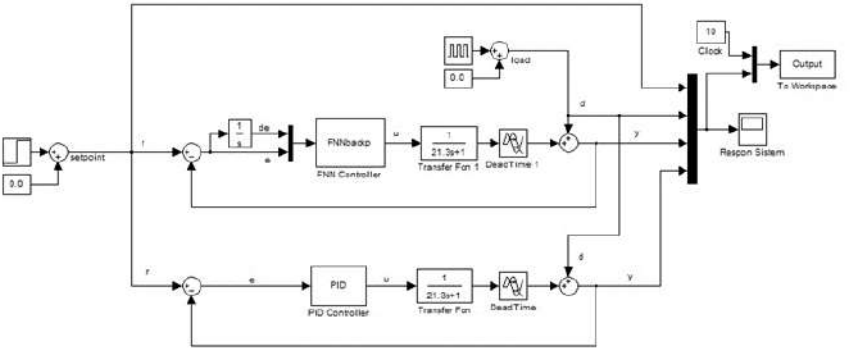
Karena ada waktu tunda / death time pada plant makan fungsi alih plant menjadi,

$$\frac{T(s)}{F_h(s)} = \frac{K_{pex}}{\tau_{ex}s+1} e^{-ts} = G_{ex}(s)$$

Dengan t merupakan death time process. Pada pengujian ini fungsi alih plant dinyatakan dengan,

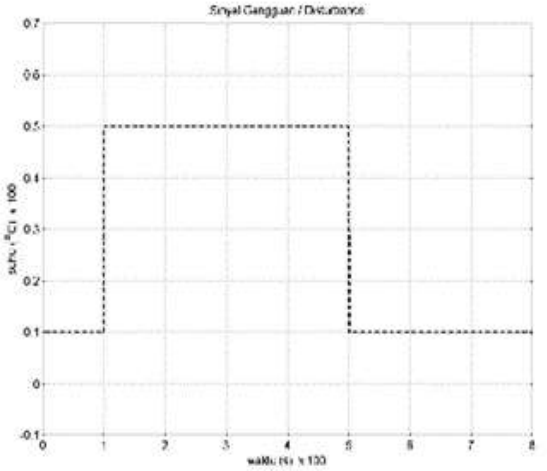
$$G(s) = \frac{1}{21,3s+1} e^{-10}$$

Berdasarkan struktur kontrol dengan fungsi alih plant pada persamaan (3,11), dibuat rangkaian simulasi untuk pengujian sistem kontrol menggunakan simulink matlab seperti pada gambar di bawah ini



Gambar 9.8: Rangkaian simulasi sistem kontrol PID dan FNN (Hanafi, M, 2011)

Pengujian dilakukan dengan step respon *system* dan pengaruh adanya gangguan (*disturbance*) atau *load*, seperti yang terlihat pada gambar 9.8. untuk input step respon system ini diuji dengan lama waktu tunda yang bervariasi, yaitu 10s, 20s, 30s dan 45s.



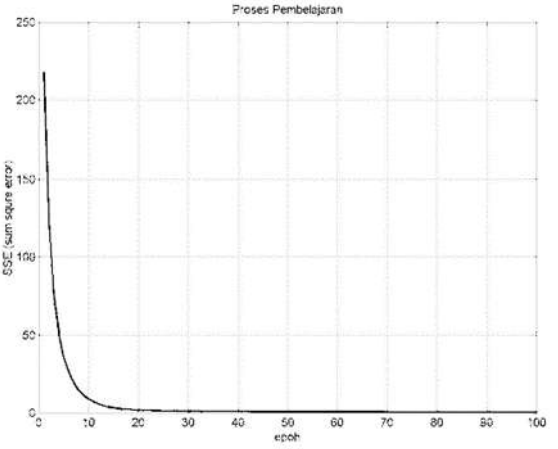
Gambar 9.9: Sinyal gangguan atau load (Hanafi, M, 2011)

Pengujian berikutnya adalah pengujian karena pengaruh adanya gangguan (*disturbance*) atau *load*. Pada pengujian ini input sistem diset sebesar 1ºC, kemudian diberi *disturbance* atau *load* dengan keanikan suhu menjadi 50 ºC dalam 400 s dan turun kembali ke posisi set poinnya, seerti yang terlihat pada gambar 1.8b. Pengujian ini juga dilakukan dengan lama waktu tunda yang bervariasi, yaitu 10s, 20s, 30s dan 45s.

9.8 Hasil Pelatihan FNN

Pada proses pelatihan FNN ini digunakan nilai bobot koneksi awal = 0 untuk semua bobot koneksi. Nilai *learning factor* (α) yang digunakan 0,001 dan momentum *factor* (β) nya adalah 0,015. Kombinasi nilai α dan β ini berpengaruh terhadap kecepatan proses pelatihan dan ketepatan nilai bobot yang diperoleh. Nilai α dan β disini didapatkan secara *trial* and *error*, semakin kecil nilai α proses pembelajar akan semakin lama, tetapi tingkat keakuratan bobot yang

diperoleh akan semakin tinggi. Proses pelatihan dibatasi sampai 100 epoh. Hasil proses pembelajarannya seperti pada gambar 9.10 di bawah ini

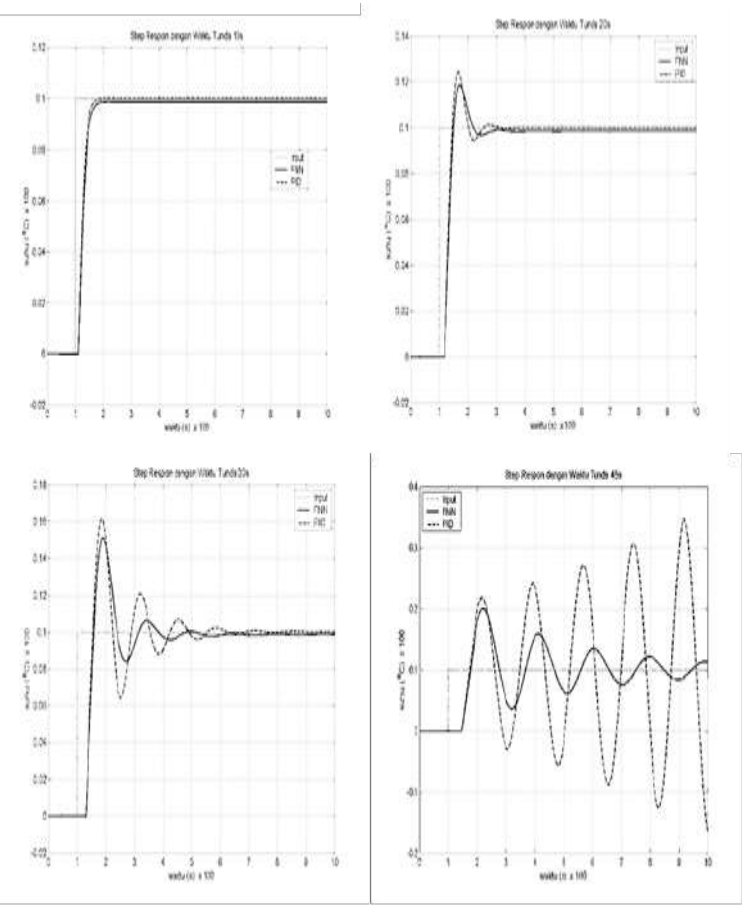


Gambar 9.10: Proses Pelatihan FNN (Hanafi, M, 2011)

Proses pembelajaran FNN pada gambar 2.1 di atas menunjukan bahwa nilai SSE (*sum square error*) semakin kecil. Hal ini berarti jaringan saraf pada FNN dapat melakukan proses pembelajaran dengan baik terhadap data yang dilatihkan.

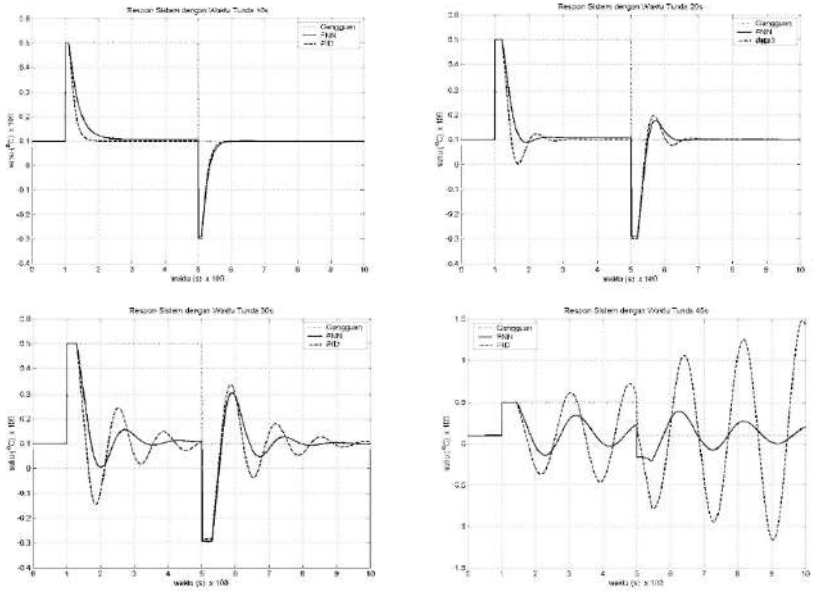
a. Respon sistem

   Hasil step respon system dari kedua sistem kontrol dengan lama waktu tunda yang bervariasi, yaitu 10s, 20s, 30s dan 45s, ditunjukan pada gambar 9.10 untuk step respon system dengan waktu tunda 10s, seperti yang ditunjukan pada gambar 14a, pada *controler* PID sudah dilakukan *tuning* parameter kontrol PID yaitu Kp, Ki dan Kd sehingga diperoleh respon sistem yang baik.



Gambar 9.11: Step respon system dengan waktu tunda: 10s, 20s, 30s dan 45s (Hanafi, M, 2011)

Hasil dari pengujian yang terlihat pada gambar 9.10. di atas, menunjukan bahwa FNN mampu memberikan step respon system yang baik untuk seluruh waktu tunda yang diujikan. Untuk pengujian sistem dengan menggunakan disturbance atau load terlihat pada gambar 9.11. berakibat adanya gangguan (disturbance), pada nilai *controller* variable dalam hal ini terlihat temperatur terlihat naik selama 400s kemudian kembali normal, sehingga *controller* meningkatkan sinyal kontrol *u*, untuk membawa sistem kembali pada nilai *set point*-nya.



Gambar 9.12: Respon sistem akibat *disturbance* atau *load* untuk waktu tunda 10s, 20s, 30s dan 45s (Hanafi, M, 2011)

Secara umum hasil simulasi pengujian, baik untuk step respon sistem akibat adanya disturbance atau load menunjukan bahwa controller FNN dapat memberikan hasil pengontrolan yang baik untuk seluruh waktu tunda yang diujikan. Di sini FNN mampu memberikan respon sistem yang baik meskipun pada unsur fuzzy yang digunakan hanya menggunakan sembilan rule dan kesembilan rule tersebut diperoleh dengan cara mengkombinasikan variabel-variabel linguistik berdasarkan masukan dari e dan de.

Kesimpulan dari hasil simulasi pengendalian sistem kontrol dengan waktu tunda menggunakan Fuzzy Neural Network (FNN) controller menunjukan bahwa, meskipun dengan menggunakan rule base yang terbatas pada unsur logika fuzzy, kemampuan belajar yang dimiliki jaringan syaraf tiruan pada FNN mampu mengoptimalkan unsur-unsur logika fuzzynya, sehingga respon sistem yang baik mampu ditunjukan oleh FNN controller untuk seluruh waktu tunda yang diujikan.

Bab 10

Algoritma Kohonen Self Organizing Map (SOM)

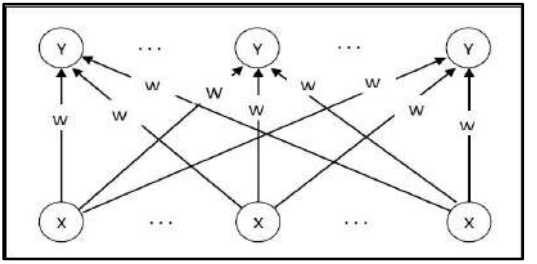
10.1 Pendahuluan

Metode Self-Organizing Map (SOM) atau sering disebut *topology-preseving map* pertama kali diperkenalkan oleh Professor Teuvo Kohonen pada tahu 1996. Pada algoritma SOM, vector bobot untuk setiap unit cluster berfungsi sebagai contoh dari input pola yang terkait dengan cluster itu. Selama proses self-organizing, cluster satuan yang bobotnya sesuai dengan pola vektor input yang paling dekat (biasanya, kuadrat dari jarak Euclidean minimum) dipilih sebagai pemenang. Unit pemenang dan unit tetangganya (dalam pengertian topoligi dari unit cluster) terus memperbaharui bobot merek. Setiap output akan bereaksi terhadap pola input tertentu sehingga hasil Kohonen SOM akan menunjukkan adanya kesamaan ciri antara anggoata dalam cluster sama (Ambarwati, 2014).

Dalam jaringan self organizing map, neuron target tidak diletakkan dalam sebuah baris seperti layaknya model JST yang lain. Neuron target diletakkan dalam dua dimensi yang bentuk/topologinya dapat diatur. Topologi yang berbeda akan menghasilkan neuron sekitar neuron pemenang yang berbeda sehingga bobot yang dihasilkan juga berbeda. Pada SOM, perubahan bobot tidak hanya dilakukan pada bobot garis yang terhubung ke neuron pemenang saja, tetapi juga pada bobot garis pada bobot garis ke neuron-neuron di sekitarnya. Neuron di sekitar neuron pemenang ditentukan berdasarkan jaraknya dari neuron pemenang (Hariri and Pamungkas, 2016).

Arsitektur Kohonen Self Organizing Map merupakan jaringan yang terdiri dari dua lapisan (layer), yaitu lapisan input dan lapisan output. Setiap neuron dalam lapisan input terhubung dengan setiap neuron pada lapisan output. Setiap neuron

dalam lapisan output merepresentasikan kelas (cluster) dari input yang diberikan.



Gambar 10.1: Arsitektur JST Self Organizing Map (Hardiansyah and Primandari, 2014)

Prinsip kerja algoritma SOM adalah pengurangan node-node tetangganya (neighbor), sehingga pada akhirnya adalah satu keluaran node yang terpilih (Munawar, 2015). Berikut ini adalah tahapan algoritma dari metode SOM:

Langkah 0: Inisialisasi pembobotan wij dengan nilai random. Menset parameter learning rate(α), pengurangan learning rate (β) dan MaxEpoch.

Langkah 1: Apabila kondisi selesai belum terpenuhi, lakukan langkah berikut

a. Untuk tiap j (j=1,…n), hitung:

$$D(j) = \sum_{i=1}^{n} \left(w_{ij} - x_i\right)^2$$

b. Cari indexs j yang membuat D(j) bernilai minimum.
c. Lakukan perbaikan nilai wij dengan nilai tertentu, yaitu:

wij(baru)= wij(lama)+α(xi-wij(lama))

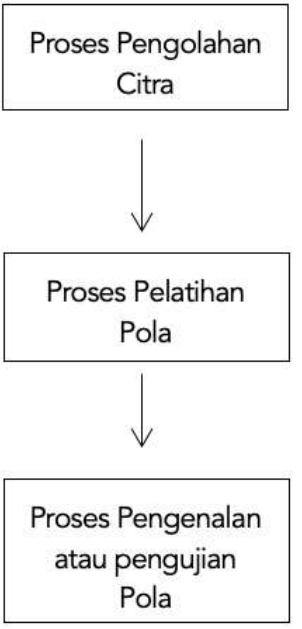
Keterangan:
wij(baru) : Bobot matriks baru.
wij(lama): Bobot matriks lama.
α: Learning rate/ alfa.
xi: Nilai setiap vektor input.

d. Update learning rate dari self-organized maps.
e. Cek kondisi stopnya, simpan bobot akhir.

10.2 Contoh Soal dan Penyelesaiannya

Algoritma kohonen self organizing map adalah sebuah metode yang dapat digunakan dalam sebuah jaringan saraf tiruan sebagai proses pengenalan pola huruf aksara lontara Bugis. Untuk pengenalan pola huruf aksara lontara Bugis menggunakan kohonen self organizing map, hal pertama terlebih dahulu dilakukan adalah menganalisa dengan menggunakan pengolahan citra, pengolahan citra tersebut dapat mengubah citra gambar menjadi biner (Threshold), hasil tersebut akan dilatih dan diuji dalam sebuah sistem yang dibangun, sehingga pola yang dilatih dan diuji dapat dikenali dalam jaringan saraf tiruan.

Proses pengenalan pola pada sistem harus melalui beberapa tahapan agar pola dapat dikenali oleh sistem. Berikut ini adalah tahapan algoritma sistem.



Gambar 10.2: Tahapan Algoritma Sistem.

Berikut ini adalah tabel bentuk huruf aksara lontara Bugis konsonan dan vocal yang akan dirubah kedalam citra biner.

Tabel 10.1: Huruf Konsonan

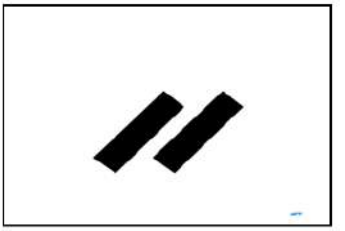
No	**Aksara Lontara**	**Latin**	**No**	**Aksara Lontara**	**Latin**
1	k	Ka	13	C	Ca
2	g	Ga	14	j	Ja
3	G	Nga	15	N	Nya
4	K	Ngka	16	C	Nca
5	p	Pa	17	y	Ya
6	b	Ba	18	R	Ra
7	m	Ma	19	l	La
8	P	Mpa	20	w	Wa
9	t	Ta	21	s	Sa
10	d	Da	22	a	A
11	N	Na	23	h	Ha
12	R	Nra			

Tabel 10.2: Huruf Vocal

No	Aksasara Lontara	Latin	No	Aksasara Lontara	Latin
1.		I	4.		ə

2.		U	5.		O
3.		E			

Berikut ini adalah contoh huruf konsonan aksara lontara Bugis yang akan diproses melalui beberapa tahapan pengolahan citra:



Gambar 10.3: Aksara Lontara "Ka"

Proses akuisisi citra merupakan proses untuk mengambil data citra digital huruf konsonan aksara lontara Bugis, setelah mengambil data citra huruf konsonan aksara lontara Bugis kemudian dilakukan proses Cropping citra, Cropping citra merupakan proses pemotongan pada koordinat tertentu pada area citra, membuang bagian area kosong yang tidak dibutuhkan. Kemudian dilakukan proses pengskalaan citra, pengskalaan citra berfungsi untuk mengubah ukuran citra menjadi 20 x 20 kotak piksel setelah itu proses mengubah citra gambar menjadi citra biner (Thersold) dengan cara menghitung setiap piksel, jika piksel bernilai lebih kecil dari 127 maka biner bernilai 0 (hitam), jika piksel berinilai lebih besar dari 127 maka biner bernilai 1 (putih).

Berikut ini adalah contoh huruf aksara lontara Bugis yang sudah diubah dalam bentuk 400 bit biner. Huruf ini akan dijadikan sebagai contoh pelatihan dan pengujian menggunakan algoritma kohonen self organizing map.

Tabel 10.3: *Bentuk Aksara Lontara Bugis dalam Biner*

Huruf aksara Lontara Bugis	Bentuk Biner Matrik 20 x 20
 Huruf Aksara Lontara Bugis "Ka"	11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111100111001111 11111111000110001111 11111110001100011111 11111100011000111111 11111000110001111111 11110001100011111111 11111011110111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111

Pada Proses pelatihan, setiap pola huruf baru dimasukkan dan dikenalkan ke sistem. Hasil dari proses ini adalah nilai bobot pelatihan yang akan diuji pada proses pengenalan. Nilai bobot ini akan digunakan untuk mengetahui jarak setiap pola huruf baru terhadap setiap

kelompok yang ada sehingga diketahui pola huruf yang memiliki kemiripan atau memiliki jarak terkecil terhadap suatu kelompok. Sebagai contoh huruf aksara lontara yang akan dilatih ke sistem adalah 4 huruf, yaitu k "Ka" , G "Ga" , g "ga" dan b"ba" dapat dilihat pada tabel berikut ini:

Tabel 10.4: Huruf Aksara Lontara Bugis yang akan dilatih

No.	Huruf Aksra Lontara Bugis	Citra Biner 20 x 20
1.	 Huruf Aksara Lontara Bugis "Ka"	11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111101111011111 11111110000110001111 11111110001100011111 11111100011000111111 11111000110001111111 11110001100011111111 11111011110111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111

2.	 Huruf Aksara Lontara Bugis "Nga"	11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111011111111111 11111111001111111111 11111111100111111111 11111111000011111111 11111110001001111111 11111100001100111111 11111110011110011111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111

No.	Huruf Aksra Lontara Bugis	Citra Biner 20 x 20
3.	 Huruf Aksara Lontara Bugis “Ga”	11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111101111111 11111111111110111111 11111110111111011111 11111100001111101111 11111000000111110111 11110001110011111011 11100011111001110001 11000111011100100001 10001110001110000011 11011111011111000111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111

4.	 Huruf Aksara Lontara Bugis "Ba"	11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111100011111111 11111111000001111111 11111110001100111111 11111110011110011111 11111110011111111111 11111111001111111111 11111110000111111111 11111100000011111111 11111000011001111111 11111100111100111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111

Berikut adalah proses pelatihan kohonen self organizing map dari ke empat huruf yang sudah di rubah dalam bentuk biner:

1. Inisialisi bobot matriks, alfa, maks iterasi/epoch, pengurangan learning rate setiap kenaikan epoh, jumlah cluster dan nilai input:

   Alfa (α) = 0,6

   MaxEpoh = 10

   Pengurangan α = 0,5(α)

   Bobot matriks/ Weight (w) = 0,5

   Jumlah Cluster (j) = 4

2. Menghitung nilai jarak setiap data

$$D(j) = \sum_{i=1}^{n} \left(w_{ij} - x_i\right)^2$$

Keterangan :
D(j): Jarak terhadap setiap Cluster
wij: Bobot matriks
Xi: nilai setiap vektor input

3. Perbaharui bobot pada Cluster dengan nilai jarak terkecil:

wij(baru)= wij(lama)+α(xi-wij(lama))

keterangan:
wij(baru) : Bobot matriks baru.
wij(lama): Bobot matriks lama.
α: Learning rate/ alfa.
xi: Nilai setiap vektor input.
Input Data 1 Huruf k " ka "(x) :

```
1111111111111111111
1111111111111111111
1111111111111111111
1111111111111111111
1111111111111111111
1111111111111111111
1111111111111111111
1111111100111001111
1111111100011000111
1111111000110001111
1111110001100011111
1111100011000111111
1111000110001111111
1110001100011111111
1111011110111111111
1111111111111111111
1111111111111111111
1111111111111111111
1111111111111111111
1111111111111111111
1111111111111111111
```

Epoch ke-1

Data ke-1 Huruf k " ka "

- bobot 1

$= (W1,1 – X1)^2 + (W12 - X2)^2 + (W13 - X3)^2 + ...+(W1,400 – X400)^2$

$= (0.50 – 1)2 + (0.50 – 1)2 + (0.50 - 1)2 + ...+(0.50 – 1)^2 =100.00$

- bobot 2

$= (W2,1 – X1)^2 + (W22 - X2)^2 + (W23 - X3)^2 + ...+(W2,400 – X400)^2$

$= (0.50 - 1)^2 + (0.50 - 1)^2 + (0.50 - 1)^2 + ...+(0.50 - 1)^2 = 100.00$

- bobot 3

$= (W3,1 – X1)^2 + (W32 - X2)^2 + (W33 - X3)^2 + ...+(W3,400 – X,400)^2$

$= (0.50 - 1)2 + (0.50 - 1)2 + (0.50 - 1)2 + ...+(0.50 - 1)^2 = 100.00$

- bobot 4

$= (W4,1 – X1)^2 + (W4,2 – X2)^2 + (W4,3 – X3)^2 + ...+(W4,400 - X400)^2$

$= (0.50 - 1)^2 + (0.50 - 1)^2 + (0.50 - 1)^2 + ...+(0.50 - 1)^2 = 100.00$

Jarak terkecil pada bobot ke-1, perbaharui bobot ke 1 dan tetangganya:

wij(baru) = wij(lama)+α(xi-wij(lama))

W1,1 = W11lama+α(X1 - W1,1)

=0.50+0.60(1 - 0.50) = 0.80

W1,2 = W1,2lama+α(X2 - W1,2)

=0.50+0.60(1 - 0.50) = 0.80

W1,3 = W1,3lama+α(X3 - W1,3)

=0.50+0.60(1 - 0.50) = 0.80

.....

W1,400 = W1,400lama+α(X400 - W1,400)

=0.50+0.60(1 - 0.50) = 0.80

revisi bobot bobot ke-2

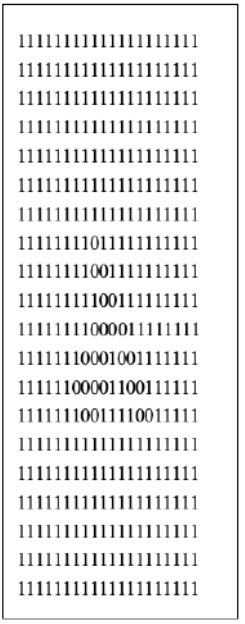
W2,1 = W2lama+α(X1 - W21)

=0.50+0.60(1 - 0.50) = 0.80

W2,2 = W2,2lama+α(X2 - W22)

=0.50+0.60(1 - 0.50) = 0.80

W2,3 = W2,3lama+α(X3 - W23)
=0.50+0.60(1 - 0.50) = 0.80
.....
W2,400 = W2,400lama+α(X400 - W2400)
=0.50+0.60(1 - 0.50) = 0.80
revisi bobot bobot ke-3
W3,1 = W3,1lama+α(X1 - W31)
=0.50+0.60(1 - 0.50) = 0.80
W3,2 = W3,2lama+α(X2 - W32)
=0.50+0.60(1 - 0.50) = 0.80
W3,3 = W3,3lama+α(X3 - W33)
=0.50+0.60(1 - 0.50) = 0.80
.....
W3,400 = W3,400lama+α(X400 - W3,400)
=0.50+0.60(1 - 0.50) = 0.80
revisi bobot Cluster ke-4
W4,1 = W41lama+α(X1 - W41)
=0.50+0.60(1 - 0.50) = 0.80
W4,2 = W42lama+α(X2 - W42)
=0.50+0.60(1 - 0.50) = 0.80
W4,3 = W43lama+α(X3 - W43)
=0.50+0.60(1 - 0.50) = 0.80
.....
W4,400 = W4,400 lama+α(X400 - W4,400)
=0.50+0.60(1 - 0.50) = 0.80
Input Data 2 Huruf G “nga”(x):



Data 2 Huruf G "nga"

- bobot 1 = $(W11 - X1)^2 + (W12 - X2)^2 + (W13 - X3)^2 + ...+(W1400 - X400)^2$
  $= (0.80 - 1)^2 + (0.80 - 1)^2 + (0.80 - 1)^2 + ...+(0.80 - 1)^2 = 40.00$
- bobot 2 = $(W21 - X1)^2 + (W22 - X2)^2 + (W23 - X3)^2 + ...+(W2400 - X400)^2$
  $= (0.80 - 1)^2 + (0.80 - 1)^2 + (0.80 - 1)^2 + ...+(0.80 - 1)^2 = 40.00$
- bobot 3 = $(W31 - X1)^2 + (W32 - X2)^2 + (W33 - X3)^2 + ...+(W3,400 - X400)^2$
  $= (0.80 - 1)^2 + (0.80 - 1)^2 + (0.80 - 1)^2 + ...+(0.80 - 1)^2 = 40.00$
- bobot 4 = $(W41 - X1)^2 + (W42 - X2)^2 + (W43 - X3)^2 + ...+(W4,400 - X400)^2$

  $= (0.80 - 1)^2 + (0.80 - 1)^2 + (0.80 - 1)^2 + ...+(0.80 - 1)^2 = 40.00$

Jarak terkecil pada bobot ke-1, revisi/perbaharui bobot ke 1 dan tetangganya:

$wij(baru) = wij(lama)+\alpha(xi-wij(lama))$

revisi bobot ke-1

$W1,1 = W1,1lama+\alpha(X1 - W11)$

$=0.80+0.60(1 - 0.80) = 0.92$

$W1,2 = W1,2lama+\alpha(X2 - W12)$

=0.80+0.60(1 - 0.80) = 0.92

W1,3 = W1,3lama+α(X3 - W13)

=0.80+0.60(1 - 0.80) = 0.92

.....

W1,400 = W1,400lama+α(X400 - W1,400)

=0.80+0.60(1 - 0.80) = 0.92

revisi bobot ke-2

W2,1 = W2,1lama+α(X1 - W2,1)

=0.80+0.60(1 - 0.80) = 0.92

W2,2 = W2,2lama+α(X2 - W2,2)

=0.80+0.60(1 - 0.80) = 0.92

W2,3 = W2,3lama+α(X3 - W2,3)

=0.80+0.60(1 - 0.80) = 0.92

.....

W2,400 = W2,400lama+α(X400 - W2400)

=0.80+0.60(1 - 0.80)= 0.92

revisi bobot ke-3

W3,1 = W3,1lama+α(X1 - W31)

=0.80+0.60(1 - 0.80) = 0.92

W3,2 = W3,2lama+α(X2 - W32)

=0.80+0.60(1 - 0.80) = 0.92

W3,3 = W3,3lama+α(X3 - W33)

=0.80+0.60(1 - 0.80) = 0.92

.....

W3,400 = W3,400lama+α(X400 - W3,400)

=0.80+0.60(1 - 0.80) = 0.92

revisi bobot ke-4

W4,1 = W41lama+α(X1 - W41)

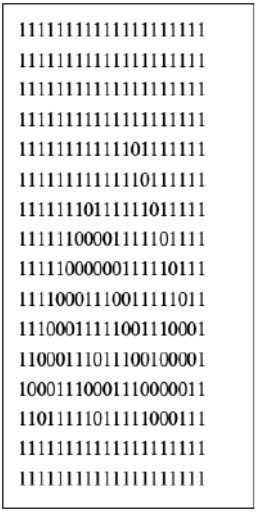
=0.80+0.60(1 - 0.80) = 0.92

W4,2 = W42lama+α(X2 - W42)

=0.80+0.60(1 - 0.80) = 0.92

W4,3 = W43lama+α(X3 - W43)
=0.80+0.60(1 - 0.80) = 0.92
.....
W4,400 = W4,400lama+α(X400 - W4,400)
=0.80+0.60(1 - 0.80) = 0.92

Input Data 3 huruf g "ga"(x):



Data 3 huruf g "ga"
- bobot 1
$= (W1,1 - X1)^2 + (W12 - X2)^2 + (W13 - X3)^2 + ...+(W1,400 - X400)^2$
$= (0.92 - 1)^2 + (0.92 - 1)^2 + (0.92 - 1)^2 + ...+(0.92 - 1)^2 = 46.96$
- bobot 2
$= (W2,1 - X1)^2 + (W22 - X2)^2 + (W23 - X3)^2 + ...+(W2,400 - X400)^2$
$= (0.92 - 1)^2 + (0.92 - 1)^2 + (0.92 - 1)^2 + ...+(0.92 - 1)^2 = 46.96$
- bobot 3
$= (W3,1 - X1)^2 + (W32 - X2)^2 + (W33 - X3)^2 + ...+(W3,400 - X400)^2$
$= (0.92 - 1)^2 + (0.92 - 1)^2 + (0.92 - 1)^2 + ...+(0.92 - 1)^2 = 46.96$
- bobot 4

$= (W4,1 - X1)^2 + (W42 - X2)^2 + (W43 - X3)^2 + ...+(W4,400 - X400)^2$
$= (0.92 - 1)^2 + (0.92 - 1)^2 + (0.92 - 1)^2 + ...+(0.92 - 1)^2 = 46.96$

Jarak terkecil pada bobot ke-1, revisi/perbaharui bobot ke 1 dan tetangganya:
wij(baru) = wij(lama)+α(xi-wij(lama))
revisi bobot ke-1
W1,1 = W11lama+α(X1 - W11)
=0.92+0.60(1 - 0.92) = 0.97
W1,2 = W12lama+α(X2 - W12)
=0.92+0.60(1 - 0.92) = 0.97
W1,3 = W13lama+α(X3 - W13)
=0.92+0.60(1 - 0.92) = 0.97
.....
W1,400 = W1400lama+α(X400 - W1400)
=0.92+0.60(1 - 0.92) = 0.97
revisi bobot ke-2
W2,1 = W2,1lama+α(X1 - W21)
=0.92+0.60(1 - 0.92) = 0.97
W2,2 = W2,2lama+α(X2 - W22)
=0.92+0.60(1 - 0.92) = 0.97
W2,3 = W2,3lama+α(X3 - W23)
=0.92+0.60(1 - 0.92) = 0.97
.....
W2,400 = W2,400lama+α(X400 - W2,400)
=0.92+0.60(1 - 0.92) = 0.97
revisi bobot ke-3
W3,1 = W3,1lama+α(X1 - W31)
=0.92+0.60(1 - 0.92) = 0.97
W3,2 = W3,2lama+α(X2 - W32)
=0.92+0.60(1 - 0.92) = 0.97
W3,3 = W3,3lama+α(X3 - W33)

=0.92+0.60(1 - 0.92) = 0.97

.....

W3,400 = W3,400lama+α(X400 - W3,400)

=0.92+0.60(1 - 0.92) = 0.97

revisi bobot ke-4

W4,1 = W41lama+α(X1 - W41)

=0.92+0.60(1 - 0.92) = 0.97

W4,2 = W42lama+α(X2 - W42)

=0.92+0.60(1 - 0.92) = 0.97

W4,3 = W43lama+α(X3 - W43)

=0.92+0.60(1 - 0.92) = 0.97

.....

W4,400 = W4,400 lama+α(X400 - W4,400)

=0.92+0.60(1 - 0.92) = 0.97

Input Data 4 Huruf b"ba"(x):

```
11111111111111111111
11111111111111111111
11111111111111111111
11111111111111111111
11111111100011111111
11111111000001111111
11111110001100111111
11111110011110011111
11111110011111111111
11111111001111111111
11111110000111111111
11111100000011111111
11111000011001111111
11111100111100111111
11111111111111111111
11111111111111111111
```

Data 4 Huruf b"ba"

- bobot 1

$= (W11 - X1)^2 + (W12 - X2)^2 + (W13 - X3)^2 + ...+(W1400 - X400)^2$

$= (0.97 - 1)^2 + (0.97 - 1)^2 + (0.97 - 1)^2 + ...+(0.97 - 1)^2 = 48.87$

- bobot 2

$= (W21 - X1)^2 + (W22 - X2)^2 + (W23 - X3)^2 + ...+(W2400 - X400)^2$

$= (0.97 - 1)^2 + (0.97 - 1)^2 + (0.97 - 1)^2 + ...+(0.97 - 1)^2 = 48.87$

- bobot 3

$= (W31 - X1)^2 + (W32 - X2)^2 + (W33 - X3)^2 + ...+(W3,400 - X400)^2$

$= (0.97 - 1)^2 + (0.97 - 1)^2 + (0.97 - 1)^2 + ...+(0.97 - 1)^2 = 48.87$

- bobot 4

$= (W41 - X1)^2 + (W42 - X2)^2 + (W43 - X3)^2 + ...+(W4,400 - X400)^2$

$= (0.97 - 1)^2 + (0.97 - 1)^2 + (0.97 - 1)^2 + ...+(0.97 - 1)^2 = 48.87$

jarak terkecil pada bobot ke-1, perbaharui bobot ke-1dan tetangganya :

revisi bobot ke-1

$W1,1 = W11lama+\alpha(X1 - W11)$

$=0.97+0.60(1 - 0.97) = 0.99$

$W1,2 = W12lama+\alpha(X2 - W12)$

$=0.97+0.60(1 - 0.97) = 0.99$

$W1,3 = W13lama+\alpha(X3 - W13)$

$=0.97+0.60(1 - 0.97) = 0.99$

.....

$W1,400 = W1400lama+\alpha(X400 - W1400)$

$=0.97+0.60(1 - 0.97) = 0.99$

revisi bobot Cluster ke-2

$W2,1 = W2,1lama+\alpha(X1 - W21)$

$=0.97+0.60(1 - 0.97) = 0.99$

$W2,2 = W2,2lama+\alpha(X2 - W22)$

$=0.97+0.60(1 - 0.97) = 0.99$

$W2,3 = W2,3lama+\alpha(X3 - W23)$

$=0.97+0.60(1 - 0.97) = 0.99$

....

$W2|400 = W2,400lama+\alpha(X400 - W2400)$

$=0.97+0.60(1 - 0.97) = 0.99$

revisi bobot Cluster ke-3

$W3,1 = W3,1lama+\alpha(X1 - W31)$

$=0.97+0.60(1 - 0.97) = 0.99$

$W3,2 = W3,2lama+\alpha(X2 - W32)$

$=0.97+0.60(1 - 0.97) = 0.99$

$W3,3 = W3,3lama+\alpha(X3 - W33)$

$=0.97+0.60(1 - 0.97) = 0.99$

....

$W3,400 = W3,400lama+\alpha(X400 - W3,400)$

$=0.97+0.60(1 - 0.97) = 0.99$

revisi bobot Cluster ke-4

$W4,1 = W41lama+\alpha(X1 - W41)$

$=0.97+0.60(1 - 0.97) = 0.99$

$W4,2 = W42lama+\alpha(X2 - W42)$

$=0.97+0.60(1 - 0.97) = 0.99$

$W4,3 = W43lama+\alpha(X3 - W43)$

$=0.97+0.60(1 - 0.97) = 0.99$

....

$W4,400 = W4,400\ lama+\alpha(X400 - W4,400\ )$

$=0.97+0.60(1 - 0.97) = 0.99$

Alpha baru = 0.60 * 0.65 = 0.39

Proses pelatihan dilakukan hingga mencapai epoch ke-10. Setelah pelatihan dilakukan sampai epoch ke-10 maka didapat nilai bobot akhir setiap Cluster sebagai berikut :

Hasil bobot cluster ke-1

1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00

1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00

1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00

1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00

1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.70 0.70 0.70 0.75 1.00 1.00 1.00 1.00 1.00 1.00 1.00

1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.70 0.70 0.70 0.70 0.70 0.75 1.00 1.00 1.00 1.00 1.00 1.00

1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.45 0.70 0.70 1.00 1.00 0.70 0.70 0.75 1.00 1.00 1.00 1.00 1.00

```
1.00 1.00 1.00 1.00 1.00 1.00 0.75 0.45 0.23 0.52 0.77 1.00 1.00 0.70 0.48 0.52 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 0.75 0.75 0.45 0.00 0.30 0.52 1.00 1.00 0.77 0.77 0.77 0.75 1.00 1.00 1.00
1.00 1.00 1.00 1.00 0.75 0.75 0.75 0.77 0.48 0.25 0.52 0.75 0.77 0.77 0.77 1.00 1.00 0.75 1.00 1.00
1.00 1.00 1.00 0.75 0.75 0.75 0.77 0.48 0.25 0.48 0.48 0.30 0.52 0.77 1.00 1.00 0.75 0.75 0.75 1.00
1.00 1.00 0.75 0.75 0.75 0.77 0.48 0.25 0.23 0.48 0.48 0.25 0.30 0.75 1.00 0.75 0.75 0.75 0.75 1.00
1.00 0.75 0.75 0.75 0.77 0.48 0.25 0.23 0.23 0.30 0.77 0.48 0.48 0.52 0.75 0.75 0.75 0.75 1.00 1.00
1.00 1.00 0.75 1.00 1.00 0.77 0.70 0.48 0.52 1.00 0.77 1.00 0.70 0.48 0.52 0.75 0.75 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
```

Hasil bobot cluster ke-2

```
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.70 0.70 0.70 0.75 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.70 0.70 0.70 0.70 0.70 0.75 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.45 0.70 0.70 1.00 1.00 0.70 0.70 0.75 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 0.75 0.45 0.20 0.55 0.80 1.00 1.00 0.70 0.51 0.55 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 0.75 0.75 0.45 0.00 0.30 0.55 1.00 1.00 0.80 0.80 0.80 0.75 1.00 1.00 1.00
1.00 1.00 1.00 1.00 0.75 0.75 0.75 0.80 0.51 0.25 0.49 0.75 0.80 0.80 0.80 1.00 1.00 0.75 1.00 1.00
1.00 1.00 1.00 0.75 0.75 0.75 0.80 0.51 0.25 0.45 0.45 0.30 0.55 0.80 1.00 1.00 0.75 0.75 0.75 1.00
1.00 1.00 0.75 0.75 0.75 0.80 0.51 0.25 0.20 0.45 0.51 0.25 0.30 0.75 1.00 0.75 0.75 0.75 0.75 1.00
1.00 0.75 0.75 0.75 0.80 0.51 0.25 0.20 0.20 0.30 0.80 0.51 0.45 0.49 0.75 0.75 0.75 0.75 1.00 1.00
1.00 1.00 0.75 1.00 1.00 0.80 0.70 0.45 0.49 1.00 0.80 1.00 0.70 0.45 0.49 0.75 0.75 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
```

Hasil bobot cluster ke-3 :

```
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.70 0.70 0.70 0.71 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.70 0.70 0.70 0.70 0.70 0.71 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.42 0.70 0.70 1.00 1.00 0.70 0.70 0.71 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 0.71 0.42 0.20 0.52 0.80 1.00 1.00 0.70 0.51 0.52 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 0.71 0.71 0.42 0.00 0.30 0.52 1.00 1.00 0.80 0.80 0.80 0.71 1.00 1.00 1.00
1.00 1.00 1.00 1.00 0.71 0.71 0.71 0.80 0.51 0.29 0.49 0.71 0.80 0.80 0.80 1.00 1.00 0.71 1.00 1.00
1.00 1.00 1.00 0.71 0.71 0.71 0.80 0.51 0.29 0.48 0.48 0.30 0.52 0.80 1.00 1.00 0.71 0.71 0.71 1.00
1.00 1.00 0.71 0.71 0.71 0.80 0.51 0.29 0.20 0.48 0.51 0.29 0.30 0.71 1.00 0.71 0.71 0.71 0.71 1.00
1.00 0.71 0.71 0.71 0.80 0.51 0.29 0.20 0.20 0.30 0.80 0.51 0.48 0.49 0.71 0.71 0.71 0.71 1.00 1.00
1.00 1.00 0.71 1.00 1.00 0.80 0.70 0.48 0.49 1.00 0.80 1.00 0.70 0.48 0.49 0.71 0.71 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
```

```
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
```

Hasil bobot cluster ke-4 :

```
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.67 0.67 0.67 0.75 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.67 0.67 0.67 0.67 0.67 0.75 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.42 0.67 0.67 1.00 1.00 0.67 0.67 0.75 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 0.75 0.42 0.20 0.55 0.80 1.00 1.00 0.67 0.48 0.55 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 0.75 0.75 0.42 0.00 0.33 0.55 1.00 1.00 0.80 0.80 0.80 0.75 1.00 1.00 1.00
1.00 1.00 1.00 1.00 0.75 0.75 0.75 0.80 0.48 0.25 0.52 0.75 0.80 0.80 0.80 1.00 1.00 0.75 1.00 1.00
1.00 1.00 1.00 0.75 0.75 0.75 0.80 0.48 0.25 0.45 0.45 0.33 0.55 0.80 1.00 1.00 0.75 0.75 0.75 1.00
1.00 1.00 0.75 0.75 0.75 0.80 0.48 0.25 0.20 0.45 0.48 0.25 0.33 0.75 1.00 0.75 0.75 0.75 0.75 1.00
1.00 0.75 0.75 0.75 0.80 0.48 0.25 0.20 0.20 0.33 0.80 0.48 0.45 0.52 0.75 0.75 0.75 0.75 1.00 1.00
1.00 1.00 0.75 1.00 1.00 0.80 0.67 0.45 0.52 1.00 0.80 1.00 0.67 0.45 0.52 0.75 0.75 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
```

Selanjutnya nilai bobot akhir akan digunakan untuk proses pengujian dan pengenalan pola huruf yang telah dilatih. Namun sebelum itu proses selanjutnya adalah mengetahui setiap data pola huruf yang telah dilatih termasuk cluster berapa yaitu dengan cara menghitung jarak setiap cluster terhadap pola huruf yang telah dilatih sebelumnya:

Data ke-1:

- bobot 1

$= (W11 - X1)^2 + (W12 - X2)^2 + (W13 - X3)^2 + ...+(W1400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 0)^2 = 41.77$

- bobot 2

$= (W21 - X1)^2 + (W22 - X2)^2 + (W23 - X3)^2 + ...+(W2400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 0)^2 = 3.38$

- bobot 3

$= (W31 - X1)^2 + (W32 - X2)^2 + (W33 - X3)^2 + ...+(W3,400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 0)^2 = 43.85$

- bobot 4

$= (W41 - X1)^2 + (W42 - X2)^2 + (W43 - X3)^2 + ...+(W4,400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 0)^2 = 43.52$

Ternyata data ke-1 lebih dekat/jarak kecil terhadap bobot ke-1 maka data tersebut termasuk dalam cluster pertama.

Data ke-2:

- bobot 1

$= (W11 - X1)^2 + (W12 - X2)^2 + (W13 - X3)^2 + ...+(W1400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 11.56$

- bobot 2

$= (W21 - X1)^2 + (W22 - X2)^2 + (W23 - X3)^2 + ...+(W2400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 10.66$

- bobot 3

$= (W31 - X1)^2 + (W32 - X2)^2 + (W33 - X3)^2 + ...+(W3,400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 11.76$

- bobot 4

$= (W41 - X1)^2 + (W42 - X2)^2 + (W43 - X3)^2 + ...+(W4,400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 11.49$

Ternyata data ke-2 lebih dekat/jarak kecil terhadap bobot ke-2 maka data tersebut termasuk dalam cluster kedua.

Data ke-3

- bobot 1

$= (W11 - X1)^2 + (W12 - X2)^2 + (W13 - X3)^2 + ...+(W1400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 31.76$

- bobot 2

$= (W21 - X1)^2 + (W22 - X2)^2 + (W23 - X3)^2 + ...+(W2400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 31.49$

- bobot 3

$= (W31 - X1)^2 + (W32 - X2)^2 + (W33 - X3)^2 + ...+(W3,400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 29.20$

- bobot 4

$= (W41 - X1)^2 + (W42 - X2)^2 + (W43 - X3)^2 + ...+(W4,400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 32.13$

Ternyata data ke-3 lebih dekat/jarak kecil terhadap bobot ke-3 maka data tersebut termasuk dalam cluster ketiga.

Data ke-4

- bobot 1

$= (W11 - X1)^2 + (W12 - X2)^2 + (W13 - X3)^2 + ...+(W1400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1^{)2} = 18.44$

- bobot 2

$= (W21 - X1)^2 + (W22 - X2)^2 + (W23 - X3)^2 + ...+(W2400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 18.24$

- bobot 3

$= (W31 - X1)^2 + (W32 - X2)^2 + (W33 - X3)^2 + ...+(W3,400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 19.14$

- bobot 4

$= (W41 - X1)^2 + (W42 - X2)^2 + (W43 - X3)^2 + ...+(W4,400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 17.00$

Ternyata data ke-4 lebih dekat/jarak kecil terhadap bobot ke-4 maka data tersebut termasuk dalam cluster empat. Pengelompokan data huruf dapat dilihat pada tabel 10.5

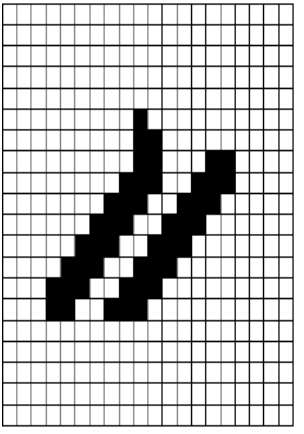
Tabel 10.5: Hasil Pengelompokan Data

Cluster	Huruf Aksara Lontara
1.	k"ka”
2.	g "nga”
3.	G “ga”
4.	b “Ba”

Proses Pengujian Pola

Dalam proses Pengujian cluster yang dipilih sebagai pemenang adalah cluster yang mempunyai vector bobot paling cocok dengan pola input (memiliki jarak paling dekat). Pola yang akan diuji adalah pola yang sebelumnya belum dilatih atau belum dikenali.

Tabel 10.6: Pola Uji Bentuk Biner

Pola Belum Yang di uji	**Citra Biner 20x20**
	11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111101111111111 11111111100111111111 11111111100111001111 11111111000110001111 11111110001100011111 11111100011000111111 11111000110001111111 11110001100011111111 11100011000111111111 11100110001111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111 11111111111111111111

Untuk proses pengujian pola huruf baru yang akan dikenali yaitu dengan cara menghitung nilai jarak setiap input terhadap Cluster(j):

$$D(j) = \sum_{i=1}^{n} \left(w_{ij} - x_i\right)^2$$

Input data Huruf baru(x) :

```
11111111111111111111
11111111111111111111
11111111111111111111
11111111111111111111
11111111111111111111
11111111101111111111
11111111100111111111
11111111100111001111
11111111000110001111
11111110001100011111
11111100011000111111
11111000110001111111
11110001100011111111
11100011000111111111
11100110001111111111
11111111111111111111
```

Jarak pada:

- bobot 1

$= (W11 - X1)^2 + (W12 - X2)^2 + (W13 - X3)^2 + ...+(W1400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 33.48$

- bobot 2

$= (W21 - X1)^2 + (W22 - X2)^2 + (W23 - X3)^2 + ...+(W2400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 35.03$

- bobot 3

$= (W31 - X1)^2 + (W32 - X2)^2 + (W33 - X3)^2 + ...+(W3,400 - X400)^2$

$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 35.42$

- bobot 4

$= (W41 - X1)^2 + (W42 - X2)^2 + (W43 - X3)^2 + ...+(W4,400 - X400)^2$
$= (1.00 - 1)^2 + (1.00 - 1)^2 + (1.00 - 1)^2 + ...+(1.00 - 1)^2 = 35.14$

Setelah melakukan proses pelatihan data baru, ternyata data tersebut lebih dekat terhadap bobot ke-1 maka data tersebut termasuk cluster pertama, dimana cluster pertama adalah huruf k "ka".

Daftar Pustaka

Adnan, R. et al. (2012) "Flood water level modelling and prediction using artificial neural network: Case study of Sungai Batu Pahat in Johor," Proceedings - 2012 IEEE Control and System Graduate Research Colloquium, ICSGRC 2012, (Icsgrc), hal. 22–25. doi: 10.1109/ICSGRC.2012.6287127.

Afriliansyah, T. et al. (2019) 'Implementation of Bayesian Regulation Algorithm for Estimation of Production Index Level Micro and Small Industry', Journal of Physics: Conference Series, 1255(1), pp. 1–6.

Alifanov and M, O. (1994) Inverse Heat Transfer Problems. Springer Verlag.

Ambarwati (2014) 'Pengelompokan Berita Indonesia Berdasarkan Histogram Kata Menggunakan Self-Organizing Map', IJCCS, 8(1).

Amin, M. H. et al. (2018) 'Quantum Boltzmann Machine', Physical Review X, 8(2). doi: 10.1103/PhysRevX.8.021050.

Amrin, A. (2016) "Analisa Komparasi Neural Network Backpropagation Dan Multiple Linear Regression Untuk Peramalan Tingkat Inflasi," Jurnal Teknik Komputer AMIK BSI, 2(2), hal. 2442–2436. doi: https://doi.org/10.31294/jtk.v2i2.1591.

Berisha, S. and Nagy, J. G. (2014) 'Iterative Methods for Image Restoration', in Academic Press Library in Signal Processing. Elsevier Masson SAS, pp. 193–247.

Bhawika, G. W. et al. (2019) 'Implementation of ANN for Predicting the Percentage of Illiteracy in Indonesia by Age Group', Journal of Physics: Conference Series, 1255(1), pp. 1–6.

Budiharjo et al. (2018a) "Predicting School Participation in Indonesia using Back-Propagation Algorithm Model," International Journal of Control and Automation, 11(11), hal. 57–68.

Budiharjo et al. (2018b) "Predicting tuition fee payment problem using backpropagation neural network model," International Journal of Advanced Science and Technology, 120, hal. 85–96. doi: 10.14257/ijast.2018.120.07.

da Silva, A. J., Ludermir, T. B. and de Oliveira, W. R. (2016) 'Quantum perceptron over a field and neural network architecture selection in a quantum computer', Neural Networks, 76, pp. 55–64. doi: 10.1016/j.neunet.2016.01.002.

Daskin, A. (2018) 'A quantum implementation model for artificial neural networks', Quanta, 7(1), pp. 7–18. doi: 10.12743/quanta.v7i1.65.

Daskin, A. (2019) 'A Simple Quantum Neural Net with a Periodic Activation Function', Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018, pp. 2887–2891. doi: 10.1109/SMC.2018.00491.

Dessy, W. M. dan Irawan, A. (2012) "Perbandingan Metode Jaringan Syaraf Tiruan Backpropagation Dan Learning Vector Quantization Pada Pengenalan Wajah," Jurnal Komputer dan Informatika, 1(1), hal. 45–51.

Farhi, E. and Neven, H. (2018) 'Classification with Quantum Neural Networks on Near Term Processors', pp. 1–21.

Fausett, L., 1994. Fundamentals of Neural Networks Architectures, Algorithms, and Aplications. ISBN 0-13-334186-0. Neural Networkas (Computer Science). Prentice Hall, Inc. Englewood Cliffs, New Jersey 07632.

Febriadi, B. et al. (2018) "Bipolar function in backpropagation algorithm in predicting Indonesia's coal exports by major destination countries," IOP Conference Series: Materials Science and Engineering, 420(012089), hal. 1–9. doi: 10.1088/1757-899X/420/1/012087.

Fletcher, R. and Reeves, C. M. (1964) 'Function minimization by conjugate gradients', The Computer Journal, pp. 149–154.

Fu, L.M., 1994, Neural Network in Computer Intelligence, International Edition, McGraw Hill.

Fuller, R. , 1995, Neural Fuzzy Systems, Abo Akademi University, Abo, Turki.

G, K. K., Widagda, I. G. A. and Suarbawa, K. N. (2018) 'Human Voice Recognition by Using Hebb Artificial Neural Network Method', Buletin Fisika, 19(1), p. 16. doi: 10.24843/bf.2018.v19.i01.p04.

Gonçalves, C. P. (2016) 'Quantum Neural Machine Learning - Backpropagation and Dynamics', pp. 1–48.

Graf, S. (2009) 'Advanced adaptivity in learning management systems by considering learning styles', Proceedings of the 2009 IEEE/WIC/ACM International …, 3(December 2007), pp. 235–238. doi: 10.1109/WI-IAT.2009.271.

Gultom, L. M. (2017) 'Klasifikasi Data Dengan Quantum Perceptron', Teknovasi, 4(1), pp. 1–9.

Hagan, M. T., Demuth, H. B. and Beale, M. (1996) Neural Network Design. Boston: MA: PWS Publishing.

Hanafi, M, 2011, Implementasi Sistem Hybrid Menggunakan Fuzzy Neural Network untuk Memperbaiki Kinerja Sistem Kontrol Suspensi Aktif Berbasis Logika Fuzzy, Prossiding Seminar Teknik Informatika. 2011 (STI 2011), edisi 1 Juli 2011, Universitas Ahmad Dahlan , Yogyakarta

Hardiansyah, B. and Primandari, P. N. (2014) 'Sistem Pakar Pengenalan Ekspresi Wajah Manusia Menggunakan Metode Kohonen Self Organizing Dan Principal Componen Analysis', (Irawan 2008), pp. 43–54.

Hariri, F. R. and Pamungkas, D. P. (2016) 'Self Organizing Map-Neural Network untuk Pengelompokan Abstrak', Citec Journal, pp. 160–169.

Hasibuan, M. S. and Nugroho, L. (2016) 'Detecting Learning Style Using Hybrid Model', 2016 IEEE Conference on e-Learning, e-Management and e-Services (IC3e) Detecting, pp. 107–111.

Hasibuan, M. S. and Purbo, O. W. (2018) 'Learning Motivation increased due to a Relaxed Assessment in a Competitive e-Learning Environment', pp. 7–10.

Hestenes, M. R. and Stiefel, E. (1952) 'Methods of conjugate gradients for solving linear systems', Journal of Research of the National Bureau of Standards, 49(6), p. 409.

Hidayat, R. N., Isnanto, R. R. and Nurhayati, O. D. (2013) 'Implementasi Jaringan Syaraf Tiruan Perambatan Balik untuk Memprediksi Harga Logam Mulia Emas Menggunakan Algoritma Lavenberg Marquardt', Jurnal Teknologi dan Sistem Komputer. doi: 10.14710/jtsiskom.1.2.2013.49-55.

Jain, L.C., Martin N.M. 1998. Fusion of Neural Networks Fuzzy Systems and Genetic Algorithms: Industrial Applications. CRC Press.

Kasabov, N.K. Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering A Bradford book. Includes bibliographical references and index. ISBN 0-262-11212-4 (hc: alk. paper), QA76.76.E95K375 1996, 006.3—dc20 95-50054

Kasabov, N.K., 1998, Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering, Massachusetts Institute of Technology, England.

Khairani, M. (2014) "Improvisasi Backpropagation menggunakan penerapan adaptive learning rate dan parallel training," TECHSI - Jurnal Penelitian Teknik Informatika, 4(1), hal. 157–172.

Killoran, N. et al. (2018) 'Continuous-variable quantum neural networks', pp. 1–21.

Koesriputranto, A. (2015) PREDIKSI HARGA SAHAM DI INDONESIA DENGAN MENGGUNAKAN METODE HYBRID PRINCIPAL COMPONENT ANALYSIS DAN SUPPORT VECTOR MACHINE (PCA-SVM).

Kopczyk, D. (2018) 'Quantum machine learning for data scientists'. Available at: http://arxiv.org/abs/1804.10068.

Lakhmi C. Jain,N.M. Martin, 1998, "Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications", CRC Press.

Lipantri Mashur Gultom (2015) Analisis Konvergensi Pada Entangled Neural Network, Universitas Sumatera Utara. doi: 10.1145/3132847.3132886.

Lubis, M. R. (2017) 'Metode Hybrid Particle Swarm Optimization - Neural Network Backpropagation Untuk Prediksi Hasil Pertandingan Sepak Bola', Jurnal Sains Komputer & Informatika (J-Sakti), 1(1), pp. 71–83.

Lubis, M. R. and Parlina, I. (2019) 'Analisis Algoritma Backpropagation Dalam', Kumpulan Jurnal Ilmu Komputer (KLik), 06(03), pp. 264–274.

Lukito, Y. (2017) 'Model Multi Layer Perceptron untuk Indoor Positioning System Berbasis Wi-Fi', Jurnal Teknologi dan Sistem Komputer, 5(3), p. 123. doi: 10.14710/jtsiskom.5.3.2017.123-128.

Mansur, M., Prahasto, T. dan Farikhin, F. (2014) "Particle Swarm Optimization Untuk Sistem Informasi Penjadwalan Resource Di Perguruan Tinggi," Jurnal Sistem Informasi Bisnis, 4(1), hal. 11–19. doi: 10.21456/vol4iss1pp11-19.

MathWorks (1994) traincgp, The MathWorks, Inc. Available at: https://www.mathworks.com/help/deeplearning/ref/traincgp.html (Accessed: 23 April 2020).

Maulana, R. dan Kumalasari, D. (2019) "Analisis Dan Perbandingan Algoritma Data Mining Dalam Prediksi Harga Saham Ggrm," Jurnal Informatika Kaputama (JIK), 3(1), hal. 22–28.

McClarren, R. G. (2018) 'Iterative Methods for Linear Systems', in Computational Nuclear Engineering and Radiological Science Using Python, pp. 145–172.

Merdekawati, G. I. and Ismail (2019) 'PREDIKSI CURAH HUJAN DI JAKARTA BERBASIS ALGORITMA LEVENBERG MARQUARDT', Jurnal Ilmiah Informatika Komputer. doi: 10.35760/ik.2019.v24i2.2366.

Modest, M. F. (2013) 'Inverse Radiative Heat Transfer', in Radiative Heat Transfer (Third Edition), pp. 779–802.

Muliono, R. and Hakim Lubis, J. (2018) 'Jaringan Syaraf Tiruan Pengenalan Pola Huruf Dengan Jaringan Habb', JTIK(Jurnal Teknik Informatika Kaputama), 2(1), pp. 46–50.

Munawar, G. (2015) 'Implementasi Algoritma Self Organizing Map (SOM) untuk Clustering Mahasiswa pada Matakuliah Proyek (Studi Kasus : JTK POLBAN)', IRWNS.

Mustafidah, H., Budiastanto, M. Z. and Suwarsito, S. (2019) 'Kinerja Algoritma Pelatihan Levenberg-Marquardt dalam Variasi Banyaknya Neuron pada

Lapisan Tersembunyi', JUITA : Jurnal Informatika. doi: 10.30595/juita.v7i2.5863.

Mustafidah, H., Rahmadhani, A. Y. and Harjono, H. (2019) 'Optimasi Algoritma Pelatihan Levenberg–Marquardt Berdasarkan Variasi Nilai Learning-Rate dan Jumlah Neuron dalam Lapisan Tersembunyi', JUITA : Jurnal Informatika. doi: 10.30595/juita.v7i1.4396.

Musthofa, M. U., Umma, Z. K. and Handayani, A. N. (2017) 'Analisis Jaringan Saraf Tiruan Model Perceptron Pada Pengenalan Pola Pulau di Indonesia', Jurnal Ilmiah Teknologi Informasi Asia, 11(1), p. 89. doi: 10.32815/jitika.v11i1.56.

Nikentari, N. et al. (2018) "Particle Swarm Optimization Untuk Prediksi Pasang Surut Air Optimization of Backpropagation Artificial Neural Network With Particle Swarm Optimization To Predict Tide Level," Jurnal Teknologi Informasi dan Ilmu Komputer, 5(5), hal. 605–612. doi: 10.25126/jtiik2018551055.

Nugraha, H. G. dan SN, A. (2014) "Optimasi Bobot Jaringan Syaraf Tiruan Mengunakan Particle Swarm Optimization," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 8(1), hal. 25–36. doi: 10.22146/ijccs.3492.

Ogie, R. I., Rho, J. C. dan Clarke, R. J. (2019) "Artificial Intelligence in Disaster Risk Communication: A Systematic Literature Review," 2018 5th International Conference on Information and Communication Technologies for Disaster Management, ICT-DM 2018. IEEE, hal. 1–8. doi: 10.1109/ICT-DM.2018.8636380.

Ohzeki, M. et al. (2018) 'Optimization of neural networks via finite-value quantum fluctuations', pp. 1–11.

Ozisik, M. N. and Orlande, H. R. B. (2000) Inverse Heat Transfer: Fundamentals and Applications. New York: Taylor & Francis.

Parulian, P. et al. (2019) 'Analysis of Sequential Order Incremental Methods in Predicting the Number of Victims Affected by Disasters', Journal of Physics: Conference Series, 1255(1), pp. 1–6.

Poustie, A. J. et al. (1999) 'All-optical pseudorandom number generator', Optics Communications, 159(4–6), pp. 208–214. doi: 10.1016/S0030-4018(98)00615-4.

Purba, I. S. et al. (2019) 'Accuracy Level of Backpropagation Algorithm to Predict Livestock Population of Simalungun Regency in Indonesia Accuracy Level of Backpropagation Algorithm to Predict Livestock Population of Simalungun Regency in Indonesia', Journal of Physics: Conference Series, 1255(1), pp. 1–6.

RapidMiner (2012) RapidMiner Studi o Manual. doi: http://docs.rapidminer.com/downloads/RapidMiner-v6-user-manual.pdf.

Rebentrost, P., Bromley, T. R. and Weedbrook, C. (2018) 'Quantum Hopfield neural network', Physical Review A, 98(4), pp. 1–13. doi: 10.1103/PhysRevA.98.042308.

Rouza, E., Jufri and Fimawahib, L. (2021) 'Implementasi Metode Perceptron Untuk Pengenalan Pola Jenis-Jenis Cacing Nematoda Usus', Jurnal Resti, 1(10), pp. 1–2.

Ruslan, F. A., Zakaria, N. K. dan Adnan, R. (2013) "Flood modelling using Artificial Neural Network," Proceedings - 2013 IEEE 4th Control and System Graduate Research Colloquium, ICSGRC 2013, hal. 116–120. doi: 10.1109/ICSGRC.2013.6653287.

S.K.Jain and V.P.Singh (2003) 'Systems Analysis Techniques', in Developments in Water Science, pp. 279–350.

Sahagun, M. A. M., Cruz, J. C. D. dan Garcia, R. G. (2017) "Wireless sensor nodes for flood forecasting using artificial neural network," HNICEM 2017 - 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, 2018-Janua, hal. 1–6. doi: 10.1109/HNICEM.2017.8269462.

Santoso F, 2003, Perbandingan Kinerja Sistem Kontrol Berumpan Balik (Feedback) Dengan Sistem Kontrol Berumpan Maju (Feedfoward) Pada Jaringan Penukar Panas (Heat Exchanger), Jurnal Teknik Mesin Universitas Kristen Petra, Vol. 5, No.1 April 2003, 36 – 42

Sarosa, M., Ahmad, A.S., Riyanto, B., Noer, A.S. 2007. Optimazion of Neuro-Fuzzy System Using Genetic Algorithm for Chromosome Classification. ITB J. ICT Vol.1., No.1., 56-69.

Scales, L. E. (1985) Introduction to Non-Linear Optimization. Macmillan International Higher Education.

Schuld, M. and Sinayskiy, I. (2014) 'The quest for a Quantum Neural Network'.

Sebatubun, M. M. and Nugroho, M. A. (2017) 'Ekstraksi Fitur Circularity untuk Pengenalan Varietas Kopi Arabika', Jurnal Teknologi Informasi dan Ilmu Komputer, 4(4), p. 283. doi: 10.25126/jtiik.201744505.

Seow, K.-L., Behrman, E. and Steck, J. (2015) 'Efficient learning algorithm for quantum perceptron unitary weights', pp. 1–10. Available at: http://arxiv.org/abs/1512.00522.

Shewchuk, J. R. (1994) 'Conjugate Gradient Method Without the Agonizing Pain', Science.

Siregar, E. et al. (2019) 'Analysis of Backpropagation Method with Sigmoid Bipolar and Linear Function in Prediction of Population Growth', Journal of Physics: Conference Series, 1255(1), pp. 1–6.

Sormin, M. K. Z. et al. (2019) 'Predictions of World Population Life Expectancy Using Cyclical Order Weight / Bias', Journal of Physics: Conference Series, 1255(1), pp. 1–6.

Straeter, T. A. (1971) On the Extension of the Davidon-Broyden Class of Rank One, Quasi-Newton Minimization Methods to an Infinite Dimensional Hilbert Space with Applications to Optimal Control Problems. North Carolina State Univ., Raleigh, NC, United States.

Sugiarti et al. (2019) 'An artificial neural network approach for detecting skin cancer', Telkomnika (Telecommunication Computing Electronics and Control), 17(2), pp. 788–793. doi: 10.12928/TELKOMNIKA.v17i2.9547.

Suhendra, C. D. dan Wardoyo, R. (2015) "Penentuan Arsitektur Jaringan Syaraf Tiruan Backpropagation (Bobot Awal dan Bias Awal) Menggunakan Algoritma Genetika," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 9(1), hal. 77. doi: 10.22146/ijccs.6642.

Sumijan et al. (2016) 'Implementation of Neural Networks in Predicting the Understanding Level of Students Subject', International Journal of Software Engineering and Its Applications, 10(10), pp. 189–204. Available at: http://dx.doi.org/10.14257/ijseia.2016.10.10.18.

Sumijan et al. (2016) "Implementation of Neural Networks in Predicting the Understanding Level of Students Subject," International Journal of Software Engineering and Its Applications, 10(10), hal. 189–204.

Susanti, I. (2014) Sistem Peramalan Kenaikan Permukaan Air Dengan Artificial Neural Networks Backpropagation. UNIVERSITAS BENGKULU. Available at: http://repository.unib.ac.id/9211/.

Sylvia Jane Annatje Sunarauw (2018) 'Algoritma Pelatihan Levenberg-Marquardt Backpropagation Artificial Neural Network Untuk Data Time Series', Jurnal Frontiers: Jurnal Sains dan Teknologi, Universitas Negeri Manado, 1(2), p. 213. doi: 10.36412/frontiers/001035e1/agustus201801.10.

Tambunan, F. (2019) 'Pengenalan Aksara Batak Dengan Metode Perceptron', Jurasik (Jurnal Riset Sistem Informasi dan Teknik Informatika), 4(1), p. 160. doi: 10.30645/jurasik.v4i1.129.

Tambunan, F., Yudi, Y. and Fauzi, M. (2019) 'Design of Artificial Neural Networks to Recognize Fingerprint Patterns', 3(1), pp. 58–63.

Torrontegui, E. and Ripoll, J. J. G. (2018) 'Universal quantum perceptron as efficient unitary approximators'.

Truatmoraka, P., Waraporn, N. dan Suphachotiwatana, D. (2016) "Water level prediction model using back propagation neural network: Case study: The lower of chao phraya basin," 2016 4th International Symposium on Computational and Business Intelligence, ISCBI 2016, hal. 200–205. doi: 10.1109/ISCBI.2016.7743284.

Türkpençe, D., Akıncı, T. Ç. and Serhat, Ş. (2018) 'Decoherence in a quantum neural network'.

Utomo, W. (2015) "Prediksi Nilai Ujian Nasional Produktif Sekolah Menengah Kejuruan Menggunakan Metode Neural Network," Techno.COM, 14(1), hal. 33–41.

Wanto, A. (2017) 'Optimasi Prediksi Dengan Algoritma Backpropagation Dan Conjugate Gradient Beale-Powell Restarts', Jurnal Teknologi dan Sistem Informasi, 03(03), pp. 370–380.

Wanto, A. dan Windarto, A. P. (2017) "Analisis Prediksi Indeks Harga Konsumen Berdasarkan Kelompok Kesehatan Dengan Menggunakan

Metode Backpropagation," Jurnal & Penelitian Teknik Informatika, 2(2), hal. 37–44.

Wanto, A. et al. (2019) 'Analysis of the Backpropagation Algorithm in Viewing Import Value Development Levels Based on Main Country of Origin', Journal of Physics: Conference Series, 1255(1), pp. 1–6.

Wanto, A. et al. (2019) "Model of Artificial Neural Networks in Predictions of Corn Productivity in an Effort to Overcome Imports in Indonesia," Journal of Physics: Conference Series, 1339(1). doi: 10.1088/1742-6596/1339/1/012057.

Wiebe, N., Kapoor, A. and Svore, K. M. (2016) 'Quantum perceptron models', Advances in Neural Information Processing Systems, pp. 4006–4014.

Windarto, A. P. dan Lubis, M. R. (2016) "Model Arsitektur Backpropagation Pada Prediksi Total Laba Rugi Komprehensif Bank Umum Konvensional Dalam Mendorong Laju Pertumbuhan Ekonomi," in Semrestek 2018, hal. 330–338.

Windarto, A. P., Lubis, M. R. dan Solikhun (2018) "IMPLEMENTASI JST PADA PREDIKSI TOTAL LABA RUGI KOMPREHENSIF BANK UMUM KONVENSIONAL DENGAN BACKPROPAGATION," Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), 5(4), hal. 411–418. doi: 10.25126/jtiik.201854767.

Wulandari, Y., 2011, Aplikasi Metode Mamdani dalam Penentuan Status Gizi dengan Indeks Massa Tubuh, Skripsi, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Yogyakarta, 2011 Systems.17(2): 129-138.

Yan, J., Ryan, M., Power, J., 1994, Using Fuzzy Logic, Prentice Hall.

Yana Simamora, R., Hasan, H. and Rizka, M. (2016) 'Deteksi Gangguan Lambung Melalui Citra Iris Mata Menggunakan Metode Jaringan Syaraf Tiruan Hebb Rule', Infomedia Vol.1 No. 1, 1(infomedia), pp. 44–47. Available at: https://iridology-research.com/wp-content/uploads/2018/07/1503561779DeteksiGangguanlambungs.pdf.

Yanto, M. (2017) 'Penerapan Jaringan Syaraf Tiruan Dengan Algoritma Perceptron Pada Pola Penentuan Nilai Status Kelulusan Sidang Skripsi', Jurnal Teknoif, 5(2), pp. 79–87. doi: 10.21063/jtif.2017.v5.2.79-87.

Yanto, M., Sovia, R. and Wiyata, P. (2018) 'Jaringan Syaraf Tiruan Perceptron untuk Penentuan Pola Sistem Irigasi Lahan Pertanian di Kabupaten Pesisir Selatan Sumatra Barat', Sebatik, pp. 111–115.

Ying, S., Ying, M. and Feng, Y. (2017) 'Quantum Privacy-Preserving Perceptron'.

Youllia Indrawaty, Asep Nana Hermana, A. R. (2012) "Implementasi Model Backpropagation Dalam Mengenali Pola Gambar Untuk Mendiagnose Penyakit Kulit," Jurnal informatika, 3(1), hal. 1–7.

Yu, H. and Wilamowski, B. M. (2016) 'Levenberg-marquardt training', in Intelligent Systems.

Zadeh, L.A. 1990. Fuzzy Sets And Systems. International Journal of General

Zulfadli (2019) Penerapan Algoritma Levenberg Marquardt untuk Memprediksi Luas Area Panen Padi Provinsi Riau. Universitas Islam Negeri Sultan Syarif Kasim Riau. Available at: http://repository.uin-suska.ac.id/22671/3/ZULFADLI-11451104780-LAPORAN FULL.pdf.

Biodata Penulis

Agus Perdana Windarto

Lahir di Pematangsiantar pada tanggal 30 Agustus 1986. Penulis menyelesaikan pendidikan Magister (S2) bidang Ilmu Komputer pada tahun 2014 di Universitas Putra Indonesia ''YPTK'' Padang. Saat ini penulis sedang melanjutkan program Doktor (S3) di Universitas Putra Indonesia ''YPTK'' Padang. Penulis aktif mengajar di STIKOM Tunas Bangsa sejak tahun 2012 sebagai Dosen Tetap pada program studi Sistem Informasi. Fokus penelitian yang dilakukan adalah bidang Artificial Intelligence (Sistem Pendukung Keputusan, Sistem Pakar, Data Mining, Jaringan Saraf Tiruan, Fuzzy Logic, Deep Learning dan Algortima Genetika). Penulis juga aktif menjadi reviewer di berbagai Jurnal Nasional Akreditasi (SINTA 2 – SINTA 6). Selain itu penulis mengelola Komunitas yang dineri nama Pemburu Jurnal (2017-now) dilingkungan STIKOM Tunas Bangsa. Komunitas ini terdiri dari 20 Mahasiswa/i angkatan pertama (2017-2018) dimana semua memiliki Scopus ID dan menghasilkan publikasi sebanyak kurang lebih 135 artikel ilmiah baik seminar, conference dan jurnal nasional). Angkatan kedua terdiri dari 12 Mahasiswa/i yang menghasilkan 52 publikasi artikel ilmiah baik seminar baik seminar, conference dan jurnal nasional. Penulis juga pemenang beberapa Proposal Hibah Penelitian DIKTI (2x) tahun 2018-2019, pemenang Proposal Hibah Pengabdian DIKTI (1x) tahun 2019, pemenang proposal Hibah PKM-P sebagai pembimbing mahasiswa (2018) dan pemenang proposal Hibah PKM-

AI sebagai pembimbing mahasiswa (2019). Sekarang ini penulis juga tergabung dalam komunitas Relawan Jurnal Indonesia (RJI) wilayah sumatera utara dan Forum Komunitas Perguruan Tinggi (FKPT).

Darmeli Nasution, S.Kom., M.Kom.

Lahir di Tapanuli Selatan, Sumatera Utara, Indonesia, dan merupakan putri ke-empat dari pasangan H. Muhammad Jasen Nasution (Alm) dan Hj. Siti Hasni Hasibuan (Almh). Memiliki dua orang anak bernama Nurinziva Zahra Wardhani Pulungan & Ahmad Fakhrizal Pulungan dan suami bernama Wandan Pulungan. Menyelesaikan Studi S-1 di Universitas Gunadarma Jakarta, Program Studi Manajemen Informatika pada tahun 1996 dan melanjutkan pendidikan ke jenjang Magister Komputer (S-2) di Program Pascasarjana Universitas Putra Indonesia (UPI-YPTK) Padang Sumatera Barat, Program Studi Teknologi Informasi dan lulus pada tahun 2008. Saat ini masih menempuh pendidikan (S-3) di University Sains Malaysia (USM) Pulau Pinang Malaysia Jurusan Computer Science. Tahun 2000 memulai Profesi sebagai Dosen di Fakultas Teknik Universitas Pembangunan Panca Budi Medan. Mulai Agustus 2019 penulis bertugas sebagai Dosen Tetap Pada Program Studi Teknik Telekomunikasi Dan Navigasi Udara Politeknik Penerbangan Medan.

Pernah menerbitkan Buku Ajar "Algoritma Dan Pemrograman" pada tahun 2012, serta aktip dalam publikasi Ilmiah baik Nasional maupun Internasional. Beberapa karya penelitian penulis juga telah terindeks di Google Scholar dan Scopus. Bidang ilmu yang di tekuni saat ini adalah Artificial Intellegent yaitu "Machine Learning" & "Deep Learning". Penullis juga aktip dalam beberapa organisasi keilmuan seperti APTIKOM dan Persatuan Insinyur Indonesia (PII). Penulis dapat dihubungi melalui Surel : darmelinasution@gmail.com

Anjar Wanto, M.Kom

Lahir di Bah Jambi, 14 Februari 1985. Menyelesaikan Pendidikan D3 Manajemen Informatika di AMIK Tunas Bangsa Pematangsiantar, S1 Teknik Informatika di STT Poliprofesi Medan, S2 Teknik Informatika di Universitas Sumatera Utara (USU), dan saat ini sedang melanjutkan studi S3 Teknologi Informasi di UPI YPTK Padang, Sumatera Barat. Konsentrasi keilmuan dibidang Kecerdasan Buatan (Jaringan Saraf, Sistem Pendukung Keputusan, Data Mining). Penulis merupakan peneliti dan Dosen di STIKOM Tunas Bangsa Pematangsiantar hingga saat ini. Penulis juga merupakan Reviewer di 21 Jurnal terakreditasi yang ada di Perguruan Tinggi di Indonesia, diantara nya Terakreditasi SINTA 2 (JTIIK - Universitas Brawijaya, JSINBIS - Universitas Diponegoro, JEPIN - Universitas Tanjungpura). Informasi WA : 0822-9436-5929, Email : anjarwanto@amiktunasbangsa.ac.id / anjarwanto@gmail.com

Frinto Tambunan

Lahir di Desa Simasom Kecamatan Pahae Julu, Kabupaten Tapanuli Utara, pada tanggal 14 April 1989. Penulis besar dan mulai mengenyam pendidikan Sekolah Dasar di SD N. 173268 Simasom dan selesai tahun 2001, kemudian melanjut pada SMP N. 1 Pahae Julu dan selesai tahun 2004, kemudian melanjut pada SMA N. 1 Pahae Julu dan selesai tahun 2007, Kemudian Kuliah (S1) di STMIK Potensi Utama tahun 2008 dan selesai pada tahun 2012. Penulis menyelesaikan pendidikan Magister (S2) bidang Ilmu Komputer pada tahun 2014 di Universitas Putra Indonesia ''YPTK'' Padang. Penulis aktif mengajar di Universitas Potensi Utama sejak tahun 2012 sebagai Dosen Tetap pada program studi Teknik Informatika. Penulis juga mulai tahun 2018 sampai sekarang menjadi dosen pembimbing skripsi dan pembanding di bodang Fakultas Teknik Informatika.

Fokus penelitian yang dilakukan adalah bidang Artificial Intelligence (Sistem Pendukung Keputusan, Sistem Pakar, Data Mining, Jaringan Saraf Tiruan, Fuzzy Logic, Deep Learning dan Algortima Genetika). Penulis juga sebagai pemenang hibah Penelitian Dosen Pemula pada pendanaan tahun 2015/2016, kemudian memenangkan lagi hibah Penelitian Dosen Pemula pada tahun 2016/2017.

Muhammad Said Hasibuan, M.Kom.

Dilahirkan di Rantau Prapat 12 Januari 1977. Saat ini bertugas sebagai dosen Magister Teknologi Informasi di kampus IBI Darmajaya Lampung. Menempuh Pendidikan S1 Teknik Informatika di Universitas Bina Darma Palembang, S2 Ilmu Komputer dari Universitas Gadjah Mada. Sejak tahun 2015 sedang menempuh Pendidikan doktor di Universitas Gadjah Mada dengan konsentrasi Teknologi Informasi pada Pendidikan. Menjadi konsultan dibeberapa kantor pemerintahan dalam membangun Smart City. Aktif dan menjadi pengurus di Aptikom Pusat dan Relawan TIK Indonesia sejak tahun 2011.

Muhammad Noor Hasan Siregar, S.T., M.Kom.

Lulusan S1. Teknik Industri (ST) dari Universitas Andalas Padang dan melanjutkan pendidikan di Teknik Informatika Universitas Putra Indonesia "YPTK" Padang (M.Kom).

Sekarang bekerja sebagai Dosen di Universitas Graha Nusantara Padangsidimpuan dari tahun 2011 dan pernah beberapa kali menduduki jabatan seperti, Dekan (2014), Ketua Lembaga (2017) dan Wakil Rektor (2019) di Universitas tersebut.

Informasi lebih lanjut dapat dilihat melalui blog di alamat http://hasan.dosen.ugn.ac.id dan bisa dihubungi melalui email noor.siregar@gmail.com

Muhammad Ridwan Lubis

ridwanlubis@amiktunasbangsa.ac.id

Lahir di Pematangsiantar, 26 Nopember 1986. Mulai aktif mengajar pada tahun 2010 di AMIK Tunas Bangsa dan melanjutkan kuliah program Pasca Sarjana (S2) di Universitas Sumatera Utara dan lulus pada tahun 2016. Mulai aktif melakukan penelitian dan memenangkan hibah Penelitian Dosen Pemula pada tahun pelaksanaan 2018 dan 2019 dan lulus Sertifikasi Dosen tahun 2019. Terus menulis dan mengembangkan diri dengan membiasakan melakukan kegiatan Kolaborasi Penelitian antar mahasiswa dan Dosen di bidang Ilmu Komputer seperti kecerdasan Buatan, Aplikasi Basis Data dan Bahasa Pemrograman. Mulai aktif menulis buku pada tahun 2020 dengan berkolaborasi dengan Dosen di luar Institusi dengan Buku pertama berjudul "**Biometrika: Teknologi Identifikasi**".

Solikhun, S.Kom., M.Kom.

Lahir di Pemalang pada tanggal 04 Januari 1983. Ia menyelesaikan kuliah dan mendapat gelar Sarjana Komputer pada 23 November 2008. Ia merupakan alumnus Program Studi Teknik Informatika Sekolah Tinggi Manajemen Informatika Dan Komputer Kristen Neumann Indonesia. Pada tahun 2011 mengikuti Program Magister Teknik Informatika Universitas Putra Indonesia "YPTK" Padang dan lulus pada tahun 2012. Pada tahun 2018 mengikuti Program Doktoral Ilmu Komputer Universitas Sumatera Utara. Pada tahun 2008 diangkat menjadi Dosen AMIK Tunas Bangsa dan ditempatkan di Program Studi Manajemen Informatika.

Yusra Fadhillah, S.Kom., M.Kom.

Lahir di Jakarta pada tanggal 18 Juni 1991. Menyelesaikan kuliah dan mendapat gelar Sarjana Teknik Informatika pada 02 Februari 2016. Dan merupakan salah satu alumni Jurusan Teknik Informatika pada Fakultas Ilmu Komputer Universitas Putra Indonesia "YPTK" Padang. Dan pada tahun yang sama, bekerja di Telkom Akses sebagai NOC (Network Operation Center) atau OMC (Operation and Maintenance Center) di Kota Padang. Dan pada tahun 2017 melanjutkan Program Magister Teknologi Informasi dan lulus pada tahun 24 September 2018 dari Universitas Putra Indonesia "YPTK" Padang. Pada tahun 2019 diangkat menjadi Dosen Universitas Graha Nusantara Padangsidimpuan dan ditempatkan di Fakultas Teknik pada Program Studi Ilmu Komputer atau Informatika.

Dicky Nofriansyah, S.Kom., M.Kom.

Saya dilahirkan di Medan Tanggal 31 Oktober 1989 anak ke-4 dari 4 bersaudara, dari Seorang Ibu yang bernama : Hj.Rodiah dan Ayah : H.Syamsul Bachri (alm).

Dan telah menikah dengan seorang wanita bernana: Febriani Sartika, S.Kom dikarunia 2(dua) orang anak laki-laki.

1. Assyadil Dzikri Nofriansyah (Putera Pertama)
2. Alkhalifi Kenzie Nofriansyah (Putera Kedua)

Saya Lulus dari Program Studi Strata -1 (S1) yaitu tahun 2011 di STMIK Budidarma Medan, dan Pascasarjana Magister Ilmu Komputer (S2) di Universitas Putera Indonesia YPTK Padang. Dan Program S3 Doktoral di Universitas Negeri Padang.

Pekerjaan saya saat ini adalah Dosen di STMIK Triguna Dharma Medan. Motivator terbesar saya saat ini adalah Ibu dan Istri. Secara Akademik

pencapaian terbesar saya yaitu menjadi Wisudawan Terbaik Program Pascasarjana di Universitas Putera Indonesia YPTK Padang. Dengan adanya buku ini mudah-mudahan dapat menambah ilmu pengetahuan kepada pembaca dan saya sebagai seorang penulis dapat meningkatkan kualitas bahan bacaan yang saya tulis. Aamiin ya Robbal Alamin.

Kata Mutiara :

" Kalau kita ingin bahagia di dunia dengan ilmu, kalau kita ingin bahagia di Akhirat dengan ilmu, serta kalau kita ingin bahagia Dunia Wal Akhirat juga dengan Ilmu".

Cara Mensitasi Buku ini:

[APA]

Perdana Windarto, A., Nasution, D., Wanto, A., Tambunan, F., Said Hasibuan, M., Noor Hasan Siregar, M., … Nofriansyah, D. (2020). *Jaringan Saraf Tiruan: Algoritma Prediksi dan Implementasi*. Medan: Yayasan Kita Menulis.

[HARVARD]

Perdana Windarto, A. et al. (2020) *Jaringan Saraf Tiruan: Algoritma Prediksi dan Implementasi*. Medan: Yayasan Kita Menulis.

[MLA]

Perdana Windarto, Agus, et al. *Jaringan Saraf Tiruan: Algoritma Prediksi dan Implementasi*. Yayasan Kita Menulis, 2020.

JARINGAN Saraf Tiruan

Algoritma Prediksi & Implementasi

Jaringan Saraf Tiruan merupakan topik yang hangat dibicarakan dan mengundang banyak kekaguman dalam beberapa tahun terakhir. Hal ini disebabkan karena kemampuan JST untuk meniru sifat sistem yang diinputkan.

Penyajian yang disampaikan kepada pembaca merupakan pengalaman menulis dari penulis yang dibantu dengan beberapa sumber baik literatur maupun internet yang mengarah kepada pokok pembahasan buku.

Secara garis besar, buku ini terdiri dari 10 (sepuluh) bab, yaitu :

YAYASAN KITA MENULIS
press@kitamenulis.id
www.kitamenulis.id

ISBN 978-623-7645-83-2