IMPLEMENTASI *BACKPROPAGATION NEURAL NETWORK* DALAM PEMBANGKITAN OTOMATIS FUNGSI KEANGGOTAAN *FUZZY* PADA PENDERITA PENYAKIT HEPATITIS

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

RYANDHIMAS EDO ZEZARIO

NIM. 105090601111007

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2014

LEMBAR PERSETUJUAN

IMPLEMENTASI *BACKPROPAGATION NEURAL NETWORK* DALAM PEMBANGKITAN OTOMATIS FUNGSI KEANGGOTAAN *FUZZY* PADA PENDERITA PENYAKIT HEPATITIS

SKRIPSI
KONSENTRASI KOMPUTASI CERDAS DAN VISUALISASI

Untuk memenuhi sebagian persyaratan untuk
Mencapai gelar Sarjana Komputer

Disusun Oleh:
Ryandhimas Edo Zezario
105090601111007

Telah diperiksa dan disetujui oleh
Dosen Pembimbing

Pembimbing I	Pembimbing II
Candra Dewi, S.Kom, M.Sc	**Novanto Yudistira,S.Kom,M.Sc**
NIP.19771114 200312 2 001	**NIK.831110 16 1 1 0425**

LEMBAR PENGESAHAN

IMPLEMENTASI *BACKPROPAGATION NEURAL NETWORK* DALAM PEMBANGKITAN OTOMATIS FUNGSI KEANGGOTAAN *FUZZY* PADA PENDERITA PENYAKIT HEPATITIS

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS DAN VISUALISASI

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun oleh:

RYANDHIMAS EDO ZEZARIO

NIM. 105090601111007

Skripsi ini diuji dan dinyatakan lulus pada tanggal 2 Oktober 2014

Penguji I	**Penguji II**
Edy Santoso, S.Si., M.Kom.	**Lailil Muflikhah, S.Kom., M.Sc.**
NIP. 19740414 200312 1 004	**NIP. 19741113 200501 2 001**

Penguji III

Budi Darma Setiawan, S.Kom., M.Cs.

NIK. 841015 06 1 1 0090

Mengetahui,

Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji., M.T.

NIP. 19670801 199203 1 001

PERNYATAAN

ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 danPasal 70.

Malang, 17 September 2014

Mahasiswa,

Ryandhimas Edo Zezario

NIM. 105090601111007

KATA PENGANTAR

Syukur dan alhamdulillah penulis panjatkan ke hadirat Allah SWT yang telah melimpahkan segala Rahmat, Karunia dan Hidayah-Nya sehingga Penulis dapat menyelesaikan skripsi dengan judul: ” IMPLEMENTASI *BACKPROPAGATION NEURAL NETWORK* DALAM PEMBANGKITAN OTOMATIS FUNGSI KEANGGOTAAN *FUZZY* PADA PENDERITA PENYAKIT HEPATITIS “.

Skripsi ini diajukan sebagai syarat ujian seminar skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Program Teknologi Informasi Dan Ilmu Komputer (PTIIK), Program Studi Teknik Informatika/Ilmu Komputer, Universitas Brawijaya Malang. Atas terselesaikannya skripsi ini, Penulis mengucapkan terima kasih kepada:

1. Candra Dewi, S.Kom.,M.Sc., selaku Dosen Pembimbing Skripsi pertama dan Novanto Yudistira, S.Kom., M.Sc., selaku Dosen Pembimbing Skripsi kedua yang telah meluangkan waktu dan juga memberikan pengarahan bagi penulis.
2. Mama dan Papa yang selalu memberikan doa, kasih sayang, motivasi, nasehat dan inspirasi dalam membesarkan dan mendidik penulis, serta Mas Rio yang selalu memberikan inspirasi dan semangat untuk segera menyelesaikan skripsi ini.
3. Ir. Sutrisno, MT., selaku Ketua Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
4. Drs. Marji, MT., selaku Ketua Program Studi Informatika/Ilmu Komputer Universitas Brawijaya.
5. Drs.Achmad Ridok, M.Kom selaku Dosen Pembimbing Akademik.
6. Seluruh civitas akademik Informatika/Ilmu Komputer yang telah banyak memberikan bantuan dan dukungan selama penulis menempuh pendidikan di PTIIK Universitas Brawijaya dan selam penyelesaian skripsi ini.
7. Teman-teman Sigura Hill E-10 Afrizal, Yhoga, Dwy, Restu, dan Dedi yang selalu memberi semangat dan canda tawa selama pengerjaan skripsi.

8. Teman-teman Bernard *and Fellas* Panser, Penusa, Andrias, dan Ivan yang selalu menemani bermain musik selama masa kuliah di Ilmu Komputer.
9. Semua teman-teman Ilmu Komputer 2010, khususnya teman-teman Ilmu Komputer A yang memberikan motivasi dan semangat selama masa kuliah di Ilmu Komputer Universitas Brawijaya.

Akhirnya penulis berharap agar skripsi ini dapat memberikan sumbangan dan manfaat bagi semua pihak yang berkepentingan.

Malang, 17 September 2014

Penulis

ABSTRAK

Fungsi Keanggotaan merupakan parameter yang penting pada penerapan logika *Fuzzy*, karena dari Fungsi Keanggotaan data dapat diketahui derajat keanggotaannya dan derajat keanggotaan memiliki pengaruh terhadap hasil akhir yang didapatkan. Pada perkembangan awal logika *Fuzzy*, pakar memiliki peran dalam menentukan batas fungsi keanggotaan. Namun pada perkembangannya sering tidak ditemukan pakar untuk menentukan batas fungsi keanggotaan. Sehingga berdasarkan permasalahan tersebut mulai dilakukan penelitian berkaitan dengan pembangkitan fungsi keanggotaan secara otomatis. Salah satu metode yang dapat digunakan untuk melakukan pembangkitan otomatis fungsi keanggotaan *fuzzy* menggunakan metode *Backpropagation Neural Network*. Proses pembangkitan batas fungsi keanggotan dengan menggunakan *Backpropagation Neural Network* dibagi dalam dua tahap yaitu tahap pelatihan dan pengujian, pada tahap pelatihan digunakan untuk memperbaiki nilai bobot. Sedangkan pada tahap pengujian dilakukan proses perhitungan nilai keluaran berdasarkan nilai bobot dari proses pelatihan. Data yang digunakan pada penelitian ini adalah data penderita penyakit hepatitis yang diperoleh dari *UCI Machine Learning Repository*. Proses pengujian yang dilakukan berdasarkan 3 tahapan yaitu pengujian *Error* Harap, *Learning Rate*, dan Uji Kombinasi Data Latih. Berdasarkan proses pengujian yang dilakukan didapatkan hasil fungsi keanggotaan optimal adalah pada nilai *Error* Harap 10^{-8}, *Learning Rate* 0.9, Jumlah data latih 64 data, jumlah data uji 16 data, serta MSE terbaik pada nilai 0.00000000000072651.

Kata Kunci: Logika Fuzzy, Fungsi Keanggotaan, Gaussian, *Backpropagation*, *Neural Network*

ABSTRACT

Membership function is an important parameter for implement fuzzy logic, because membership function determine degrees of membership and degree of membership affect on the final results. Fuzzy logic was earlier developed by experts who have capability in determining boundary of membership function, but it's often harder to find experts for determining boundary of membership functions. Based on these problems, research that was related on the automatically generating membership function was started. One of the methods that can be used to automatically generating membership function was Backpropagation Neural Network. Generating membership function by using Backpropagation Neural Network was divided in to two different step. First step was training phase that was used to obtain the best weights. While in the testing phase was used to calculate output value based on the weights from training phase. The data which was used in this research was hepatitis disease data. It was obtained from the UCI Machine Learning Repository. The testing process were carried out based on the 3 step, that were Target Error, Learning Rate, and Combination of Training Data. Based on the test performance showed the optimal membership function was 10^{-8} of Error Target value , 0.9 of Learning Rate, 64 amount of training data, 16 amount of testing data, and 0.00000000000072651 of the best MSE value.

Keywords *: Fuzzy Logic, Membership Function, Gaussian, Backpropagation, Neural Network*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCE CODE*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Logika Fuzzy merupakan logika yang diciptakan oleh Prof. Zadeh untuk mengatasi logika tegas yang dianggap kurang merepresentasikan kondisi yang ada di kehidupan. Pada logika *Fuzzy* memiliki karakteritik berupa derajat keanggotaan dengan nilai 0 hingga 1, sehingga pada nilai permasalahan tertentu dapat memiliki nilai derajat keanggotaan pada dua kondisi yang berbeda [ZAD-65]. Dalam logika *Fuzzy* terdapat proses Fuzzyfikasi, Fuzzyfikasi merupakan proses merubah bilangan Tegas ke dalam bilangan *Fuzzy* melalui suatu fungsi keanggotaan [ROS-04]. Sedangkan fungsi keanggotaan adalah suatu kurva yang menunjukan pemetaan elemen ke dalam rentang nilai 0-1 [ROS-04]. Fungsi keanggotaan memiliki peranan dalam penerapan logika *Fuzzy* karena akurasi output yang didapatkan bergantung bagaimana representasi yang dilakukan pada fungsi keanggotaan.

Penentuan batas fungsi keanggotaan pada permasalahan tertentu dapat dilakukan dengan melakukan konsultasi dengan pakar atau dengan melakukan pembangkitan fungsi keanggotaan secara otomatis. Pembangkitan fungsi keanggotaan dengan melakukan konsultasi dengan pakar memiliki kelebihan dalam hal tingkat validasi fungsi keanggotaan yang dibuat, namun pendekatan tersebut juga memiliki kekurangan yaitu jika dalam kasus tertentu tidak terdapat pakar dalam merepresentasikan fungsi keanggotaan. Berdasarkan pertimbangan beberapa aspek tersebut maka telah banyak dilakukan penelitian dengan berbagai metode untuk melakukan pembangkitan otomatis fungsi keanggotaan *Fuzzy*, salah satu contoh adalah dengan menggunakan metode Jaringan Syaraf Tiruan [YUN-05].

Jaringan syaraf tiruan merupakan pemodelan informasi berdasarkan system syaraf pada makhluk hidup [ROJ-96]. Pada Jaringan syaraf tiruan sendiri terdapat beberapa metode yang dibedakan berdasarkan pembelajaran yang dilakukan yaitu *Supervised Learning* dan *Unsupervised Learning*. Perbedaan dari kedua pembelajaran tersebut adalah pada keluaran yang dihasilkan dari

pembelajaran yang dilakukan. *Supervised Learning* digunakan ketika kita ingin mendapat hasil keluaran yang terukur dan jaringan yang terawasi. Sedangkan *Unsupervised Learning* digunakan untuk permasalahan yang hasil keluarannya belum diketahui [ROJ-96].

Pembelajaran *Supervised Learning* dan *Unsupervised Learning* sendiri telah digunakan dalam pembangkitan otomatis fungsi keanggotaan Fuzzy. Sebagai contoh pada penelitian sebelumnya telah digunakan pembelajaran *Supervised Learning* dengan menggunakan metode *Backpropagation* [YUN-12], dan pada pembelajaran *Unsupervised Learning* dengan Menggunakan metode *Self Organizing Feature Maps* (SOFM) [YAN-05]. Pada penelitian kali ini, peneliti akan melakukan pembangkitan otomatis fungsi keanggotaan *Fuzzy* pada data penderita hepatitis. Perbedaan penilitian ini dengan penelitian yang dilakukan oleh Yunizar adalah pada arsitektur jaringan yang digunakan dan contoh kasus yang digunakan. Pada penelitian Yunizar arsitektur yang digunakan berupa tiga *input layer*, tiga *hidden layer*, dan lima *output layer*, dimana metode Kohonen digunakan sebagai inputan pada *input layer* . Contoh kasus yang digunakan adalah pembangkitan fungsi keanggotaan untuk data umur, suhu, nilai, dan berat badan [YUN-05].

Pada penelitian ini arsitektur yang digunakan adalah berupa satu *node input layer*, dua *node hidden layar*, dan dua *node output layer*. Data yang digunakan untuk penelitian ini adalah data penderita hepatitis yang terdiri dari beberapa atribut. Penilitian ini dimaksudkan untuk mengetahui hasil dari pembangkitan otomatis fungsi keanggotaan fuzzy dengan menggunakan algoritma *Backpropagation Neural Network* untuk penerapan data yang memiliki atribut beragam.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, rumusan masalah dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana pembangkitan otomatis fungsi keanggotaan *Fuzzy* dengan menggunakan metode *Backpropagation Neural Network* pada pendetita penyakit Hepatitis?

2. Bagaimana hasil fungsi keanggotaan dari penerapan metode *Backpropagation Neural Network* dalam pembangkitan otomatis fungsi keanggotaan *Fuzzy* pada penderita Hepatitis?

1.3 Tujuan

Berdasarkan rumusan masalah diatas, tujuan yang ingin dicapai oleh penulis adalah sebagai berikut:

1. Untuk melakukan pembangkitan otomatis fungsi keanggotaan Fuzzy dengan menggunakan *Backpropagation Neural Network* .
2. Untuk mengetahui hasil fungsi keanggotaan dari penerapan algoritma *Backpropagation Neural Network* dalam pembangkitan otomatis fungsi keanggotaan *Fuzzy* pada penderita penyakit Hepatitis.

1.4 Manfaat

Adapun Manfaat yang diharapkan penulisan dari tugas akhir ini adalah sebagai berikut:

1. Memberikan informasi tentang metode yang dapat digunakan dalam pembangkitan otomatis fungsi keanggotaan *Fuzzy*.
2. Memberikan informasi tentang optimasi yang dihasilkan dari penerapan algoritma *Backpropagation Neural Network* dalam pembangkitan otomatis fungsi keanggotaan *Fuzzy*.
3. Memberikan informasi tentang penerapan lain dari algoritma *Backpropagation Neural Network.*

1.5 Batasan Masalah

Batasan masalah dalam tugas akhir adalah sebagai berikut:

1. Jenis dataset yang dipilih dalam penelitian ini adalah dataset penderita penyakit hepatitis. Dataset hepatitis tersebut didapatkan dari *UCI Machine Learning Repository* yang disumbangkan oleh Josef Stefan *Institute* pada tahun 1988.
2. Jenis Dataset yang digunakan dalam penelitian adalah dataset tentang kemampuan bertahan hidup penderita penyakit hepatitis.

3. Atribut Data yang yang digunakan dalam pembangkitan fungsi keanggotaan *fuzzy* adalah umur, bilirubin, alk phosphate, sgot, albumin, protime.

1.6 Sistematika Penulisan

Sistematika penulisan tugas akhir ini dibagi menjadi enam bagian, yaitu:

BAB I PENDAHULUAN

Bab ini berisikan latar belakang, rumusan masalah, tujuan dan manfaat, ruang lingkup, serta sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisikan dasar teori yang melandasi penulisan laporan tugas akhir ini.

BAB III METODOLOGI DAN PERANCANGAN

Bab ini berisikan analisis dan metode berupa penjabaran kebutuhan pemakai serta perancangan yang meliputi perancangan data, perancangan fungsi, dan perancangan antarmuka.

BAB IV IMPLEMENTASI

Bab ini berisikan proses pembuatan perangkat lunak dan hasil yang didapat pada tahap perancangan.

BAB V PENGUJIAN DAN ANALISIS

Bab ini berisi pengujian pada metode yang digunakan dan analisa pada perancangan proses pembuatan perangkat lunak

BAB VI KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang diambil berkaitan dengan sistem yang dibuat dan saran untuk pengembangan sistem lebih lanjut.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan tentang teori dasar yang berhubungan dengan Pembangkitan Otomatis Fungsi Keanggotaan Fuzzy dengan Menggunakan Metode *Backpropagation Neural Network*. Sedangkan Kajian pustaka membahas penelitian yang telah dilakukan sebelumnya dan penelitian yang diusulkan.

2.1 Kajian Pustaka

Fungsi Keanggotaan *Fuzzy* merupakan hal terpenting dalam penerapan logika fuzzy karena dalam fungsi keanggotaan fuzzy berfungsi untuk memetakan suatu nilai tegas kedalam suatu nilai dalam rentang 0 dan 1[JAN-97]. Sehingga dalam perkembangannya telah banyak dilakukan penelitian berkaitan dengan pembentukan fungsi keanggotaan *Fuzzy*, hal tersebut dilakukan untuk mendapatkan nilai optimasi terbaik dalam pembentukan fungsi keanggotaan.

Salah satu algoritma yang digunakan dalam pembangkitan fungsi keanggotaan *Fuzzy* adalah dengan menggunakan algoritma Jaringan Syaraf Tiruan [YUN-12]. Pada Algoritma Jaringan Syaraf Tiruan terdapat dua metode yang dapat digunakan dalam pembangkitan fungsi keanggotaan *Fuzzy*. Metode tersebut dibedakan berdasarkan metode pembelajaran yang dilakukan yaitu metode *Supervised Learning* (Metode Pembelajaran Terawasi) dan *Unsupervised Learning* (Metode Pembelajaran Tidak Terawasi).

Backpropagation Neural Network merupakan salah satu contoh dari metode *Supervised Learning* [ROJ-96]. Penelitian dengan menggunakan metode tersebut telah dilakukan oleh Yunizar pada tahun 2012 untuk melakukan pembangkitan otomatis fungsi keanggotaan *Fuzzy* pada data umur, suhu, nilai, dan berat badan [YUN-12]. Pada penelitian tersebut proses yang dilakukan dalam pengolahan data *input layer* adalah dengan menggunakan metode *Self Organizing Maps*, selanjutnya dilakukan penentuan bobot dan target yang nantinya akan digunakan pada saat pelatihan dan pengujian data.

Proses selanjutnya adalah melakukan pelatihan data dan pengujian data dengan menggunakan algoritma *Backpropagation Neural Network*. Data yang

digunakan dalam proses pelatihan adalah data umur dan suhu, sedangkan data yang digunakan sebagai data pengujian adalah data nilai siswa, dan data berat. Hasil dari penilitian tersebut adalah sebagai berikut :

- Metode *Backpropagation Neural Network* dapat digunakan sebagai metode dalam pembangkitan fungsi keanggotaan *Fuzzy* untuk data umur, suhu, nilai, berat badan.
- Metode Kohonen dapat digunakan untuk mendapatkan nilai variabel tambahan sebagai nilai input.
- Besarnya jumlah epoch tidak selalu menghasilkan target keluaran terbaik.
- Bobot hasil pelatihan dapat digunakan sebagai bobot awal untuk proses pelatihan.
- Maksimum epoch yang diberikan belum memberikan nilai learning rate yang diharapkan.

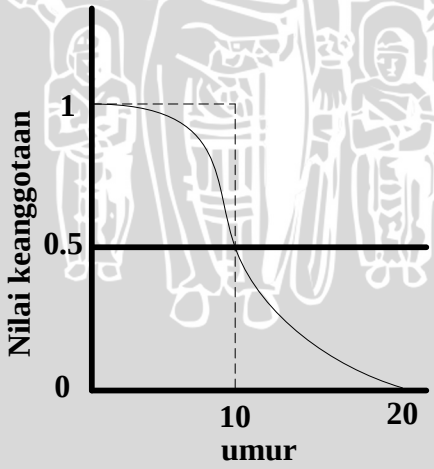
Penelitian dengan menggunakan metode *Unsupervised Learning* telah dilakukan oleh Chin Cun Yang dan N.K Bose pada tahun 2005 dengan menggunakan metode *Self Organizing Feature Maps* untuk data Iris [YAN-05]. Proses yang dilakukan dalam penelitian hanya menggunakan metode *Self Organizing Maps,* perbedaan utama antara metode *Supervised Learning* adalah terletak pada nilai keluarannya, dimana pada metode *Unsupervised Learning* tidak memerlukan target nilai keluaran. Hasil yang didapatkan dari penelitian ini adalah sebagai berikut :

- Metode *Self Organizing Feature Maps* dapat digunakan dalam pembangkitan fungsi keanggotaan *Fuzzy* untuk data Iris.
- Permasalahan data Multidimensional pada pembangkitan fungsi keanggotaan *Fuzzy* dapat diselesaikan dengan metode *Self Organizing Feature Maps.*
- Kelemahan Kecil yang didapatkan dari penerapan *Unsupervised Learning* adalah Nilai keluaran diskrit disebabkan oleh terbatasnya jumlah output layer pada SOFM.

2.2 Himpunan *Fuzzy*

Himpunan *Fuzzy* merupakan himpunan yang terikat dengan derajat keanggotaan [ZAD-65]. Himpunan *Fuzzy* dikembangkan oleh Profesor Zadeh dikarenakan pada beberapa kasus tertentu proses klasifikasi dengan himpunan Klasik dianggap masih belum bisa merepresentasikan secara tepat kondisi yang ada, sebagai contoh adalah ketika kita mengklasifikasikan bilangan real yang memiliki nilai lebih besar daripada 1, ataupun mengklasifikasikan tinggi seseorang [ZAD-65]. Menurut Prof.Zadeh, Himpunan *Fuzzy* dapat definisikan sebagai X merupakan suatu titik poin (Objek), dengan elemen umum dari X dinotasikan sebagai x. Sehingga $X = \{x\}$ [ZAD-65].

Pada Himpunan *Fuzzy* suatu himpunan tidak hanya memiliki batas nilai 0 dan 1 seperti yang digunakan pada himpunan tegas, sehingga himpunan dianggap dapat lebih merepresentasikan masalah yang dihadapi dalam kehidupan nyata [JAN-97]. Perbedaan utama antara Himpunan Tegas dengan Himpunan *Fuzzy* adalah batas nilai yang digunakan didalamnya, pada himpunan *Fuzzy* batas nilai yang digunakan memiliki nilai [0,1] [ZAD-65]. Penerapan himpunan *Tegas* dan himpunan *Fuzzy* dapat dilihat pada gambar 2.1:



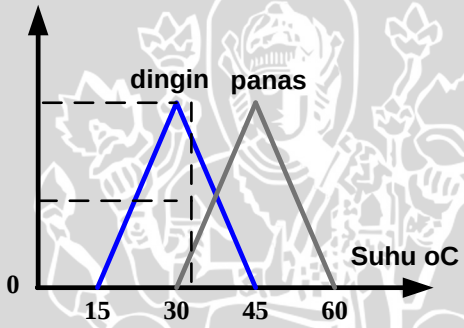
Gambar 2.1 Ilustrasi Himpunan Tegas dan Himpunan Fuzzy [YUN-12].

Berdasarkan Ilustrasi diatas, pada himpunan tegas umur 10 dan 20 hanya memiliki dua kemungkinan nilai yaitu nilai 0 dan 1. Sehingga pada kondisi umur 10 berdasarkan himpunan tegas akan dikategorikan dalam ke dalam umur muda atau kategori umur tua. Sedangkan pada Himpunan Fuzzy nilai dari umur 10 dan

20 ditentukan oleh derajat keanggotaannya, dan kurva seperti gambar 2.1 disebut fungsi keanggotan [YUN-12].

2.3 Fuzzyfikasi

Fuzzykasi merupakan proses merubah bilangan tegas kedalam suatu fungsi keanggotaan, dimana fungsi keanggotaan yang dapat digunakan meliputi fungsi keanggotaan Linear, fungsi keanggotaan Segitiga, fungsi keanggotaan Trapesium, fungsi keanggotaan Gaussian, fungsi keanggotaan Bel, dan fungsi keanggotaan Sigmoid [ROS-12]. Berikut ini merupakan contoh fuzzyfikasi dengan menggunakan fungsi keanggotaan segitiga, dimana diketahui nilai tegas dari suhu adalah 32^o C [YUN-12].



Gambar 2.2 Fungsi keanggotaan suhu [YUN-12].

Pada gambar 2.2 merupakan grafik fungsi keanggotaan suhu, berdasarkan grafik diatas maka nilai *fuzzy* dari suhu 32^o C dapat ditentukan dengan beberapa langkah berikut:

1. Dilakukan penentukan nilai lingustik dari suhu 32^o C, dimana variabel lingustik dari suhu 32^o C berada pada dingin dan panas.
2. Dilakukan perhitungan fungsi keanggotaan dingin, dengan menggunakan rumus :

$$\mu_{a1} = \frac{c-x}{c-b}, b \le x \le c \quad \text{.....} \quad (2.1)$$

3. Berdasarkan rumus diatas maka hasil fungsi keanggotaan dingin adalah :

$$\mu_{a1} = \frac{45-32}{45-30} = 0.867$$

4. Selanjutnya dilakukan perhitungan untuk fungsi keanggotaan panas dengan menggunakan rumus ;

$$\mu_{a2} = \frac{x-b}{b-a}, a \le x \le b \quad \text{..... (2.2)}$$

5. Berdasarkan rumus diatas maka hasil dari fungsi keanggotaan panas adalah

$$\mu_{a2} = \frac{32-30}{30-15} = 0.13$$

Maka nilai *fuzzy* yang didapatkan dari suhu 35^{o} C adalah 0.867 untuk fungsi keanggotaan dingin dan 0.133 untuk fungsi keanggotaan panas

2.4 Fungsi Keanggotaan

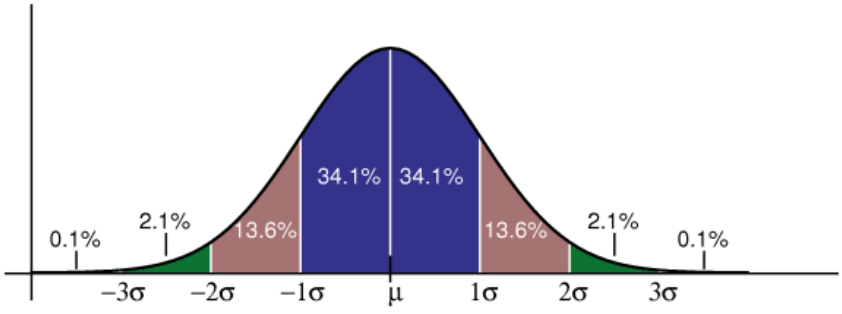
Fungsi Keanggotaan *(MF)* merupakan fungsi yang memetakan setiap elemen masukan kedalam derajat keangotaan antara 0 hingga 1. Pada logika *fuzzy* fungsi keanggotaan keanggotaan merupakan bagian yang sangat penting dalam, karena karakteris dari logika *fuzzy* ditunjukan oleh fungsi keanggotaan yang dimiliki [JAN-97].

Terdapat beberapa fungsi keanggotaan yang digunakan dalam logika *fuzzy*, antara lain fungsi keanggotaan Linear, fungsi keanggotaan Segitiga, fungsi keanggotaan Trapesium, fungsi keanggotaan Gaussian, fungsi keanggotaan Bel, dan fungsi keanggotaan Sigmoid [JAN-97]. Pada penilitan ini fungsi Keanggotaan yang digunakan adalah fungsi keanggotaan Gaussian. Penggunaan fungsi keanggotan Gaussian dimaksudkan untuk menghasilkan grafik fungsi keanggotaan yang lebih halus, dimana penjelasan fungsi keanggotaan gaussian adalah sebagai berikut :

1. Fungsi keanggotaan Gaussian

Fungsi keanggotaan gaussian didasari oleh 2 parameter $\{c, \sigma\}$, dimana c (pusat) merepresentasikan titik tengah dari fungsi keanggotaan sedangkan σ (standar deviasi) mendasari lebar dari fungsi keanggotaan [JAN-97]. Persamaan matematika dari fungsi keanggotaan gaussian adalah sebagai berikut :

$$gaussian\{x; c, \sigma\} = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2} \quad \text{..... (2.3)}$$



Gambar 2.3 Fungsi keanggotaan gaussian [SED-06].

2.5 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan merupakan salah satu algoritma dari kecerdasan buatan, di mana pada jaringan syaraf tiruan pemrosesan informasi yang dilakukan berdasarkan adaptasi dari jaringan syaraf biologi [FAU-94]. Jaringan syaraf tiruan memiliki sifat adapatif sehingga mampu mengubah struktur yang dimiliki untuk memecahkan permasalahan tertentu berdasarkan informasi internal ataupun ekternal yang melalui jaringan teresbut [YUN-12].

Terdapat dua metode pembelajaran yang biasa digunakan dalam dalam Jaringan Syaraf Tiruan, metode tersebut yaitu metode pembelajaran terawasi (*Supervised Learning)* dan metode pembelajaran tidak terawasi (*Unsupervised Learning)*[ROJ-96]. Perbedaan dari kedua metode pembelajaran tersebut adalah pada nilai keluaran yang didapatkan, metode *Supervised Learning* digunakan pada kasus dimana nilai keluaran telah ditentukan sebelumnya [ROJ-96]. Sedangkan metode *Unsupervised Learning* digunakan pada kasus dimana tidak memerlukan target nilai keluar. Pada penilitian ini metode pembelajaran yang digunakan adalah *Supervised Learning* dengan menggunakan metode *Backpropgation Neural Network* [ROJ-96].

Jaringan syaraf tiruan memiliki beberapa karekteristik antara lain arsitektur yang digunakan dalam jaringan syaraf tiruan disebut neuron, terdapat metode penentuan dan pembaharuan bobot (metode pembelajaran), dan pada jaringan syaraf tiruan terdapat fungsi aktivasi [FAU-94].

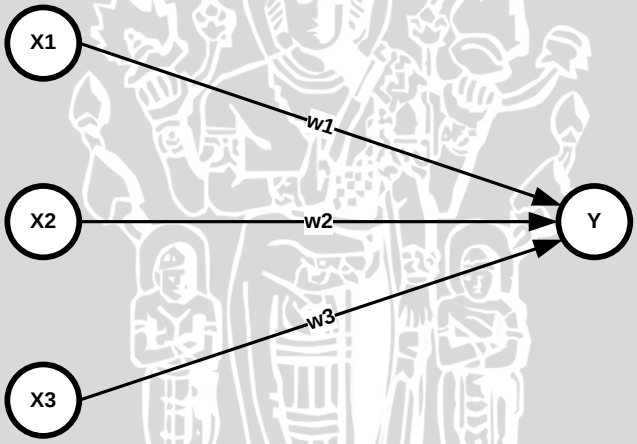
Pada jaringan syaraf terdiri dari *neurons, units, cells, nodes*. Setiap *neurons* terhubung dengan *neurons* lainnya secara langsung, dimana masing-masing diasosiasikan sebagai bobot [FAU-94]. Bobot tersebut mereprentasikan

infromasi yang akan digunakan untuk menyelesaikan permasalahan. Masing-masing *neurons* memiliki kondisi internal atau biasa disebut aktivasi dimana fungsi dari masukan diterima [FAU-94].

Secara umum *neurons* mengirimkan aktivasi tersebut sebagai suatu sinyal kepada *neurons* lainnya Dan terdapat hal yang perlu digaris bawahi bahwa, *neurons* hanya mampu mengirimkan sinyal dalam sekali waktu, walaupun sinyal tersebut telah dikirimkan pada beberapa *neurons* lainnya [FAU-94].

Sebagai contoh, terdapat *neurons Y* yang menerima masukan dari *neurons* X_1,X_2,X_3. Aktivasi dari *neurons* tersebut adalah x_1,x_2,x_3, sedangkan bobot dari X_1,X_2,X_3 menuju *Y* adalah w_1,w_2,w_3[FAU-94]. Nilai masukan jaringan atau *y_in* didapatkan berdasarkan persamaan matematika berikut :

$$y\_in = x_1w_1 + x_2w_2 + x_3w_3 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.4)$$



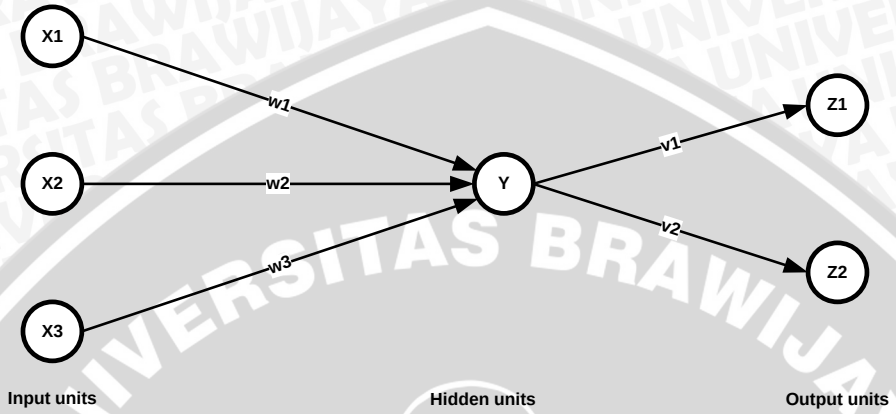
Gambar 2.4 Arsitektur Sederhana Neuron [FAU-94].

Pada gambar 2.4 merupakan arsitektur sederhana dari neuron. Aktivasi *y* dari *neuron* didapatkan berdasarkan fungsi dari nilai masukan dari jaringan, dimana persamaannya adalah sebagai berikut [FAU-94]:

$$y = f(y\_in) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.5)$$

Pada jaringan syaraf tiruan aktivasi dilakukan ketika sinyal akan dikeluarkan, dan salah satu hal yang yang mencirikan jaringan syaraf tiruan adalah terdapat proses penjumlahan sinyal input dan pengaktivasian sinyal output. Dalam pengaktivasian sinyal keluaran terdapat beberapa metode yang digunakan

salah satu contohnya adalah fungsi aktivasi sigmoid. Selanjutnya berdasarkan arsistektur pada gambar 2.4 dapat dikembangkan untuk membuat arsitektur sederhana pada jaringan syaraf tiruan. Diketahui bahwa *neuron Y* terhubung dengan *neuron* Z_1 dan Z_2 dengan bobot v_1 dan v_2.



Gambar 2.5 Arsitektur sederhana jaringan Syaraf Tiruan [FAU-94]

Pada gambar 2.5 merupakan contoh arsitektur sederhana dari jaringan syaraf tiruan yang terdiri dari tiga *input layer*, satu *hidden layer*, dan dua *output layer*. Berdasarkan arsitektur diatas nilai keluaran yang diterima oleh Z_1 dan Z_2 secara umum berbeda, hal tersebut disebabkan setiap sinyal membawa bobot yang berbeda [FAU-94].

2.6 Fungsi Aktivasi

Fungsi aktivasi merupakan salah satu ciri utama dari jaringan syaraf tiruan. Pada penerapannya fungsi aktivasi dapat digunakan secara *linear* pada arsitektur yang digunakan namun fungsi aktivasi juga dapat digunakan secara *nonlinear* pada permasalah tertentu, salah satu contohnya adalah pada arsitektur *multilayer*. Pada penelitian ini fungsi aktivasi yang digunakan adalah fungsi aktivasi sigmoid, dimana penjelasan dari fungsi aktivasi sigmoid adalah sebagai berikut:

1. Fungsi Aktivasi Sigmoid

   Persamaan matematika dari fungsi aktivasi *Sigmoid* adalah sebagai berikut[ROJ-96] :

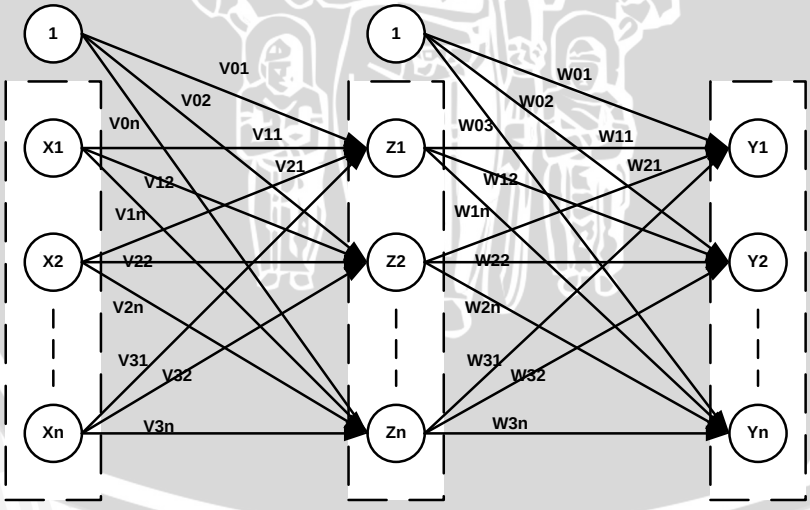
$$y = f(x) = \frac{1}{1+\exp^{-ax}}, 0 \le f(x) \le 1 \quad \text{.....} \quad (2.6)$$

Dimana a merupakan parameter bentuk dari fungsi sigmoid, besarnya nilai mempengaruhi bentuk grafik yang terbentuk dalam fungsi aktivasi sigmoid.

2.6 *Backpropagationa Neural Network(BPNN)*

Metode *Backpropagation Neural Network* merupakan metode *supervised learning* dimana salah ciri utama dari metode tersebut adalah meminimalkan kesalahan dari keluaran yang dihasilkan [ROJ-96]. Metode ini merupakan salah satu metode dari Jaringan Syaraf Tiruan yang dapat digunakan dalam pembangkitan otomatis fungsi keanggotaan Fuzzy [YUN-05].

Terdapat tiga tahapan dasar dari metode *Backpropagation Neural Network* antara lain perambatan maju berkaitan dengan pelatihan nilai masukan, perambatan mundur berkaitan dengan error, dan pengaturan bobot [FAU-94]. Secara arsitektur yang sering digunakan dalam jaringan syaraf tiruan adalah arsitektur multilayer [ROJ-96]. Gambar 2.6 merupakan contoh arsitektur dari metode *Backpropagation Neural Network.*



Gambar 2.6 Arsitektur Jaringan Syaraf Tiruan [FAU-94]

Arsitektur jaringan yang digunakan pada gambar 2.6 terdiri dari tiga *layer* yaitu *input layer* ($X_1,X_2,...,Xn$), *hidden layer* ($Z_1,Z_2,...,Zn$), dan *output layer* ($Y_1,Y_2,...,Yn$). Pada bagian *input layer* dan *hidden layer* terdapat node dengan nilai

satu, node tersebut merupakan bias dimana bias memiliki bobot nilai tersendiri [FAU-94].

Pada perambatan maju masing-masing *input layer* menerima sinyal selanjutnya sinyal dikirimkan pada masing *hidden layer* meliputi $Z_1, Z_2, \ldots, Zn$. Masing-masing *hidden layer* melakukan perhitungan aktivasi, selanjutnya mengirimkan sinyal kepada *output layer*. Pada *output layer*, masing-masing *output layer* melakukan perhitungan aktivasi, selanjutnya hasil dari *output layer* dibandingkan dengan target yang diberikan. Jika hasil yang didapatkan telah memenuhi target yang diberikan maka proses selesai, namun jika hasil yang didapatkan belum selesai maka dilakukan pengevaluasian bobot [FAU-94].

Seperti yang telah dijelaskan diatas, untuk melakukan pengevaluasian bobot adalah dengan menggunakan perambatan mundur. Bobot baru tersebut digunakan pada proses perambatan maju selanjutnya dan proses tersebut dilakukan hingga mendapatkan hasil keluaran sesuai dengan target yang telah diberikan [FAU-94].

Pada *Backpropagation Neural Network* proses yang dilakukan terdiri dari dua tahapan, tahapan yang pertama adalah tahap pelatihan untuk mendapatkan bobot sesuai dengan nilai target yang telah ditentukan dan tahapan yang kedua adalah proses implementasi bobot [FAU-94]. Untuk proses pelatihan dari *Backpropagation Neural Network* adalah sebagai berikut :

1. Inisialisasi Awal

   Pada tahapan ini dilakukan inisialisasi bobot, maksimum *epoch*, dan *Maximum Square Error* (MSE).
2. Lakukan langkah berikut jika nilai epoch kurang dari maksimum *epoch* dan nilai MSE lebih besar dari nilai *error* target:
   a. Lakukan langkah berikut ini pada setiap pasangan node di jaringan:

      Fase 1 : Perambatan Maju:

      i. Masing masing *node input* menerima sinyal masukan x_i , selanjutnya sinyal tersebut dikirimkan pada *layer* selanjutnya (*hidden layer).*

      ii. Masing-masing *node* pada *hidden layer* melakukan penjumlahan sinyal masukan terbobot berdasarkan persamaan matematika berikut:

$$Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i v_{ij} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.7)$$

Digunakan fungsi aktivasi untuk menghitung nilai keluaran :

$$Z_{outj} = f(Z_{inj}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.8)$$

Selanjutnya proses dilanjutkan pada masing-masing *node* pada *output units.*

iii. Masing-masing *node* pada *output layer* melakukan penjumlahan sinyal masukan terbobot berdasarkan persamaan matematika berikut :

$$Y_{ink} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.9)$$

Digunakan fungsi aktivasi untuk menghitung nilai keluaran :

$$Y_{outk} = f(Y_{ink}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.10)$$

Fase II : Perambatan mundur

iv. Dilakukan perhitungan nilai *error* berdasarkan nilai *error* dan nilai target, berdasarkan persamaan matematika berikut :

$$\delta_k = (t_k - y_k) f'(y_{ink}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.11)$$

Hitung nilai koreksi nilai bobot :

$$\Delta w_{jk} = \alpha.\delta_k z_j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.12)$$

Hitung nilai koreksi nilai bias :

$$\Delta w_{ok} = \alpha.\delta_k \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.13)$$

v. Masing-masing *node* pada *hidden layer* melakukan penjumlahan delta unit (node dari *output layer*) :

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.14)$$

Hitung nilai error dengan mengalikan nilai turunan aktivasi dengan delta input :

$$\delta_j = \delta_{inj} f'(z\_in_j) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.15)$$

Hitung koreksi nilai bobot :

$$\Delta v_{ij} = \alpha.\delta_j x_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.16)$$

Hitung koreksi nilai bias :

$$\Delta v_{oj} = \alpha.\delta_j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.17)$$

vi. Masing-masing *node output layer* melakukan perbaikan nilai bobot dan nilai bias :

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.18)$$

$$w_{ok}(baru) = w_{ok}(lama) + \Delta w_{ok} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.19)$$

Pada *hidden layer* masing-masing node juga dilakukan perbaikan nilai bobot dan nilai bias :

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.20)$$

$$v_{oj}(baru) = v_{oj}(lama) + \Delta v_{oj} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.21)$$

b. Dilakukan kondisi berhenti, jika nilai MSE dan Max_Epoch telah terpenuhi.

2.7 Normalisasi Data

Normalisasi Data merupakan suatu proses untuk melakukan penskalaan data sehingga suatu data berada dalam suatu rentang nilai tertentu. Proses normalisasi dilakukan untuk menghasilkan data yang baik, karena kualitas data yang digunakan mempengaruhi keluaran yang dihasilkan. Metode normalisasi data yang digunakan dalam penelitian ini adalah metode *Min-Max*. Normalisasi *Min Max* merupakan metode yang melakukan transformasi linear terhadap data asli, persamaan matematika dari metode ini adalah sebagai berikut [BUD-13] :

$$v' = \frac{v - \min A}{\max A - \min A} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.22)$$

Dimana :

v’ = data baru

min A = nilai minimum dari atribut.

max A = nilai maksimum dari atribut.

2.8 Denormalisasi

Denormalisasi adalah proses merubah bilangan hasil normalisasi ke dalam keluaran nilai data asli atau bukan dalam rentang nilai tertentu. Persamaan matematika dari denormalisasi adalah sebagai berikut [IND-12]:

$$x = Y * (Max\_x_p - Min\_x_p) \quad \text{(2.23)}$$

Dimana :

x = nilai data denormalisasi.

Y = Hasil Keluaran Pelatihan

Max_x_p = nilai maksimum sebelum dinormalisasi

Min_x_p = nilai minimum sebelum dinormalisasi

2.9 *Mean Square Error (MSE)*

Mean Square Error (MSE) merupakan salah satu metode yang dapat digunakan untuk melakukan evaluasi kesalahan. Pada jaringan syaraf tiruan MSE dapat digunakan untuk meningkatkan optimasi dari proses yang dilakukan. Persamaan Matematika dari MSE adalah sebagai berikut [AGU-10] :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad \text{(2.24)}$$

Dimana:

n = jumlah data.

y_i = nilai data sebenarnya.

$\hat{y}_i$ = nilai estimasi data.

2.10 Hepatitis

Hepatitis merupakan penyakit peradangan liver atau hati [WOR-13]. Hepatitis sendiri termasuk dalam kategori penyakit yang berbahaya karena hepatitis menyerang hati dimana hati merupakan salah satu organ penting dalam tubuh manusia yang memilik berbagai macam fungsi.

Dalam perkembangannya terdapat beberapa dataset dari hasil penelitian untuk mengetahui kemampuan bertahan seseorang dari penyakit hepatitis. Dimana parameter yang dapat dijadikan tolak ukur meliputi bilirubin, Alk Phosphate, SGOT, Albumin, Protime.

Dari beberapa parameter diatas, nilai yang didapatkan sesorang dari hasil tes kesehatan yang dilakukan dapat mempengaruhi kemampuan bertahan dari penyakit hepatitis yang diderita. Karena nilai yang ditunjukan dari paremeter

diatas dapat dijadikan sebagai suatu penanda apakah kondisi sesorang dapat dikatakan baik, kurang baik, ataupun buruk. Untuk penjelasan rentang nilai dari beberapa paremeter diatas adalah sebagai berikut :

- Bilirubin

  Bilirubin merupakan pigmen kuning yang ada dalam darah dan kotoran manusia [KRU-14]. Rentang nilai normal dari bilirubin berkisar antara 0.3-1.9 (ml/dl) [KRU-14]. Nilai Bilirubin yang melebihi batas normal merupakan salah satu indikasi bahwa Bilirubin dalam kondisi buruk.

- Alk Phosphate

  Alk Phosphate merupakan protein yang ditemukan pada seluruh jaringan tubuh [MED-14a]. Rentang nilai normal dari Alk Phosphate berkisar antara 44-147 IU/L [MED-14a]. Nilai Alk Phosphate yang melebihi batas normal merupakan salah satu indikasi bahwa Alk phosphate dalam kondisi buruk.

- SGOT

  SGOT merupakan enzim yang digunakan hati dalam menjalankan fungsinya [MED-14b]. Rentang nilainya adalah 5-40 unit/L [MED-14b].

  Nilai SGOT yang melebihi batas normal merupakan salah satu indikasi bahwa SGOT dalam kondisi buruk.

- Albumine

  Albumin merupakan protein yang dibuat oleh hati. Jumlah serum protein tersebut dapat diuji pada cairan bening darah [MED-14c]. Rentang nilai normal dari albumin berkisar antara 3.4-5.4 (g/dl)[MED-14c]. Nilai Albumine yang melebihi atau kurang dari batas normal merupakan salah satu indikasi bahwa Albumine dalam kondisi buruk.

- Protime

  Protime merupakan waktu yang dibutuhkan bagian cair plasma untuk membeku [MED-14d]. Rentang nilai normal dari protime berkisar antara 11 - 13.5 detik[MED-14d]. Nilai Protime yang melebihi batas normal merupakan salah satu indikasi bahwa Protime dalam kondisi buruk.

Kondisi kesehatan dari penderita hepatitis dapat diketahui berdasarkan masing-masing nilai dari parameter diatas. Jika kombinasi yang didapatkan lebih

banyak dalam ambang normal, maka penderita hepatitis tersebut memiliki kemungkinan bertahan hidup lebih baik. Namun jika nilai dari masing-masing parameter lebih banyak dalam kondisi di luar batas maka normal maka kemungkinan untuk bertahan hidup akan lebih kecil. Sebagai contoh terdapat pasien dengan nilai parameter sebagai berikut :

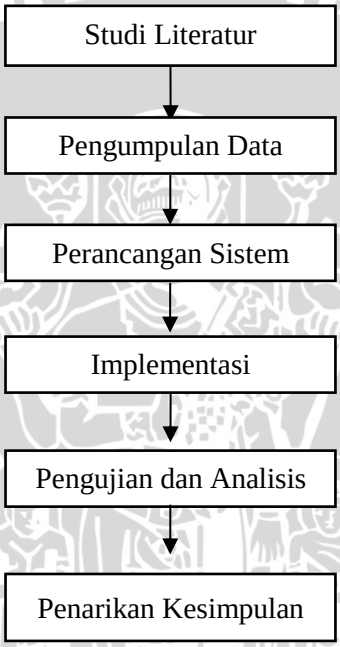
- Bilirubin : 2.3
- Alk Phosphate : 280
- SGOT : 98
- Albumin : 3.8
- Protime : 40

Berdasarkan paremeter data diatas, diketahui bahwa kondisi parameter yang masuk dalam rentang normal adalah Albumin. Dari kondisi pasien seperti itu maka kemampuan dalam bertahan hidup akan semakin kecil, dan hal tersebut dibuktikan dengan dataset dari *UCI Machine Learning* [UCI-88]. Pada dataset tersebut dijelakan bahwa dengan kondisi tersebut, pasien masuk dalam kondisi tidak mampu bertahan hidup. Dan diharapkan dengan adanya penelitian ini, mampu membantu meningkatkan optimasi dalam pembangkitan otomatis fungsi keanggotaan *Fuzzy* pada data penderita penyakit hepatitis. Sehingga kedepannya nilai akurasi yang didapatkan pada saat melakukan proses prognosis dapat lebih akurat.

BAB III
METODOLOGI DAN PERANCANGAN

3.1 Metodologi

Bab ini menjelaskan mengenai algoritma dan langkah-langkah yang akan digunakan dalam penelitian Implementasi *Backpropagation Neural Network* dalam pembangkitan otomatis fungsi keanggotaan *Fuzzy* pada penderita penyakit hepatitis. Gambar 3.1 menjelaskan mekanisme penelitian secara umum.



Gambar 3.1 Diagram Blog Metodologi

Penjelasan dari diagram blog diatas adalah sebagai berikut :

1. Studi literatur berkaitan dengan data hepatitis, Jaringan Syaraf Tiruan, *Backpropagation*, Logika *Fuzzy*.
2. Pengumpulan data hepatitis, dimana data hepatitis inilah yang nantinya akan digunakan dalam pembangkitan otomatis fungsi keanggotaan fuzzy.
3. Perancangan sistem untuk pembangkitan otomatis fungsi keanggotaan *Fuzzy* pada penderita penyakit hepatitis.

4. Implementasi sistem berdasarkan perancangan dan analisis yang telah dilakukan berkaitan dengan pembangkitan otomatis fungsi keanggotaan *Fuzzy* dengan menggunakan *Backpropagation Neural Network*
5. Pengujian dan analisis terhadap sistem yang telah diimplementasikan.
6. Penarikan Kesimpulan dari penilitian yang telah dilakukan.

3.2 Perancangan

3.2.1 Analisa Data

Pengumpulan data dilakukan melalui pencarian dataset hepatitis dari UCI *Machine Learning* [UCI-88]. Dataset tersebut dapat diunduh pada situs www.ics.uci.edu/~mlearn/. Dataset yang digunakan merupakan hasil donasi dari Josef Stefan *Institute* pada tahun 1988.

Penggunaan dataset dari UCI *Machine Learning* disebabkan karena pada penelitian yang berkaitan dengan hepatitis dataset yang digunakan secara umum menggunakan dataset hepatitis dari UCI *Machine Learning*, sehingga dapat disimpulkan bahwa dataset hepatitis UCI *Machine Learning* memiliki kualitas data yang baik. Tabel 3.1 merupakan contoh dataset hepatitis dari UCI *Machine Learning.*

Tabel 3.1 Dataset Hepatitis

Umur	***Bilirubin***	***Alk Phosphate***	***SGOT***	***Albumin***	***Protime***
34	0.9	95	28	4	75
39	1.3	78	30	4.4	85
32	1	59	249	3.7	54
41	0.9	81	60	3.9	52
30	2.2	57	144	4.9	78

Data tersebut merupakan data yang nantinya akan digunakan pada penelitian, penggunaan lima kriteria data tersebut disebabkan karena pada dataset hepatitis dari UCI *Machine Learning* hanya lima atribut diatas yang memiliki rentang nilai tertentu, sehingga hanya lima atribut yang memungkinkan untuk

dilakukan pembangkitan otomatis fungsi keanggotaan *Fuzzy*. Rentang nilai dari masing-masing dataset diatas adalah sebagai berikut :

- Umur : 10-80
- *Bilirubin* : 0.39-4.00
- *Alk Phosphate* : 33-250
- *SGOT* : 13-500
- *Albumin* : 2.1-6.0
- *Protime* : 10-90

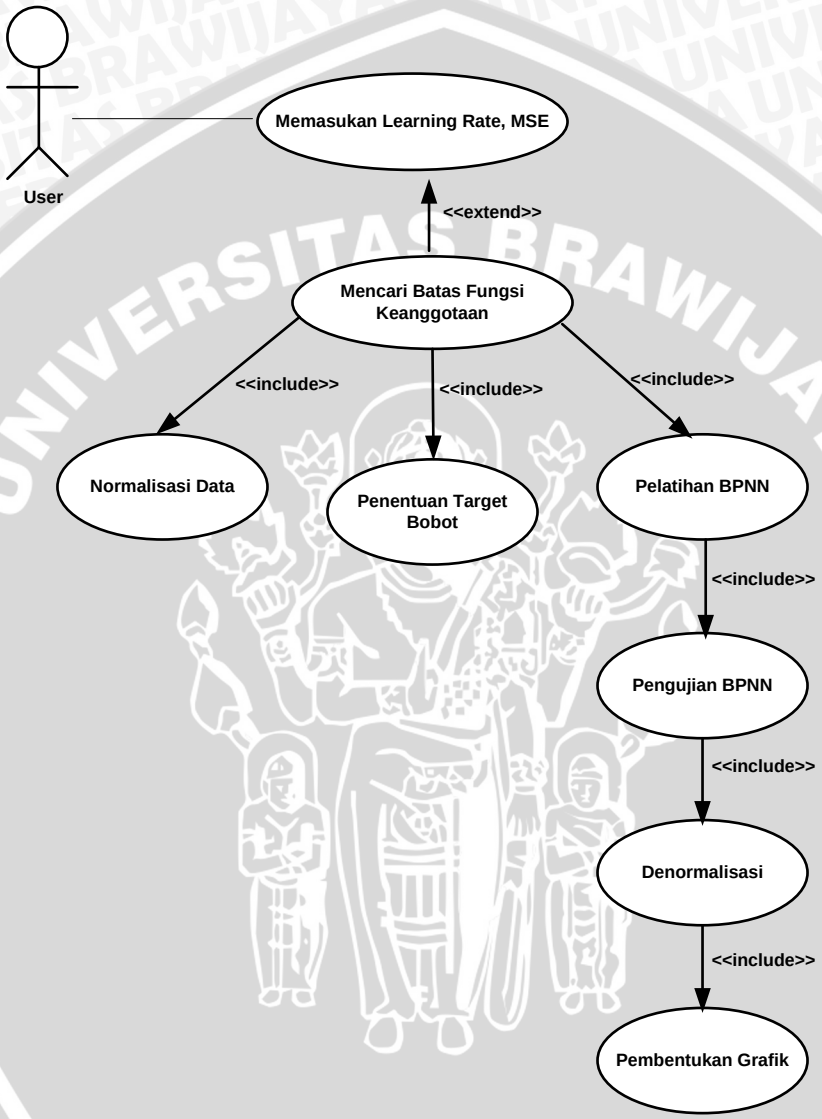
Sedangkan untuk variabel linguistik dari masing-masing variabel adalah sebagai berikut :

Tabel 3.2 Variabel Linguistik Data Hepatitis [PUT-14]

Parameter	**Linguistik**	**a**	**b**	**c**
Umur	Anak	0	8	15
Umur	Remaja	12	16	20
Umur	Dewasa	18	38	55
Umur	Tua	50	65	80
Bilirubin	Rendah	0	0,2	0,35
Bilirubin	Normal	0,3	0,9	1,5
Bilirubin	Tinggi	1,3	3	8
Alk Phosphate	Rendah	20	30	45
Alk Phosphate	Normal	44	100	147
Alk Phosphate	Tinggi	115	147	300
SGOT	Rendah	0	4	10
SGOT	Normal	8	25	40
SGOT	Tinggi	34	50	450
Albumin	Rendah	0	2	3,5
Albumin	Normal	3,4	4	5,5
Albumin	Tinggi	5,4	6	8
ProTime	Rendah	5	8	11
ProTime	Normal	10	12	14
ProTime	Tinggi	13,5	50	150

3.2.2 *Use Case* Sistem

Use Case Sistem digunakan untuk menjelaskan kumpulan proses pada sistem yang diawasi atau dijalankan oleh aktor. Gambar 3.2 merupakan *use case* sistem pada penelitian ini.

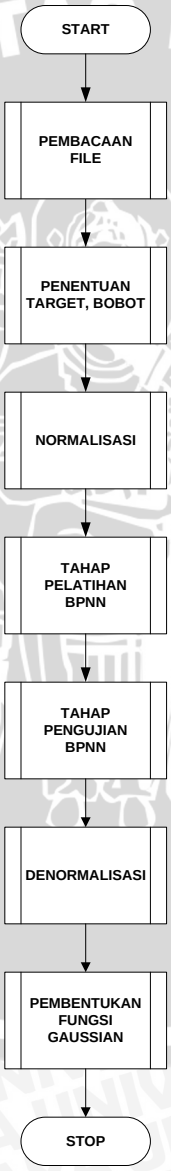


Gambar 3.2 *Use Case System*

Pada gambar 3.2 dijelaskan bahwa proses yang dilakukan oleh aktor adalah menetukan nilai *learning rate* dan nilai MSE. Setelah melakukan penentuan nilai tersebut sistem dapat melakukan proses normalisasi, penentuan bobot, hingga pelatihan dan pengujian menggunakan algoritma *Backpropagation Neural Network.*

3.2.3 Analisa Proses

Pada bagian analisa proses menjelaskan tentang rancangan proses atau alur kerja secara keseluruhan dari penelitian yang akan dilakukan. Sistem penyimpanan data yang digunakan pada penelitian adalah dalam file berekstensi .xls. Nantinya data tersebut diproses dengan menggunakan algoritma *Backpropagation Neural Network* untuk melakukan pembangkitan fungsi keanggotaan *Fuzzy* secara otomatis.



Gambar 3. 3 Diagram alir prosedur kerja Backpropagation Neural Network

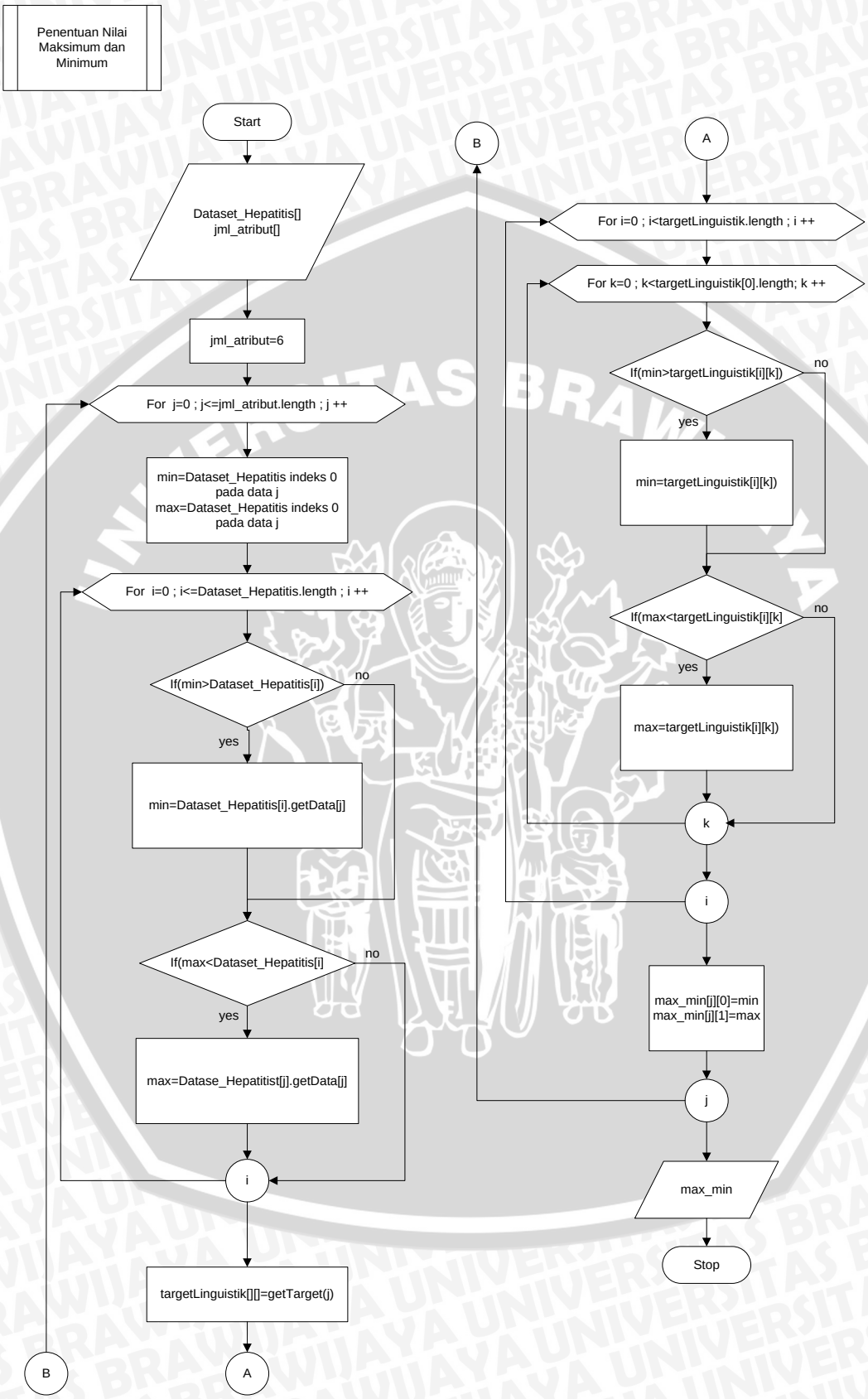
Gambar 3.3 menjelaskan mengenai proses yang dilakukan sistem dalam penelitian ini. Dimana proses awal yang dilakukan adalah melakukan pembacaan isi file, selanjutnya proses yang dilakukan adalah normalisasi data. Setelah dilakukan normalisasi data, proses yang dilakukan adalah melakukan penentuan nilai bobot dan target. Ketika proses penentuan nilai bobot dan target selesai dilakukan, maka dapat dilakukan penerapan algortima *Backpropagation Neural Network* untuk melakukan pelatihan dan pengujian, hingga nanti didapatkan hasil batas nilai fungsi keanggotaan.

Setelah didapatkan batas fungsi keanggotaan dilakukan proses denormalisasi untuk mengembalikan data pada rentang nilai yang sebenarnya. Proses terakhir yang dilakukan adalah melakukan pembentukan kurva fungsi keanggotaan menggunakan fungsi keanggotaan Gaussian, dimana nilai yang digunakan adalah nilai standar deviasi dan nilai titik tengah. Nilai titik tengah dan standar deviasi yang digunakan merupakan hasil proses *Backpropagation Neural Network* yang sebelumnya telah dilakukan.

3.2.3 Normalisasi Data

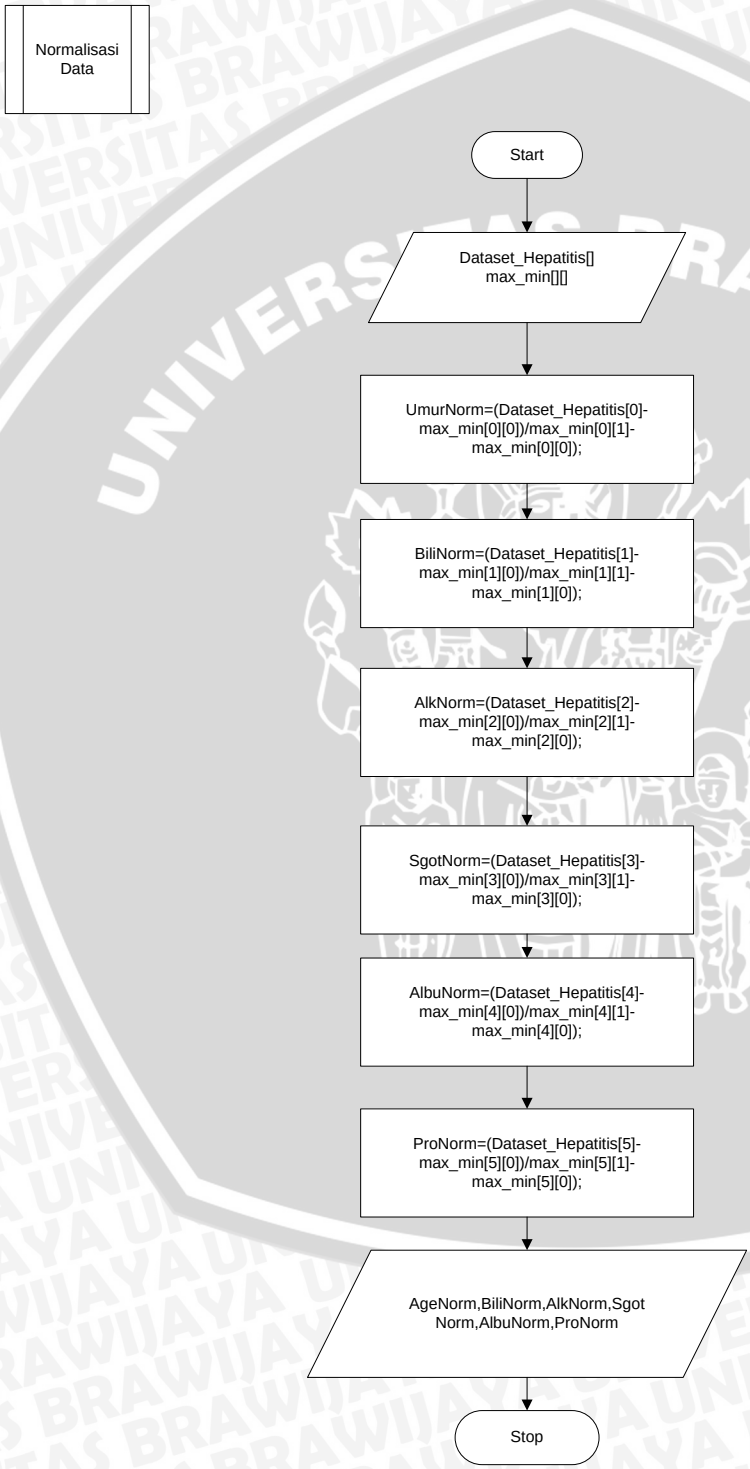
Proses yang dilakukan pada tahapan ini adalah melakukan penyeragaman data dalam rentang nilai nilai nol hingga satu. Penyeragaman data pada rentang nilai nol hingga satu karena fungsi aktivasi yang digunakan adalah fungsi aktivasi sigmoid sehingga data perlu dirubah pada rentang nilai nol hingga 1 untuk mendapatkan kualitas data yang lebih baik.

Proses awal yang dilakukan sebelum melakukan normalisasi data, adalah melakukan proses pencarian nilai terbesar dan terkecil. Pencarian nilai terbesar dan terkecil dilakukan untuk nantinya nilai tersebut digunakan pada pada proses normalisasi yang dilakukan. Proses yang dilakukan pada tahapan pencarian nilai maksimum dan minimum adalah melakukan perulangan sebanyak jumlah data. Setiap data tersebut nantinya dibandingkan, dan nilai terkecil dan terbesar yang didapatkan merupakan nilai maksimum dan minimum yang digunakan. Gambar 3.4 menjelaskan tentang proses kerja dari pencarian nilai maksimum dan minimum.



Gambar 3.4 *Flow Chart* Penentuan Nilai Maksimal dan Minimum

Fungsi normalisasi yang digunakan pada penelitian ini adalah normalisasi *Min-Max* , dimana gambaran umum dari proses normalisasi *Min-Max* adalah sebagai berikut.



Gambar 3.5 Flow Chart Normalisasi Data

Pada proses normalisasi diatas dijelaskan bahwa data diproses dengan menggunakan fungsi normalisasi *Min-Max*, dan proses tersebut diawali dengan mencari nilai terbesar dan terkecil dari data yang akan dilakukan proses normalisasi. Selanjutnya setelah didapatkan nilai *min max*, dilakukan perhitungan dengan menggunakan fungsi normalisasi *Min-Max* berdaskan persamaan 2.22.

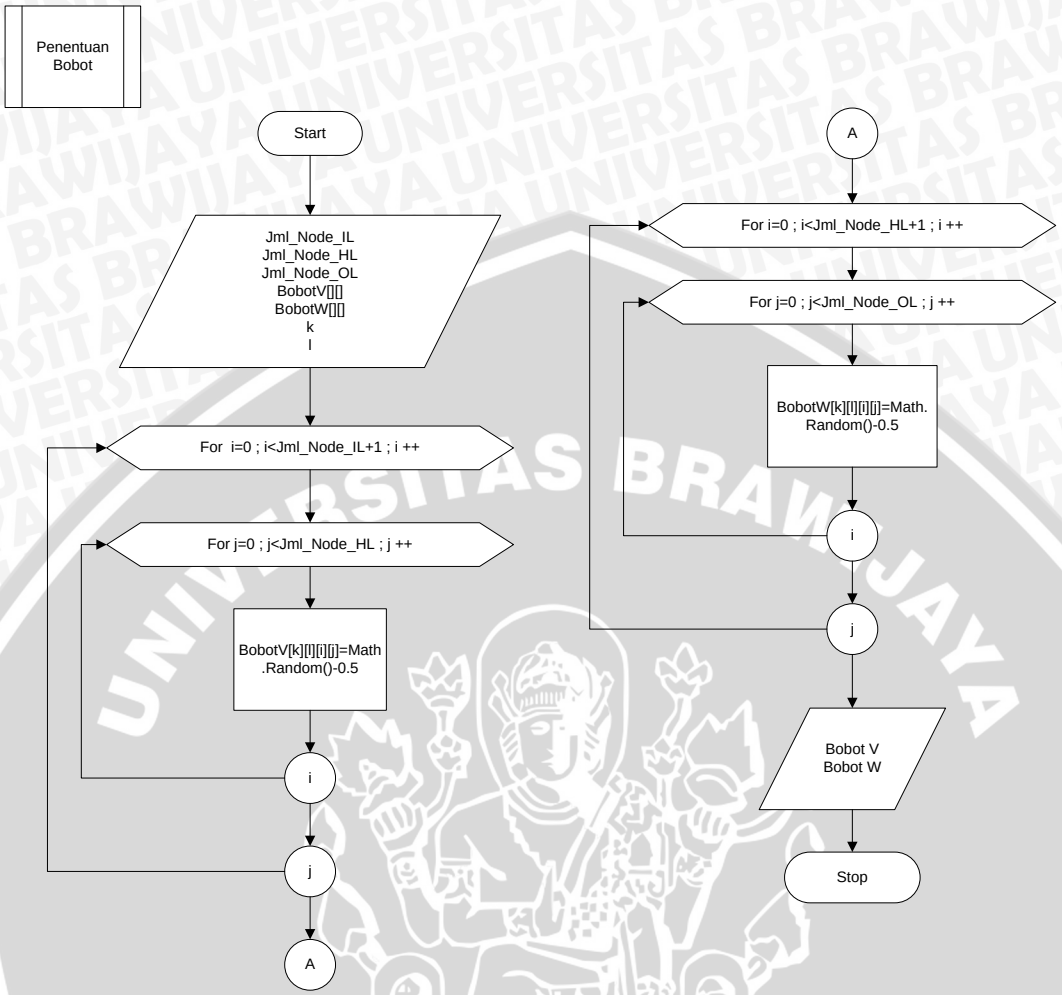
Berdasarkan persamaan tersebut, proses yang dilakukan adalah merubah nilai dalam rentang nilai tertentu berdasarkan parameter yang digunakan. Seperti dijelaskan pada *flowchart* diatas, parameter yang digunakan adalah nilai *min* dan *max* dari data yang akan dinormalisasi, nilai *max* baru dan nilai *min* baru. Penetapan nilai *max* baru dan *min* baru didasarkan pada rentang nilai yang akan digunakan pada data setelah proses normalisasi.

Pada penelitian ini rentang nilai yang diharapkan adalah nilai nol hingga satu, sehingga nilai *min* baru yang digunakan adalah nol dan nilai *max* baru yang digunakan adalah satu. Proses dengan menggunakan persamaan 2.22 tersebut dilakukan pengulangan hingga semua data telah dilakukan proses normalisasi.

3.2.4 Penentuan Nilai Target dan Bobot

Proses yang dilakukan pada tahapan ini adalah mendapatkan nilai target dan bobot. Pada algoritma *Backpropagation Neural Network* nilai target dan bobot memiliki peranan sangat penting terhadap hasil akhir yang didapatkan. Bobot digunakan untuk mendapatkan hasil akhir yang sesuai dengan target, sehingga dalam penerapan algortima *Backpropagation Neural Network* selalu dilakukan pembaharuan nilai bobot untuk mendapatkan bobot yang paling optimal untuk mendapatkan hasil keluaran sesuai dengan target yang ditentukan.

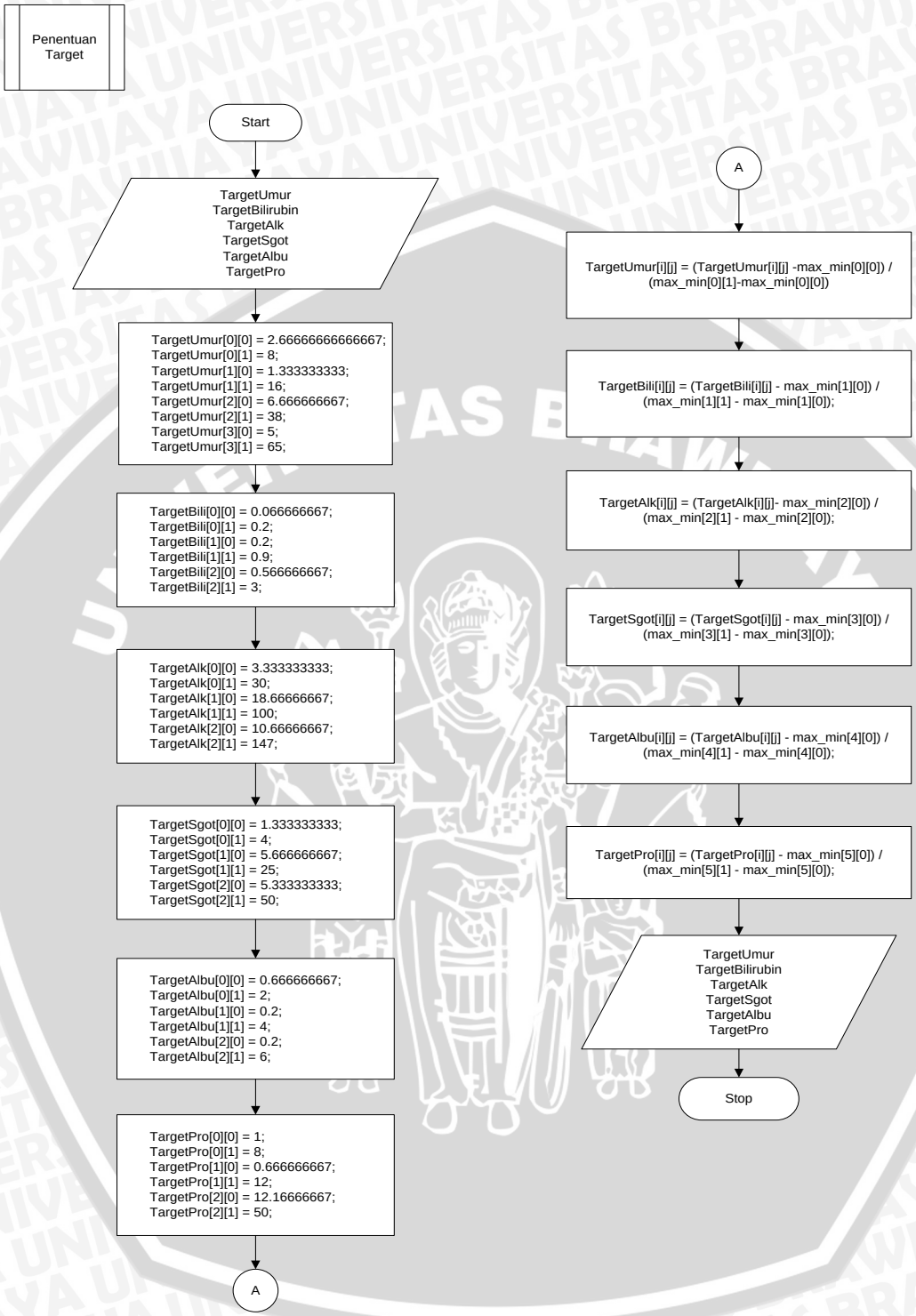
Nilai bobot pada penelitian ini menggunakan fungsi *random* nilai bobot, dengan ketentuaan nilai bobot antar -0.5 hingga 0.5. Proses yang dilakukan pada tahapan penentuan nilai bobot adalah melakukan *random* data pada semua bobot *v* dan *w*. Gambar 3.6 menjelaskan tentang proses *random* nilai bobot *v* dan bobot *w* pada penelitian ini.



Gambar 3.6 *Flow Chart* Penentuan Bobot

Proses yang dilakukan pada gambar 3.6 adalah melakukan perulangan sebanyak jumlah *hidden layer*, dengan ketentuan nilai *random* data adalah antara -0.5 hingga 0.5 untuk mencari bobot *v*. Selanjutnya dilakukan perluangan sebanyak jumlah *output layer*, dengan ketentuan nilai *random* data antara -0.5 hingga 0.5 untuk mencari bobot *w*.

Tahapan yang dilakukan selanjutnya setelah didapatkan nilai bobot *v* dan bobot *w* adalah menentukan nilai target. Target yang digunakan pada tahapan ini didapatkan berdasarkan acuan variabel linguistik yang telah ada, karena pada penelitian ini dimaksudkan untuk mengetahui optimasi yang dihasilkan dari penerapan metode *Backpropagation Neural Network.* Gambar 3.7 merupakan *flow chart* untuk proses penentuan target pada penelitian ini.



Gambar 3.7 Flow Chart Penentuan Target

Proses yang dilakukan pada gambar 3.7 adalah memasukan nilai target untuk semua atribut yang digunakan pada penelitian ini. Setelah didapatkan nilai atribut proses selanjutnya yang dilakukan adalah melakukan penghitung nilai

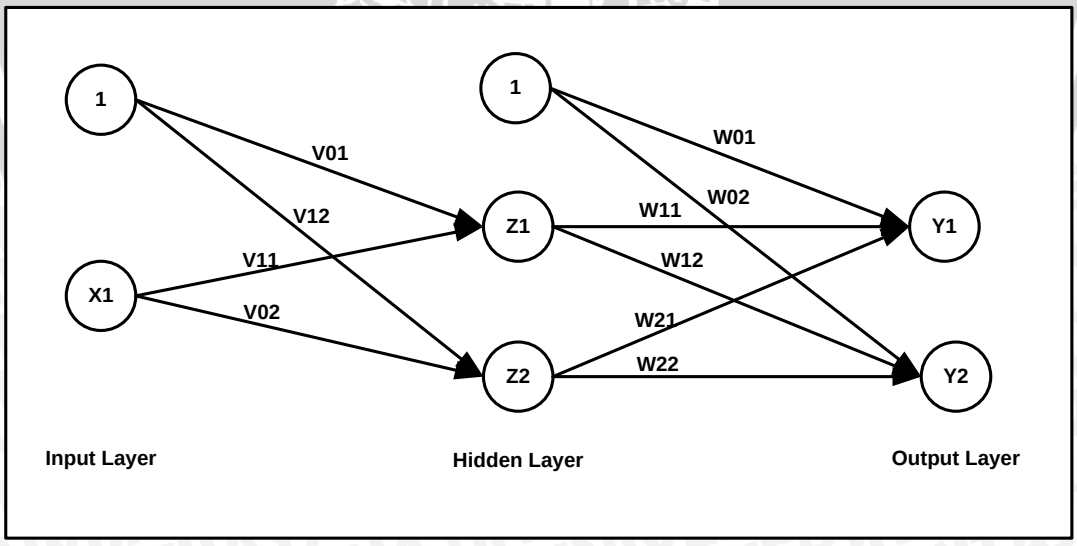
normalisasi untuk semua nilai target data. Penghitungan nilai normalisasi pada nilai target dilakukan untuk menyeragamkan nilai data pda rentang nilai nol hingga satu. Proses akan berhenti jika semua nilai target telah dilakukan proses normalisasi.

3.2.5 Penerapan Algoritma *Backpropagation Neural Network(BPNN)*

Backpropagation Neural Network merupakan metode pelatihan terawasi, salah satu ciri utama dari metode penelitian terawasi adalah ditentukannya nilai keluaran. Pada *Backpropagation Neural Network* proses yang dilakukan dibagi ke dalam dua tahapan yaitu proses pelatihan dan proses pengujian.

Pada proses pelatihan dilakukan proses *feedforward* atau perambatan maju dan *backpropagation* atau perambatan mundur. Sedangkan untuk proses pengujian proses yang dilakukan hanya proses *feedforward*, karena pada proses pengujian digunakan untuk mendapatkan nilai keluaran bukan untuk melakukan perbaikan nilai bobot.

Arsitektur yang digunakan dalam penelitian ini adalah bias, satu *node input layer*, dua *node hidden layer*, dan dua *node output layer*. Gambar 3.8 merupakan arsitektur yang digunakan dalam penelitian ini.



Gambar 3.8 Arsitektur *Backpropagation Neural Network*

Penggunaan satu *node* pada *input layer* didasarkan pada jumlah kriteria data penderita hepatitis yang digunakan pada penelitian ini. Pada penelitian ini, setiap atribut data dilakukan proses pembangkitan fungsi keanggotaan secara sendiri, karena jika semua atribut data dilakukan pembangkitan secara bersama maka data yang dihasilkan bukan menunjukan batas fungsi keangotaan pada atribut tertentu melainkan menunjukan batas fungsi keanggotaan pada gabungan atribut secara keseluruhan.

Pada arsitektur *Backpropagation Neural Network* antara *layer* satu dengan *layer* lain terhubung oleh bobot dan bobot yang digunakan pada penelitian ini didapatkan dengan menggunakan *random* data, sedangkan untuk nilai bobot yang menghubungkan antara *node hidden layer* dengan *node output layer* juga didapatkan dengan melakukan *random* data dengan rentang nilai nol hingga satu.

Penggunaan dua *node hidden layer* pada penelitian disebabkan oleh pengaruh jumlah *node input layer* yang digunakan karena dengan jumlah *node input layer* sebanyak satu akan lebih optimal jika memakai *hidden layer* sebanyak dua[JEF-08]. Sedangkan penggunaan dua *node output layer* pada penelitian ini menjelaskan tentang batasan nilai dari fungsi keanggotaan tersebut.

Fungsi keanggotaan *Fuzzy* pada penelitian ini ditunjukan oleh nilai target yang didapatkan dengan menggunakan fungsi keanggotaan Gaussian, penggunaan fungsi keanggotaan Gaussian dikarenakan fungsi keanggotaan Gaussian menghasilkan grafik yang lebih halus dibandingkan dengan penggunaan fungsi keanggotaan lain seperti fungsi keanggotaan segitiga maupun trapesium.

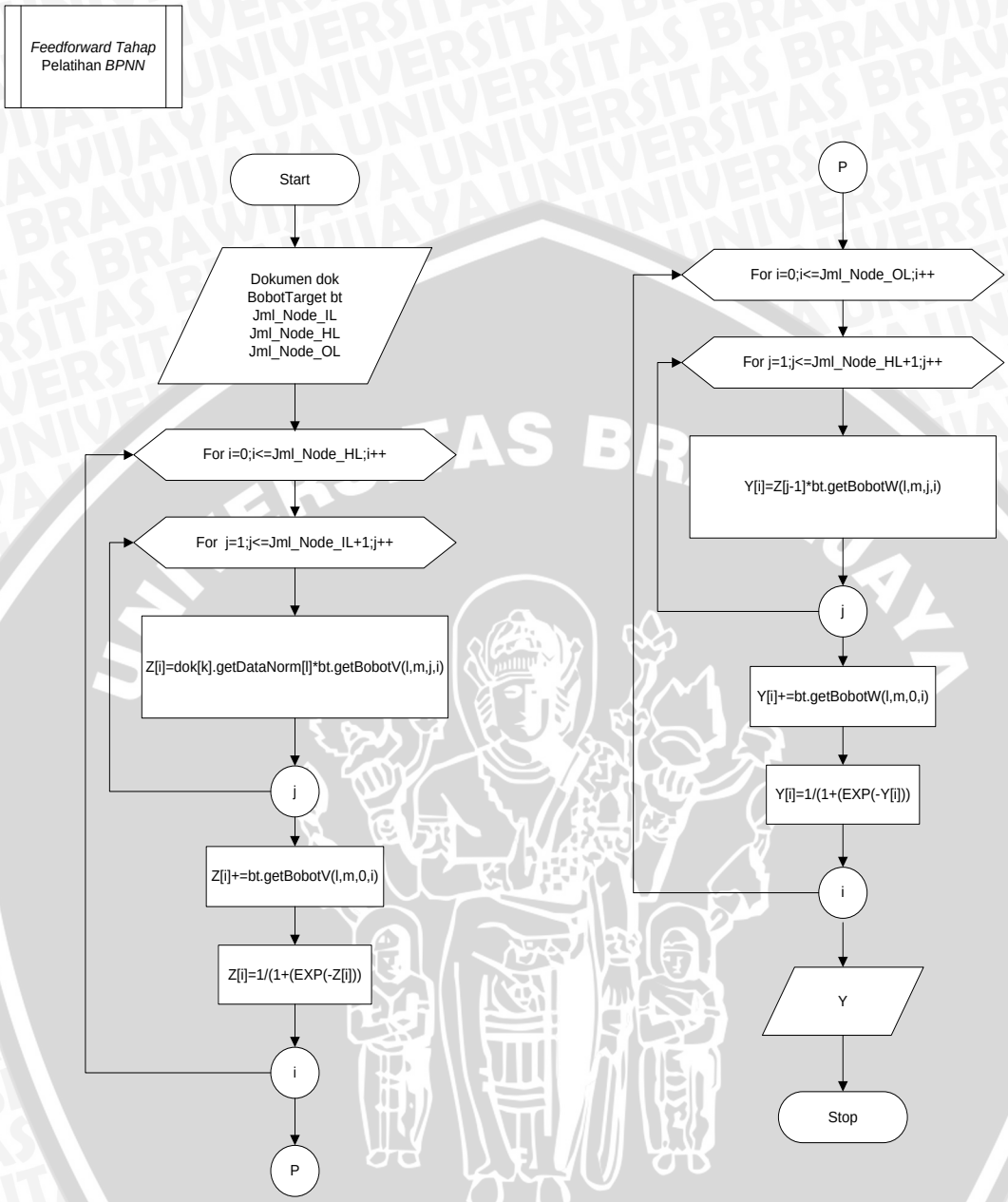
3.2.6 Pelatihan *Backpropagation Neural Network (BPNN)*

Secara umur proses yang dilakukan dalam tahap pelatihan adalah melakukan perbaikan nilai bobot. Proses yang dilakukan pada tahap ini terdiri dari proses *feedforward* dan *backpropagation.* Tahapan awal yang harus dilakukan adalah proses *feedforward*, setelah proses *feedforward* selesai maka proses *backpropagation* baru dapat dilakukan. Proses *feedforward* sendiri dilakukan untuk mendapatkan nilai keluaran dan proses *backpropagation* dilakukan untuk memperbaiki nilai bobot.

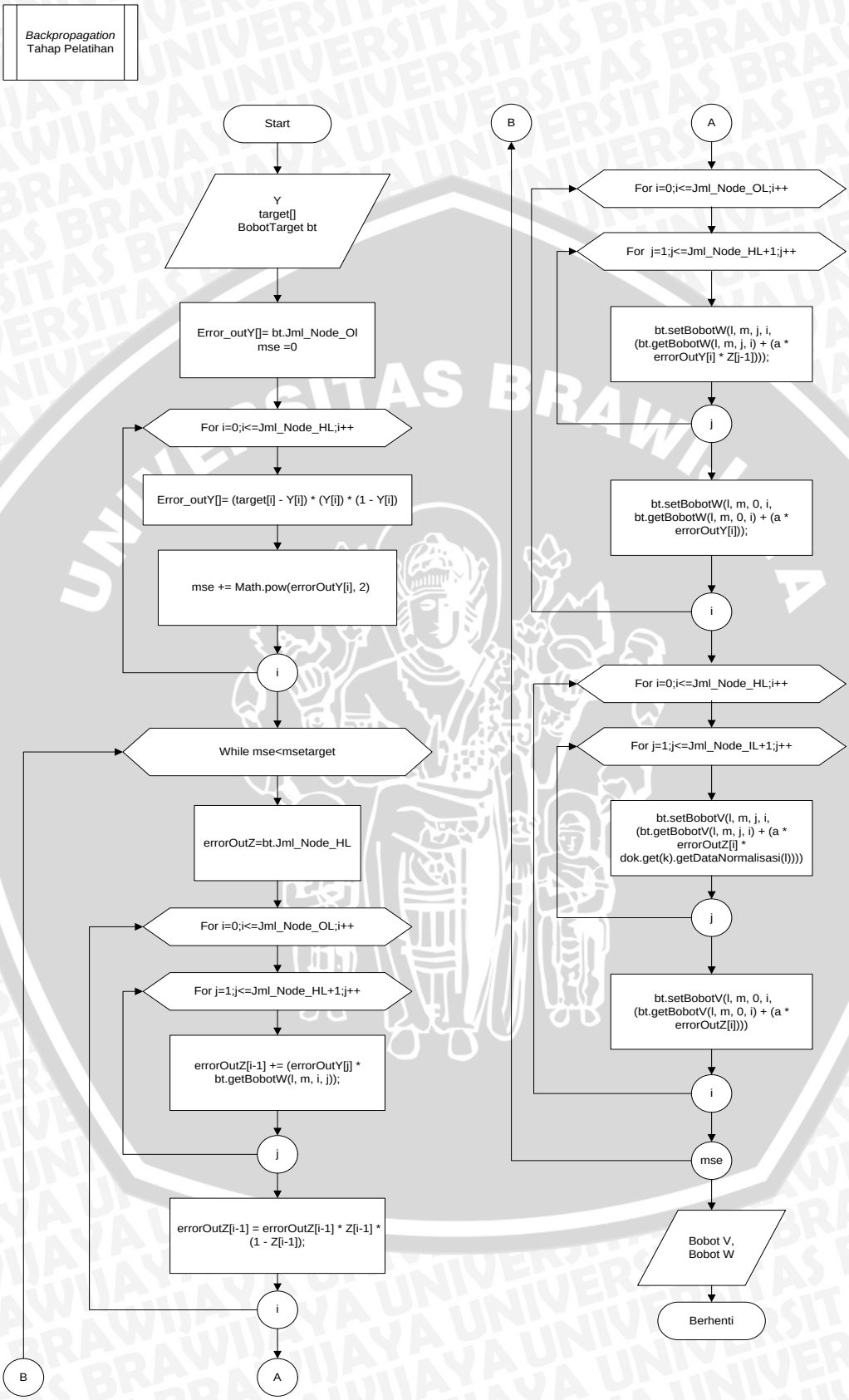
Proses awal dari *feedforward* diawali dengan pengiriman nilai dari setiap node pada input layer dan pada *hidden layer*. Proses yang dilakukan saat melakukan pengiriman adalah dengan melakukan perkalian nilai masing-masing node dengan bobot yang dimiliki ditambahkan dengan bobot dari bias, proses tersebut dihitung dengan menggunakan persamaan 2.7.

Setelah dilakukan perhitungan dengan menggunakan persamaan 2.7, selanjutnya dilakukan proses aktivasi dari *Z_in* dengan menggunakan persamaan 2.8 untuk mendapatkan nilai *Z_out*. Pada saat pengiriman sinyal ini, dilakukan perulangan hingga didapatkan nilai *Z_in* dan *Z_out* pada semua node di *hidden layer*.

Selanjutnya setelah didapatkan nilai pada kedua node di *hidden layer* maka proses selanjutnya yang dilakukan adalah melakukan perkalian node dengan bobot yang dimiliki ditambakan dengan bobot bias untuk mendapatkan nilai keluaran pada *output layer*. Proses yang dilakukan adalah dengan menggunakan persamaan 2.9 untuk mendapatkan *Y_in*, setelah didapatkan nilai *Y_in* maka dilakukan aktivasi dengan menggunakan persamaan 2.10 untuk mendapatkan nilai *Y_out* . Proses perulangan dilakukan hingga semua nilai *Y_in* dan *Y_out* pada node *output* layer telah berhasil didapatkan. Gambar 3.9 merupakan tahapan proses *feedforward* yang dilakukan pada penelitian ini.



Gambar 3.9 *Flow Chart feedforward* tahap Pelatihan



Gambar 3.10 *Flow Chart backpropagation* Tahap Pelatihan *BPNN*

Gambar 3.10 menjelaskan tentang proses yang dilakukan pada tahapan *backpropagation*, proses yang dilakukan adalah mendapatkan nilai error Y berdasarkan persamaan 2.11. Hasil error yang telah didapatkan selanjutnya dilakukan perhitungan nilai MSE berdasarkan persamaan 2.24. Proses selanjutnya yang dilakukan adalah melakukan pengecekan apakah apakah nilai MSE sudah memenuhi target atau tidak. Jika nilai *MSE* tidak memenuhi target proses selanjutnya yang dilakukan adalah proses perulangan berdasarkan banyaknya node pada output layer untuk mendapatkan nilai *error Z_k*, dimana pencarian nilai *error Z_k* dilakukan berdasarkan persamaan 2.14. Setelah didapatkan nilai *error Z_k* dilakukan proses aktivasi nilai *error Z_k* berdasarkan persamaan 2.15.

Selanjutnya dilakukan perulangan berdasarkan banyaknya data pada *node hidden layer* dan *node output layer* untuk menentukan $w_{jk}(baru)$ dan $w_{ok}(baru)$. Proses yang dilakukan pada tahapan ini adalah melakukan penjumlahan bobot lama ditambahkan dengan nilai Δw_{jk} berdasarkan persamaan 2.12. Sedangkan untuk proses perhitungan $w_{ok}(baru)$ dilakukan dengan melakukan penjumlahan bobot lama ditambahkan dengan Δw_{0k}. Proses perhitungan Δw_{0k} dilakukan berdasarkan persamaan 2.13.

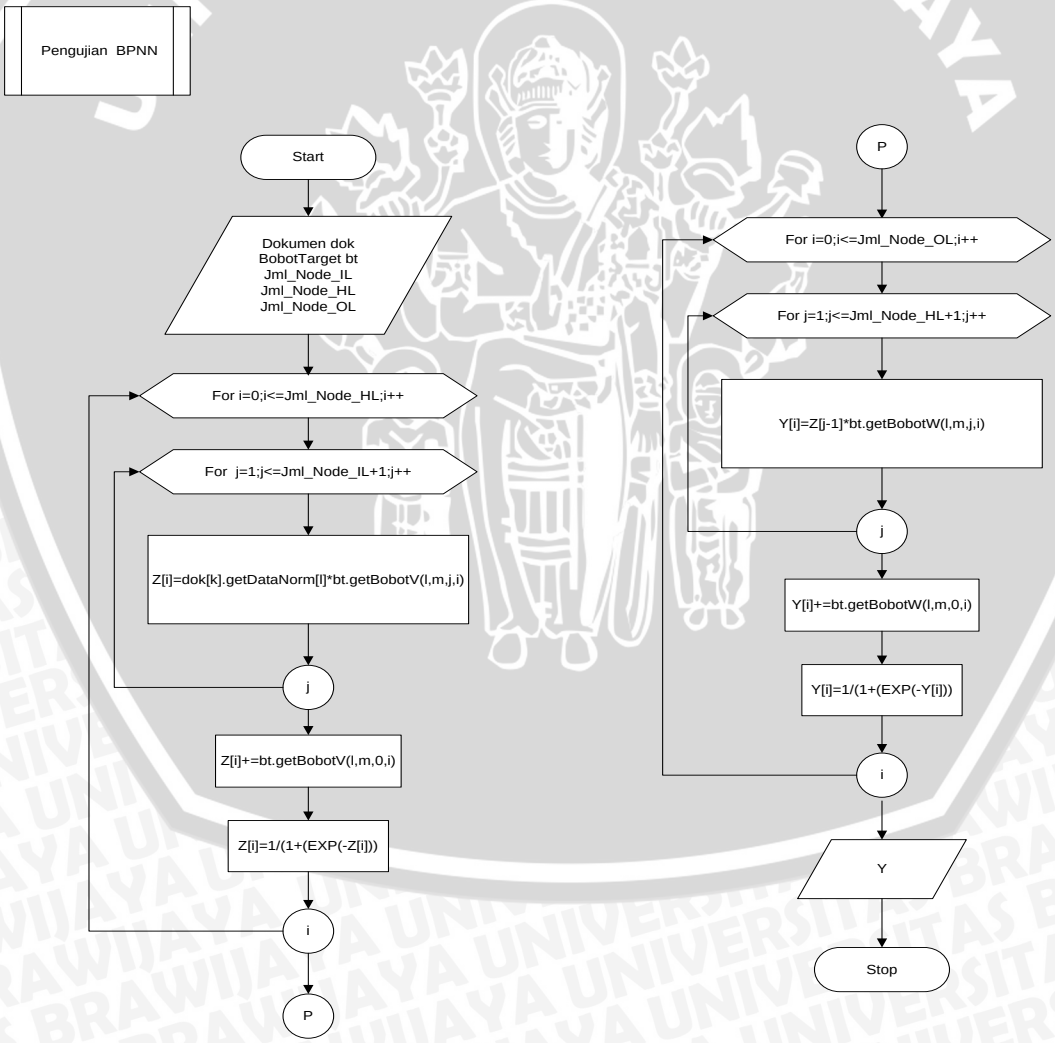
Proses selanjutnya yang dilakukan adalah perbaikan nilai bobot V, dimana pada tahap ini dilakukan perulangan berdasarkan banyaknya jumlah *node hidden layer* dan *node input layer*. Proses yang dilakukan adalah melakukan perhitungan $v_{ij}(baru)$ berdasarkan persamaan 2.20, sementara untuk mendapatkan nilai $v_{oj}(baru)$ adalah menggunakan persamaan 2.21.

Pada proses perhitungan $v_{ij}(baru)$ proses yang dilakukan adalah melakukan penjumlahan bobot lama dengan nilai Δv_{ij} dan Δv_{oj}. Sedangkan pada proses perhitungan $v_{oj}(baru)$ proses yang dilakukan adalah melakukan penjumlahan bobot lama dengan nilai Δv_{oj}. Perhitungan Δv_{ij} sendiri dilakukan berdasarkan persamaan 2.16, sedangakan untuk nilai dari Δv_{oj} proses perhitungan dilakukan berdasarkan persamaan 2.17.

3.2.7 Pengujian *Backpropagation Neural Network(BPNN)*

Proses yang dilakukan pada proses pengujian *Backpropagation Neural Network* secara umum sama dengan proses yang dilakukan pada pada tahap pelatihan yang membedakan hanyalah pada tahap pengujian proses yang dilakukan hanya mencakup proses *feedforward* untuk mendapatkan nilai keluaran dari proses yang dilakukan.

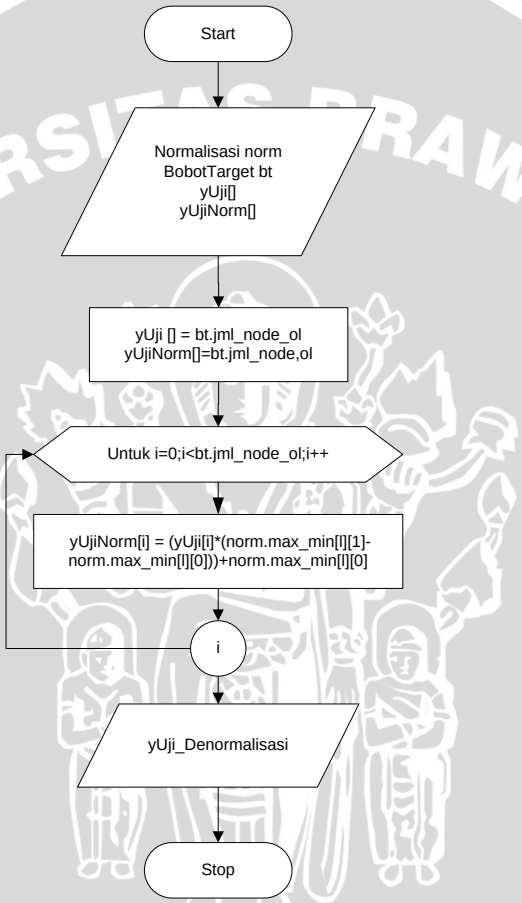
Secara umum proses *feedforward* pada tahap pelatihan dan tahan pengujian sama yang membedakan hanyalah pada bobot yang digunakan, karena pada *feedforward* tahap pelatihan bobot yang digunakan adalah bobot dari hasil peltihan sehingga bobot yang digunakan merupakan bobot yang telah diperbaiki. Untuk gambaran umum dari proses pengujian dijelaskan pada gambar 3.11.



Gambar 3.11 *Flow Chart backpropagation* Tahap Pelatihan *BPNN*

3.2.8 Denormalisasi

Proses yang dilakukan pada tahapan ini adalah merubah rentang nilai yang sebelumnya dalam rentang nilai 0-1 pada rentang nilai asli atau rentang nilai sebelum dilakukan proses normalisasi.

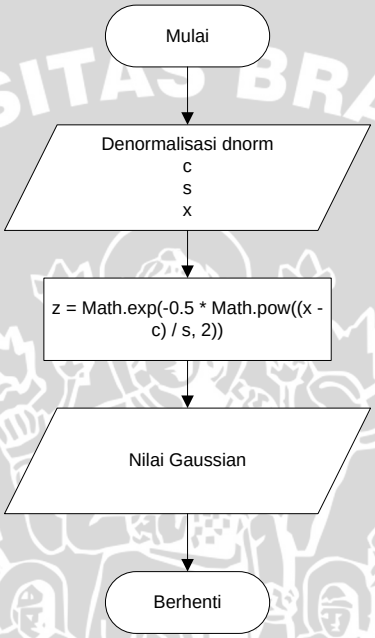


Gambar 3.12 *Flow Chart* Denormalisasi

Gambar 3.12 menjelaskan tentang proses yang dilakukan pada tahapan denormalisasi. Pada proses denormalisasi dilakukan perulangan sebanyak jumlah *node* pada *output layer*, proses selanjutya yang dilakukan adalah merubah nilai *y_out* kedalam rentang nilai awal. Proses pada tahapan denormalisasi akan berhenti jika semua *y_out* telah dilakukan denormalisasi.

3.2.9 Fungsi Gaussian

Proses yang dilakukan pada tahapan ini adalah melakukan perhitungan derajat keanggotaan suatu data, dimana parameter utama yang digunakan adalah standar deviasi dan nilai titik tengah yang sebelumnya didapatkan dari hasil *Backpropagation Neural Network.*

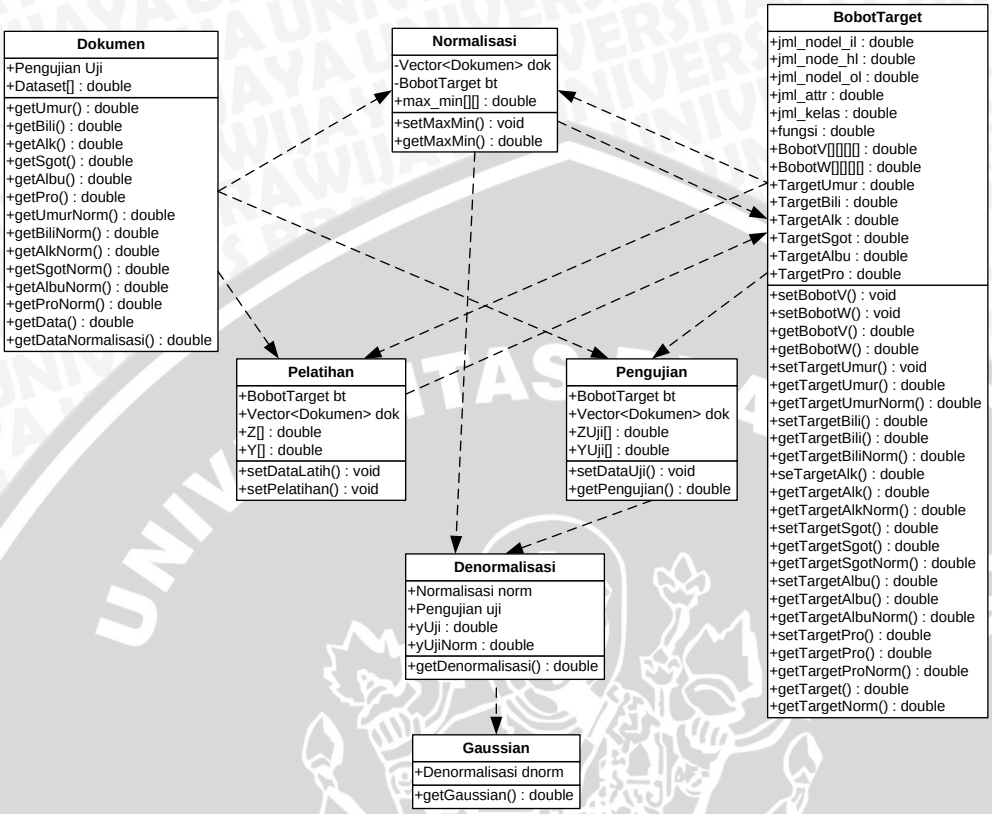


Gambar 3.13 *Flow Chart* Fungsi Gaussian

Pada Gambar 3.13 proses yang dilakukan pada tahapan ini adalah mendapatkan derajat keanggotan *x*, *x* sendiri merupakan data pada rentang nilai tertentu. Sedangkan *c* merupakan titik tengah, dan *s* merupakan standar deviasi.

3.3 *Class* Diagram

Class diagram digunakan untuk menunjukan proses yang terjadi dalam sistem, mencakup variabel yang digunakan, hingga method pada masing-masing *class* yang digunakan pada program. Gambar 3.14 menjelasakan tentang *Class* Diagram yang digunakan pada penelitian ini.



Gambar 3.14 *Class Diagram*

3.4 Perhitungan Manual

Atribut yang digunakan dalam penelitian umur, *bilirubin, alk phosphate, SGOT, Albumine, dan Protime.* Data dibawah ini merupakan data yang digunakan pada penelitian ini.

Tabel 3.3 Data Penelitian

Umur	Bilirubin	Alk Phosphate	SGOT	Albumin	Protime
34	0.9	95	28	4	75
39	1.3	78	30	4.4	85
32	1	59	249	3.7	54
41	0.9	81	60	3.9	52
30	2.2	57	144	4.9	78
38	2	72	89	2.9	46
40	0.6	62	166	4	63
38	0.7	53	42	4.1	85

Setelah data yang digunakan dalam penelitian ditentukan, proses yang dilakukan selanjutnya adalah melakukan proses normalisasi data, proses normalisasi data dilakukan untuk membuat data yang digunakan berada dalam suatu rentang tertentu. Proses perhitungan normalisasi dilakukan untuk semua data dari atribut yang digunakan, dan dalam proses perhitungannya dilakukan berdasarkan persamaan 2.25, contoh dari perhitungan normalisasi untuk data umur 34 adalah sebagai berikut :

$$v' = \frac{v - \min A}{\max A - \min A} * (new\_\max A - new\_\min A) + new\_\min A$$

$$v' = \frac{34 - 0}{72 - 1.333333333} * (1 - 0) + 0 = 0.462264$$

Tabel 3.4 merupakan hasil normalisasi untuk data lain yang digunakan pada penelitian ini.

Tabel 3.4 Data Normalisasi

Umur	Bilirubin	Alk Phosphate	SGOT	Albumin	Protime
0.462264	0.176056	0.3313253	0.063694268	0.655172	0.748322
0.533019	0.260563	0.2698795	0.068471338	0.724138	0.848993
0.433962	0.197183	0.2012048	0.59156051	0.603448	0.536913
0.561321	0.176056	0.2807229	0.140127389	0.637931	0.516779
0.40566	0.450704	0.1939759	0.340764331	0.810345	0.778523
0.518868	0.408451	0.2481928	0.209394904	0.465517	0.456376
0.54717	0.112676	0.2120482	0.393312102	0.655172	0.627517
0.518868	0.133803	0.1795181	0.097133758	0.672414	0.848993

Proses yang dilakukan setelah dilakukan proses normalisasi adalah proses penentuan nilai target dan bobot, dimana untuk nilai bobot *V* dan *W* dilakukan secara acak dengan rentang nilai.

Tabel 3.5 Bobot *V*

Data	Bobot v baru
v11	0.168298692
v12	0.534516478

Setelah didapatkan nilai dari bobot *V*, proses yang dilakukan selanjutnya adalah melakukan random data untuk mendapatkan bobot bias v. Berdasarkan persamaan 2.24 nilai bobot bias *V* merupakan nilai random dengan rentang nilai dari -0.5 hingga 0.5. Tabel 3.6 merupakan hasil dari nilai random untuk bobot bias v.

Tabel 3.6 Bobot Bias *V*

Bias	Bobot
v01	0.57
v02	0.14

Proses penentuan bobot w dan bobot bias w, dilakukan dengan melakukan pembangkitan secara acak dengan nilai nol hingga satu. Tabel 3.7 merupakan hasil dari pembangkitan secara acak nilai dari bobot w dan bobot bias w.

Tabel 3.7 Bobot w dan Bobot Bias w

Data	Bobot w
w01	0.043427405
w02	0.479195033
w11	0.73091685
w12	0.104579879
w21	0.780318349
w22	0.154138073

Nilai target yang digunakan pada penelitian ini didasarkan pada variabel linguistik yang telah ditentukan sebelumnya. Penggunaan variabel linguistik sebagai target yang digunakan dimaksudkan untuk mengetahui optimasi terbaik yang dapat dihasilkan dari penerapan *Backpropagation Neural Network* pada pembangkitan fungsi keanggotaan *fuzzy*. Tabel 3.8 merupakan nilai target yang digunakan pada data umur untuk fungsi keanggotaan anak.

Tabel 3.8 Nilai Target Data Umur Fungsi Keanggotaan Anak

Target	Nilai
a	0.018867925
b	0.094339623

Nilai target yang digunakan pada penelitian ini adalah nilai a dan b, dimana nilai a adalah standar deviasi sedangkan b adalah nilai titik tengah. Pada contoh perhitungan manual kali ini, data latih yang digunakan adalah data umur, dengan ketentuan data latih yang digunakan sebanyak 8 data latih. Tabel 3.9 merupakan tabel data latih yang digunakan.

Tabel 3.9 Data Latih

Umur
0.462264
0.533019
0.433962
0.561321
0.40566
0.518868
0.54717
0.518868

Proses yang dilakukan selanjutnya adalah melakukan perhitungan *Z_in* berdasarkan persamaan 2.7. Contoh perhitungan nilai Z_in untuk data 0.425 adalah sebagai berikut :

$$Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

$$Z_{inj} = 0.57 + (0.462264 * 0.168298692) = 0.647798$$

Selanjutnya dilakukan proses perhitungan nilai *Z_out* dengan menggunakan persamaan 2.8. Pada proses perhitungan nilai *Z_out* proses yang dilakukan adalah melakukan proses aktivasi pada nilai *Z_in*. Contoh perhitungan nilai Z_out untuk nilai *Z_in* 0.647798 adalah sebagai berikut :

$$Z_{outj} = f(Z_{inj})$$

$$Z_{outj} = 1/(1+\exp^{0.647798}) = 0.656514178$$

Tabel 3.10 merupakan hasil perhitungan *Z_in* dan *Z_out* untuk data latih pertama.

Tabel 3.10 Hasil *Z_in* dan *Z_out* Umur

Hidden Layer	*Z_in* Umur	*Z_out* Umur
1	0.647798452	0.656514178
2	0.387087806	0.595581452

Setelah didapatkan nilai *Z_in* dan *Z_out* proses yang dilakukan selanjutnya adalah mendapatkan nilai *Y_in* dan *Y_out*. Dimana proses perhitungan *Y_in* dilakukan dengan menggunakan persamaan 2.9 dan proses perhitungan *Y_out* dilakukan dengan menggunakan persamaan 2.10. Contoh perhitungan *Y_in* dan perhitungan *Y_out* pada *output layer* pertama adalah sebagai berikut :

$$Y_{ink} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$$

$$Y_{ink} = 0.043427405 + ((0.656514178 * 0.73091685) + (0.595581452 * 0.780318349)) = 0.988028$$

$$Y_{outk} = f(Y_{ink}) = 1/(1+\exp^{0.988028}) = 0.728698203$$

Tabel 3.11 merupakan hasil perhitungan *Y_in* dan *Y_out* untuk data latih pertama.

Tabel 3.11 Nilai *Y_in* dan *Y_out*

Output layer	*Y_in*	*Y_out*
1	0.988027815	0.728698203
2	0.639654984	0.654675465

Hasil *Y_out* yang telah didapatkan selanjutnya digunakan sebagai acuan untuk mendapatkan nilai error atau δ_k. Pada saat perhitungan nilai error pada node yang menghubungkan *hidden layer* dengan *output layer*, persamaan yang digunakan adalah persamaan 2.11. Contoh perhitungan nilai *error* untuk *output layer* pertama adalah sebagai berikut :

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

$$\delta_k = (0.018867924528301 - 0.728698203) * (0.728698203) * (1 - 0.728698203)$$

$$\delta_k = -0.140331401$$

Tabel 3.12 merupakan hasil perhitungan niai δ_k untuk data latih pertama.

Tabel 3.12 Nilai *error* δ_k

Output layer	Target	Error/ Sk
1	0.018867925	-0.14033141
2	0.094339623	-0.126678206

Setelah didapatkan nilai δ_k, proses yang dilakukan selanjutnya adalah mendapatkan nilai δ_{in} dengan menggunakan persamaan 2.14. Contoh perhitungan δ_{in} untuk *hidden layer* pertama adalah sebagai berikut :

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

$$\delta_{inj} = (-0.140331411 * 0.73091685) + (-0.126678206 * 0.104579879)$$

$$\delta_{inj} = -0.115818584$$

Perhitungan nilai δ_{in} digunakan untuk mengtahui nilai error yang didapatkan pada node yang menghubungkan antara *input layer* dengan *hidden layer*. Setelah didapatkan nilai δ_{in} proses yang dilakukan selanjutny adalah melakukan perhitungan error δ_j dengan menggunakan persamaan 2.15. Contoh perhitungan δ_j *hidden layer* pertama adalah sebagai berikut :

$$\delta_j = \delta_{inj} f'(z\_in_j)$$

$$\delta_j = (-0.115818584) * (0.656514178) * (1 - 0.656514178)$$

$$\delta_j = -0.026117474$$

Tabel 3.13 merupakan hasil perhitungan nilai δ_{in} dan δ_j untuk data latih pertama.

Tabel 3.13 Nilai δ_{in} dan Nilai δ_j

hidden layer	δ_in	nilai error
1	-0.115818584	-0.026117474
2	-0.129029109	-0.031078491

Proses yang dilakukan selanjutnya adalah melakukan perbaikan nilai bobot pada bobot yang menghubungkan *hidden layer* dengan *output layer*. Pada tahapan ini proses yang dilakukan adalah berdasarkan persamaan 2.18 untuk mendapatkan melakukan bobot baru dan persamaan 2.19 untuk mendapatkan bobot bias baru.

Perbedaan persamaan 2.18 dan 2.19 adalah pada proses perhitungan Δw, dimana pada perhitungan bobot bias proses yang dilakukan adalah melakukan perkalian antara *learning rate* dengan nilai δ_k. Sedangkan pada perhitungan Δw bobot adalah melakukan perkalian antara *learning rate*, δ_k, dan *Z_out*. Contoh perhitungan untuk pembaharuan bobot *w11* dan bias *w01* adalah sebagai berikut :

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$$

$$w_{jk}(baru) = 0.73091685 + (0.1 * -0.140331411 * 0.656514178)$$

$$w_{jk}(baru) = 0.721704$$

$$w_{ok}(baru) = w_{ok}(lama) + \Delta w_{ok}$$

$$w_{ok}(baru) = 0.043427405 + (0.1 * -0.14033141)$$

$$w_{ok}(baru) = 0.029394264$$

Tabel 3.14 merupakan hasil pembaharuan bobot dan bobot bias pada node yang menghubungkan *hidden layer* dengan *output layer*

Tabel 3.14 Bobot Baru W

Hidden/Output	Bobot
w11	0.721703894
w12	0.096263275
w21	0.77196047
w22	0.146593354
bias *w01*	0.029394264
bias *w02*	0.466527212

Pada node yang menghubungkan antar *input layer* dengan *hidden layer* proses yang dilakukan adalah dengan menggunakan persamaan 2.20 dan persamaan 2.21 untuk melakukan perhitungan nilai bobot dan bobot bias *V*. Pada persamaan 2.20 dan 2.21 proses perhitungan yang dilakukan adalah melakukan penjumlahan bobot lama dengan Δv. Perhitungan Δv pada bobot dilakukan dengan menggunakan persamaan 2.16, dan hitung koreksi pada bias dengan menggunakan persamaan 2.17. Contoh perhitungan untuk pembaharuan bobot baru *v11* dan bias *v01* adalah sebagai berikut :

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$$

$$v_{ij}(baru) = 0.16829869\,2 + (0.1 * -0.026117474 * 0.462264)$$

$$v_{ij}(baru) = 0.167091375$$

$$v_{oj}(baru) = v_{oj}(lama) + \Delta v_{oj}$$

$$v_{oj}(baru) = 0.57 + (0.1 * -0.02611747 4)$$

$$v_{oj}(baru) = 0.567388253$$

Tabel 3.15 merupakan hasil perbaikan bobot v dan bobot bias v.

Tabel 3.15 Bobot Baru V

Input/hidden	bobot
v11	0.167091375
v12	0.533079831
bias *v01*	0.567388253
bias *v02*	0.136892151

Proses yang dilakukan selanjutnya adalah melakukan perhitungan *MSE* untuk mengetahui *error* rata-rata yang didapatkan. Persamaan yang digunakan dalam perhitungan *MSE* adalah dengan menggunakan persamaan 2.24. Contoh perhitungan *MSE* pada data latih pertama adalah sebagai berikut :

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$MSE \frac{(-0.14033)^2 + (-0.12668)^2}{2}$$

$$MSE = 0.01787$$

Proses perhitungan pada tahap pelatihan dilakukan untuk semua data latih yang digunakan, dan proses tersebut berhenti jika *MSE* telah memenuhi kondisi berhenti. Pada tahap pelatihan hasil keluaran yang nantinya digunakan pada tahap pengujian adalah bobot yang telah diperbaharui. Tabel dibawah merupakan bobot akhir hasil pembelajaran yang dilakukan.

Tabel 3.16 Bobot akhir W

Hidden/output	bobot
w11	0.656429282
w12	0.038868439

w21	0.712470824
w22	0.094285198
bias w*01*	-0.070231508
bias w*02*	0.378930092

Tabel 3.17 Bobot akhir V

Input/hidden	bobot
v11	0.158540715
v12	0.522786892
bias *v01*	0.550359016
bias *v02*	0.116390258

Proses yang dilakukan selanjutnya proses pengujian, tahapan yang dilakukan pada tahapan pengujian sama dengan tahapan *feedforward* yang dilakukan pada tahap pelatihan. Proses yang membedakan hanyalah pada bobot yang digunakan, pada tahap pengujian proses perhitungan *Z_in* dan *Y_in* dilakukan dengan menggunakan bobot akhir hasil pembelajaran untuk semua data latih yang digunakan. Tabel 3.18 merupakan hasil perhitungan *Y_in*, *Y_out*, *Z_in*, dan *Z_out* pada semua data uji yang digunakan.

Tabel 3.18 Hasil Pengujian Data Uji

Data uji	Node	*Z_in*	*Z_out*	Node	*Y_in*	*Y_out*
0.462264	1	0.623647	0.651047	1	0.776474	0.68491966
	2	0.358056	0.58857	2	0.459729	0.61294982
0.433962	1	0.61916	0.650027	1	0.773248	0.68422314
	2	0.34326	0.584982	2	0.459351	0.61286016
0.40566	1	0.614673	0.649006	1	0.770015	0.68352421
	2	0.328464	0.581386	2	0.458972	0.61277028
0.54717	1	0.637108	0.654099	1	0.786104	0.6869942
	2	0.402443	0.599275	2	0.460857	0.61321737

Setelah didapatkan nilai *y_out* dari hasil pengujian, nilai *y_out* yang nantinya digunakan sebagai nilai standar deviasi akhir dan nilai titik tengah adalah

nilai rata-rata *y_out* dari semua *y_out* dari hasil pengujian. Contoh perhitungan rata-rata y_out adalah sebagai berikut :

$$YRata-Rata = \frac{(0.6849196\,6)^2 + (0.68422314)^2 + (0.68352421)^2 + (0.6869942)^2}{4}$$

$$YRata-Rata = 0.684915302$$

Selanjutnya proses yang dilakukan adalah melakukan denormalisasi data atau mengembalikan data pada rentang nilai awal, perhitungan denormalisasi dilakukan berdasarkan persamaan 2.23. Contoh dari perhitungan denormalisasi adalah sebagai berikut :

$$Denormalisasi = 0.684915302 * (72 - 1.333333333) + 1.333333333 = 49.734014\,68$$

Tabel 3.19 Hasil *Y_out* rata-rata dan Denormalisasi

Y_rata-rata	Denormalisasi
0.684915302	49.73401468
0.612949406	44.64842472

Hasil akhir dari denormalisasi inilah yang nantinya digunakan sebagai batas fungsi keanggotaan *fuzzy* pada data Umur untuk fungsi keanggotaan rendah.

3.5 Skenario Uji Coba

Pada penelitian ini terdapat tiga skeanario uji coba yang dilakukan, ketiga skenario uji coba tersebut antara lain uji coba berdasarkan *Error* Harap, jumlah data latih, dan yang terakhir berdasarkan *learning rate* yang digunakan.

3.5.1 Skenario Berdasarkan *Error* Harap

Pengujian dengan menggunakan nilai *Error* Harap merupakan salah satu metode pengujian untuk mengetahui hasil terbaik dari pengujian berdasarkan *error* target yang ditentukan. Selain itu besarnya nilai *Error* Harap juga memiliki pengaruh terhadap kondisi berhenti dan *epoch* yang dihasilkan, dimana semakin besar *epoch* maka proses yang dilakukan dalam pengolahan data akan semakin besar.

Tabel 3.20 Contoh Skenario Uji Coba berdasarkan *Error* Harap

Pelatihan	Umur *MF* Anak		
Error Harap	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah
10^{-1}			
10^{-2}			
10^{-3}			
10^{-4}			
10^{-5}			
10^{-6}			
10^{-7}			
10^{-8}			
10^{-9}			
10^{-10}			

Tabel 3.20 merupakan skenario yang diujikan dalam proses pengujian berdasarkan nilai *Error* Harap untuk nilai *Error* Harap yang digunakan pada penelitian ini adalah 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4},10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10} sedangkan untuk jumlah data latih yang digunakan adalah 64 dan nilai learning rate sebesar 0.05 serta data uji sebanyak 16 data.

Penggunaan data latih sejumlah 64 data karena diasumsikan bahwa dengan semakin banyaknya keragaman data maka proses pelatihan *BPNN* akan semakin baik. Sedangkan penggunaan *learning rate* sebesar 0.1 karena diasumsikan bahwa semakin kecil *learning rate* maka nilai *error* yang dihasilkan akan semakin mendekati nilai *Error* Harap yang diharapkan, dimana pada *Backpropagation Neural Network* nilai *learning rate* memiliki pengaruh terhadap perbaikan bobot baru yang dilakukan dan bobot baru yang didapatkan ini memiliki pengaruh terhadap nilai *error* yang dihasilkan. Data uji yang digunakan pada penelitian ini adalah sebanyak 16, dimana data yang digunakan merupakan data yang sama untuk semua pengujian. Penggunaan jumlah data uji yang sama untuk semua

pengujian dimaksudkan untuk lebih menghasilkan nilai keluaran yang memudahkan dalam proses analisa.

Penggunaan rentang nilai *Error* Harap pada rentang nilai tersebut dikarenakan pada rentang nilai tersebut akan memberikan informasi yang jelas tentang nilai kondisi berhenti, epoch dan hasil yang didapatkan dari proses pelatihan *BPNN* karena dengan perubahan nilai *Error* Harap yang bertahap dapat dilihat perubahan epoch dan hasil yang didapatkan secara lebih jelas. Sehingga dari pengujian ini diharapkan dapat memberikan informasi tentang korelasi antara *MSE* dengan kondisi berhenti, epoch serta hasil data yang dihasilkan.

3.5.2 Skenario Berdasarkan *Learning Rate*(α)

Pengujian dengan menggunakan *learning rate* digunakan untuk mengetahui pengaruh *learning rate* yang diberikan terhadap hasil yang didapatkan, *epoch*, dan *MSE* yang didapatkan.

Tabel 3.21 Contoh Skenario Uji Coba berdasarkan *learning rate*

Pelatihan	Umur *MF* Anak			
α	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE
0.10				
0.15				
0.20				
0.25				
0.30				
0.35				
0.40				
0.45				
0.50				
0.55				
0.60				
0.65				

Pelatihan	Umur *MF* Anak			
α	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE
0.70				
0.75				
0.80				
0.85				
0.90				

Pada tabel 3.22 merupakan skenario yang diujikan dalam proses pengujian berdasarkan nilai *learning rate*, untuk nilai *learning rate* yang digunakan adalah 0.1, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70,0.75, 0.80, 0.85, 0.90.

Penggunaan nilai *learning rate* pada rentang nilai tersebut dikarenakan pada rentang nilai tersebut laju perubahan bobot dapat dilihat secara bertahap. Sehingga dari nilai tersebut dapat digunakan untuk mengetahui apakah besarnya nilai *learning rate* memiliki pengaruh terhadap kondisi berhenti, epoch dan hasil yang didapatkan, karena nilai learning memiliki pengaruh terhadap nilai bobot baru pada *Backpropagation Neural Network* dan nilai *error* yang dihasilkan.

Variabel lain yang digunakan pada penilitian ini adalah data uji, *Error* Harap, dan data latih. Penggunaan data latih pada pengujian ini adalah menggunakan data latih sebanyak 64 data. penggunaan data dengan jumlah tersebut karena diasumsikan bahwa pada rentang nilai tersebut memberikan variasi data yang lebih beragam sehingga diasumsikan akan memberikan hasil yang lebih baik. *Error* Harap yang digunakan pada pengujian ini berdasarkan *Error* Harap terbaik dari pengujian sebelumnya. Sedangkan untuk data uji yang digunakan adalah 16 dimana jumlah data uji yang digunakan adalah sama untuk semua skenario yang digunakan. Penggunaan jumlah data uji yang sama untuk semua pengujian dimaksudkan untuk lebih menghasilkan nilai keluaran yang memudahkan dalam proses analisa.

3.5.3 Skenario Berdasarkan Kombinasi Data Latih

Pengujian dengan menggunakan kombinasi data latih digunakan untuk mengetahui pengaruh banyaknya data dan keragaman data terhadap hasil perbaikan bobot yang dilakukan.

Tabel 3.22 Contoh Skenario Uji Coba berdasarkan Kombinasi Data Latih

Pelatihan	Umur *MF* Anak			
Data Latih	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE
64 data				
56 data				
48 data				
40 data				
32 data				

Tabel 3.21 merupakan skenario yang diujikan dalam proses pengujian. Penggunaan jumlah data pada rentang nilai tersebut karena pada rentang nilai tersebut memberikan rentang nilai data dan jumlah yang berbeda sehingga hasil yang didapatkan dapat dilakukan analisa tentang pengaruh dari jumlah data latih yang digunakan terhadap nilai epoch dan hasil yang didapatkan.

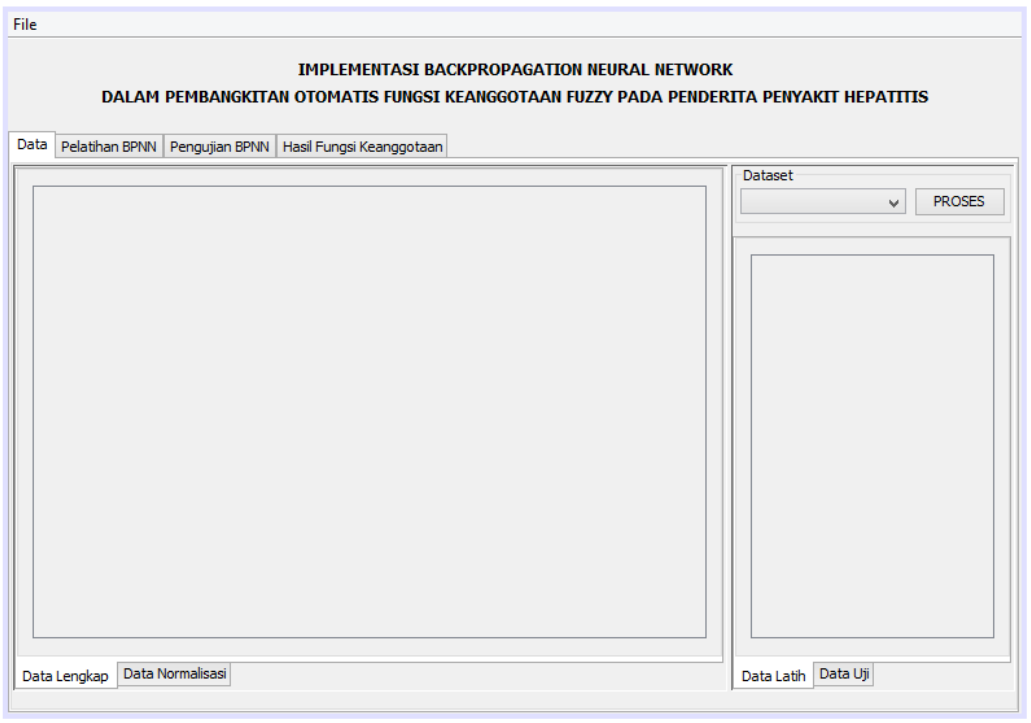
Pada pengujian ini nilai *Error* Harap dan *learning rate* yang digunakan adalah nilai terbaik dari hasil pengujian sebelumnya, dan data uji yang digunakan sebanyak 16 data, dimana data yang digunakan merupakan data yang sama untuk semua pengujian. Penggunaan jumlah data uji yang sama untuk semua pengujian dimaksudkan untuk lebih menghasilkan nilai keluaran yang memudahkan dalam proses analisa.

3.6 Rancangan Antar Muka

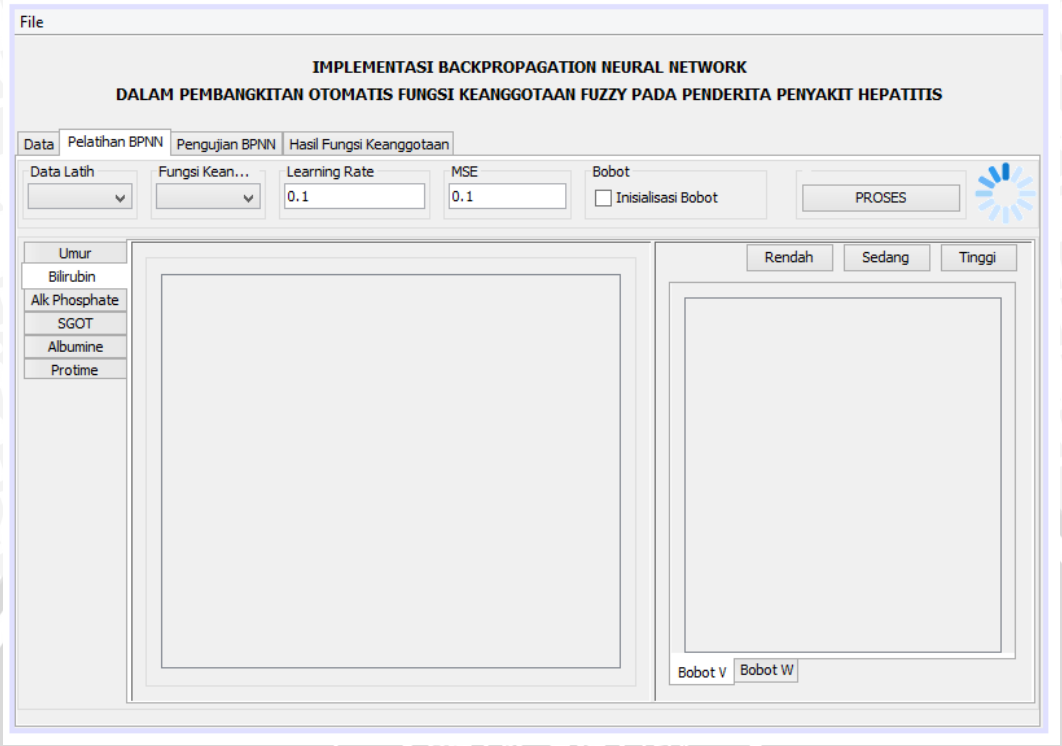
Rancangan Antar Muka digunakan untuk menggambarkan secara jelas bagian-bagian dari program, dengan dilakukannya perancangan antar muka

diharapkan mampu menghasilkan program dengan antar muka yang memudahkan pengguna dalam penggunaanya.

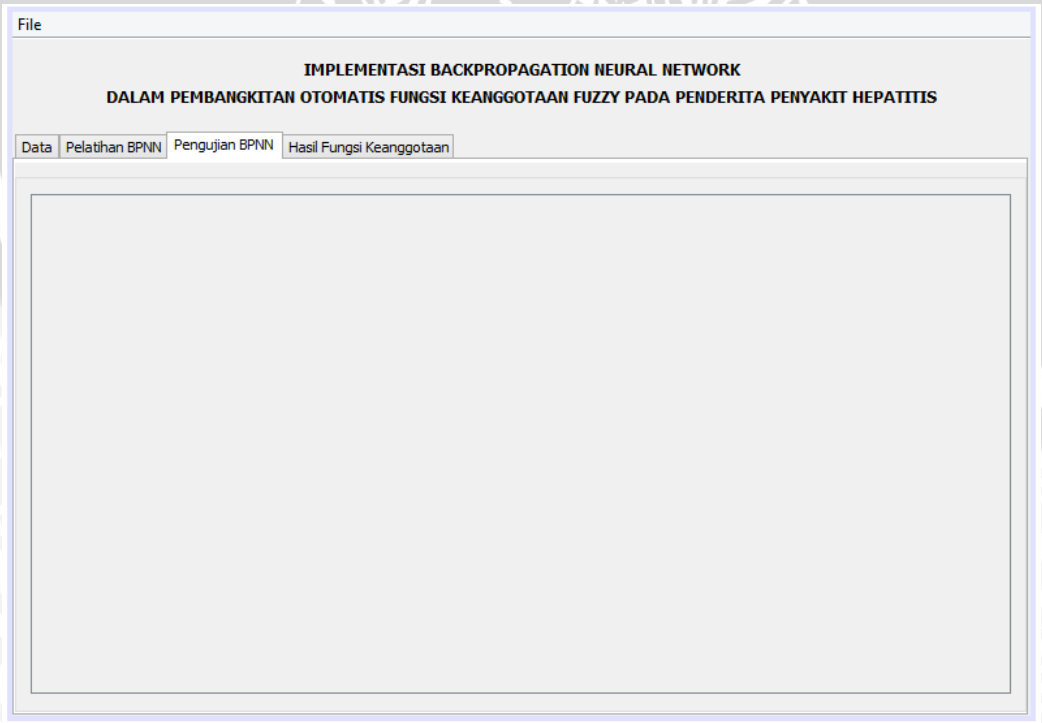
Pada program Implementasi *Backpropagation Neural Network* dalam pembangkitan otomatis fungsi keanggotaan *Fuzzy* pada data penderita hepatitis, antara muka yang digunakan dibagi dalam lima *tab*. Lima *tab* tersebut terdiri dari Data, Pelatihan *BPNN*, Pengujian *BPNN*, dan Hasil Fungsi Keanggotaan. Gambaran umum dari rancangan antar muka tersebut dapat dilihat pada gambar di bawah ini.



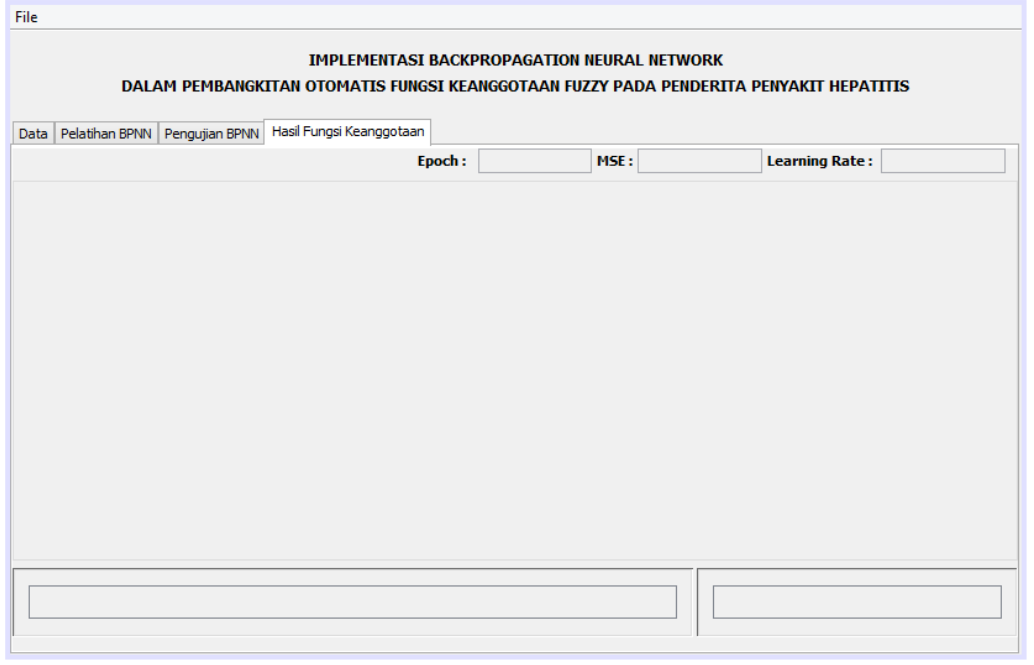
Gambar 3.15 Halaman Data



Gambar 3.16 Halaman Pelatihan *BPNN*



Gambar 3.17 Halaman Pengujian *BPNN*



Gambar 3.18 Halaman Hasil Fungsi Keanggotaan Fuzzy

3.7 Spesifikasi *Hardware* dan *Software*

Spesifikasi *Hardware* dan *Software* digunakan untuk mengetahui semua kebutuhan yang dalam pengimplementasian *Backpropagation Neural Network* untu pembangkitan otomatis fungsi keanggotaan *Fuzzy*. Kebutuhan yang diperlukan dalam penelitian ini adalah sebagai berikut:

1. Spesifikasi *Hardware*
   - Processor Intel® Core™ i3-2330M CPU @ 2.20GHz.
   - RAM 2 Gb.
   - VGA Intel HD 3000
2. Spesifikasi *Software*
   - Sistem operasi *Windows* 7 *Ultimate* 32-bit
   - Netbeans IDE 7.0
   - *Java Development Kit* (JDK) 7

BAB IV
IMPLEMENTASI

Bab ini membahas tentang Implementasi perangkat lunak yang dibangun untuk pembangkitan otomatis fungsi keanggotaan *fuzzy* pada data penderita penyakit hepatitis, berdasarkan analisa data dan perancangan yang telah dilakukan pada bab sebelumnya.

4.1 Implementasi Program

Implementasi Program akan menjelaskan tentang prosedur-prosedur yang sebelumnya telah dirancang ke dalam suatu sistem dengan menggunakan bahasa pemrogramana Java. Proses yang dilakukan dibagi dalam beberapa tahapan, dimana semua tahapan tersebut merupakan tahapan yang saling terhubung.

Tahapan awal yang dilakukan pada implementasi program adalah melakukan *upload* data ke dalam sistem dengan menggunakan file dengan ekstensi .xls. Selanjutnya setelah dilakukan proses pembacaan file, dilakukan proses normalisasi, proses penentuan bobot dan target, proses pelatihan *BPNN*, pelatihan *BPNN*, hingga melakukan proses fungsi Gaussian. Penjelasan detail dari masing-masing proses tersebut, akan dijelaskan pada masing-masing sub bab.

4.1.1 Pembacaan File

Proses yang dilakukan pada tahapan ini adalah melakukan pembacaan data excel atau data dengan eksensi .xls untuk selanjutnya dimasukkan dalam vector pada saat menggunakan bahasa pemrograman Java.

Pada tahapan ini, proses yang pertama kali dilakukan adalah melakukan baca data pada *sheet* 0. Selanjutnya dilakukan proses perulangan untuk tiap *row* pada *sheet* 0, proses tersebut dilanjutkan dengan melakukan perulangan untuk setiap *cell*. Perulangan yang dilakukan dimaksudkan untuk melakukan pengecekan setiap data, proses selanjutnya yang dilakukan adalah memasukan data ke dalam vector berdasarkan pengecekan perulangan data untuk seriap *row* dan *cell*. *Source Code 4.1* merupakan implementasi dari proses pembacaan file .xls pada bahasa pemrograman Java.

```
private Vector BacaDataExcel(String FileName) {
        Vector VectorData = new Vector();
        try {
     FileInputStream FileInputStream = new FileInputStream(FileName);
     POIFSFileSystem          PoifsFileSystem          =          new
POIFSFileSystem(FileInputStream);
     HSSFWorkbook HssfWorkBook = new HSSFWorkbook(PoifsFileSystem);
            // baca data pada sheet 0
            HSSFSheet HssfWorkSheet = HssfWorkBook.getSheetAt(0);
            Iterator RowIteration = HssfWorkSheet.rowIterator();
            //perulangan setiap row pada sheet 0
            while (RowIteration.hasNext()) {
                HSSFRow HssfRow = (HSSFRow) RowIteration.next();
                Iterator CellIteration = HssfRow.cellIterator();
                Vector VectorCellEachRow = new Vector();
                //perulangan setiap cell pada sheet 0
                while (CellIteration.hasNext()) {
                    HSSFCell HssfCell = (HSSFCell) CellIteration.next();
                    VectorCellEachRow.addElement(HssfCell);
                }
                VectorData.addElement(VectorCellEachRow);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return VectorData;
    }
```

Source Code 4.1 Pembacaan File Excel

4.1.2 Normalisasi Data

Proses awal yang dilakukan pada tahapan normalisasi data adalah melakukan perhitungan nilai maksimum dan minimum. Fungsi utama dari perhitungan nilai maksimum dan minimum untuk nantinya digunakan pada tahapan normalisasi *Max min.*

Pada tahapan pencarian nilai maksimum dan minimum proses yang pertama kali dilakukan adalah melakukan perulangan berdasarkan banyak atribut. Fungsi dari perulangan berdasarkan banyaknya atribut adalah untuk mendapatkan

nilai maksimum dan minimum berdasarkan atribut datanya, sehingga nilai minimum dan maksimum yang didapatkan adalah spesifik untuk atribut tertentu.

Selanjutnya proses perulangan yang dilakukan adalah berdasarkan banyaknya data pada atribut tertentu. Pada tahapan ini terhadap penentuan nilai maksimum dan minimum awal, dimana nilai maksimum dan minimum awal merupakan nilai data pada indeks ke 0 pada atribut tertentu. Proses selanjutnya adalah melakukan perulangan berdasarkan banyak data pada atribut tertentu.

Pada tahapan selanjutnya dilakukan proses pengecekan, jika nilai min lebih besar dari data pada indeks ke *x* maka nilai min terbaru adalah data pada indeks ke *x* tersebut. Jika nilai max lebih kecil dari data pada indeks ke x maka nilai max adalah data indeks ke *x* tersebut, dan proses ini dilakukan hingga semua data telah dilakukan pengecekan.

Proses selanjutnya adalah membandingkan nilai max dan min yang telah didapatkan dengan nilai dari target. Tahapan awal yang dilakukan adalah menentukan nilai target yang akan dibandingkan berdasarkan atribut datanya. Setelah proses tersebut, selanjutnya dilakukan perulangan sebanyak jumlah target linguistik.

Pada tahapan ini terdapat kondisi untuk menentukan nilai max dan nilai min akhir, dimana jika nilai jika nilai min lebih besar dari target pada indeks ke *x* maka nilai min terbaru adalah target pada indeks ke *x* tersebut. Jika nilai max lebih kecil dari data pada indeks ke x maka nilai max adalah target indeks ke *x* tersebut. Proses tersebut dilakukan hingga semua target dilakukan pengecekan, *source code 4.2* merupakan implementasi dari proses penentuan nilai maksimum dan minimum pada bahasa pemrograman Java.

```
public void setMaxmin(){
        for(int j=0;j<bt.jml_Attr;j++){
            double max = dok.get(0).getData(j);
            double min = dok.get(0).getData(j);
            for(int i=0;i<dok.size();i++){
                if(min>dok.get(i).getData(j))
                    min = dok.get(i).getData(j);
                if(max<dok.get(i).getData(j))
                    max = dok.get(i).getData(j);
            }
```

```
            double [][] targetLinguistik = bt.getTarget(j);
            for(int i=0;i<targetLinguistik.length;i++){
                for(int k=0;k<targetLinguistik[0].length;k++){
                    if(min>targetLinguistik[i][k])
                        min = targetLinguistik[i][k];
                    if (max < targetLinguistik[i][k])
                        max = targetLinguistik[i][k];
                }
            }
            max_min[j][0] = min;
            max_min[j][1] = max;
        }
    }
```

Source Code 4.2 Penentuan Nilai Maksimum dan Minimum

Setelah didapatkan nilai maksimum dan minimum proses selanjutnya adalah melakukan perhitungan normalisasi, dimana dalam proses perhitungan normalisasi untuk setiap atribut dilakukan pada beberapa method yang berbeda. Proses perhitungan normalisasi yang dilakukan menggunakan rumus normalisasi *max min* adalah dengan melakukan pengurangan data pada indeks tertentu dengan nilai maksimum, kemudian hasil nilai tersebut dibagi dengan hasil pengurungan antara nilai maksimum dengan minimum dari data pada atribut tersebut. *Source code 4.3* merupakan implementasi dari proses penentuan nilai maksimum dan minimum pada bahasa pemrograman Java.

```
public double getUmurNormalisasi(){
return(Dataset[0]-View.norm.max_min[0][0])/(View.norm.max_min[0][1]-
View.norm.max_min[0][0]);
}
public double getBiliNormalisasi(){
return(Dataset[1]-View.norm.max_min[1][0])/(View.norm.max_min[1][1]-
View.norm.max_min[1][0]);
}
public double getAlpNormalisasi(){
return(Dataset[2]-View.norm.max_min[2][0])/(View.norm.max_min[2][1]-
View.norm.max_min[2][0]);
}
```

```
public double getSgotNormalisasi(){
return(Dataset[3]-View.norm.max_min[3][0])/(View.norm.max_min[3][1]-
View.norm.max_min[3][0]);
}
public double getAlbuNormalisasi(){
return(Dataset[4]-View.norm.max_min[4][0])/(View.norm.max_min[4][1]-
View.norm.max_min[4][0]);
}
public double getProNormalisasi(){
return(Dataset[5]-View.norm.max_min[5][0])/(View.norm.max_min[5][1]-
View.norm.max_min[5][0]);
}
public double getDataNormalisasi(int i){
return(Dataset[i]-View.norm.max_min[i][0])/(View.norm.max_min[i][1]-
View.norm.max_min[i][0]);
}
```

Source Code 4.3 Normalisasi data

4.1.2 Penentuan Nilai Bobot dan Target

Proses penentuan bobot dilakukan dengan melakukan perulangan sebanuak jumlah hidden layer dengan batas nilai -0.5 hinga 0.5 untuk bobot V, dan perulangan berdasarkan banyaknya jumlah output layer dengan batas nilai -0.5 hingga 0.5 untuk bobot W. Pada class penentuan bobot terdapat method penentuan bobot inisialisasi, penentuan bobot inisialisasi ini nantinya digunakan untuk mengetahui pengaruh bobot terhadap epoch, kondisi berhenti, dan hasil yang didapatkan. *Source code 4.4* merupakan implementasi dari proses penentuan bobot pada bahasa pemrograman Java.

```
public void setBobotV(int k,int l) {
        int i, j;
        for (i = 0; i < BobotV[0][0].length; i++) {
            for (j = 0; j < BobotV[0][0][0].length; j++) {
                BobotV[k][l][i][j] = Math.random() - 0.5;
            }
        }
    }
public void setBobotW(int k,int l) {
        int i, j;
        for (i = 0; i < BobotW[0][0].length; i++) {
            for (j = 0; j < BobotW[0][0][0].length; j++) {
                BobotW[k][l][i][j] = Math.random() - 0.5;
```

```
            }
        }
    }
//** Bobot Inisialisasi
public void setBobotAwalV(int k,int l){
        BobotV[k][l][0][0] = 0.57;
        BobotV[k][l][0][1] = 0.14;
        BobotV[k][l][1][0] = 0.168298692;
        BobotV[k][l][1][1] = 0.534516478;
}
public void setBobotAwalW(int k,int l){
        BobotW[k][l][0][0] = 0.043427405;
        BobotW[k][l][0][1] = 0.479195033;
        BobotW[k][l][1][0] = 0.73091685;
        BobotW[k][l][1][1] = 0.104579879;
        BobotW[k][l][2][0] = 0.780318349;
        BobotW[k][l][2][1] = 0.154138073;
}
```

Source Code 4.4 Penentuan Bobot

Setelah dilakukan penentuan bobot, proses selanjutnya yang dilakukan adalah melakukan penentuan nilai target dimana pada tahapan ini proses penentuan target dilakukan pada setiap method yang berbeda. Method yang berbeda tersebut mewakali atribut yang digunakan pada penelitian ini.

Pada setiap method tersebut berisi variabel target yang digunakan, variabel yang digunakan merupakan data dalam bentuk array dua dimensi dimana indeks array yang pertama mewakili fungsi keanggotaan dari atribut data tersebut dan indeks yang kedua mewakili nilai target yang digunakan. Nilai target yang digunakan pada penilitian terdiri dari dua nilai yaitu nilai standar deviasi untuk dan nilai titik tengah. *Source code 4.5* merupakan implementasi dari proses penentuan target pada bahasa pemrograman Java.

```
public void setTargetUmur() {
        TargetUmur[0][0] = 2.66666666666667;
        TargetUmur[0][1] = 8;
        TargetUmur[1][0] = 1.333333333;
        TargetUmur[1][1] = 16;
        TargetUmur[2][0] = 6.666666667;
        TargetUmur[2][1] = 38;
        TargetUmur[3][0] = 5;
```

```
        TargetUmur[3][1] = 65;
}
public void setTargetBili() {
        TargetBili[0][0] = 0.066666667;
        TargetBili[0][1] = 0.2;
        TargetBili[1][0] = 0.2;
        TargetBili[1][1] = 0.9;
        TargetBili[2][0] = 0.566666667;
        TargetBili[2][1] = 3;
}
public void setTargetAlk() {
        TargetAlk[0][0] = 3.333333333;
        TargetAlk[0][1] = 30;
        TargetAlk[1][0] = 18.66666667;
        TargetAlk[1][1] = 100;
        TargetAlk[2][0] = 10.66666667;
        TargetAlk[2][1] = 147;
}
public void setTargetSgot() {
        TargetSgot[0][0] = 1.333333333;
        TargetSgot[0][1] = 4;
        TargetSgot[1][0] = 5.666666667;
        TargetSgot[1][1] = 25;
        TargetSgot[2][0] = 5.333333333;
        TargetSgot[2][1] = 50;
}
public void setTargetAlbu() {
        TargetAlbu[0][0] = 0.666666667;
        TargetAlbu[0][1] = 2;
        TargetAlbu[1][0] = 0.2;
        TargetAlbu[1][1] = 4;
        TargetAlbu[2][0] = 0.2;
        TargetAlbu[2][1] = 6;
}
public void setTargetPro() {
        TargetPro[0][0] = 1;
        TargetPro[0][1] = 8;
        TargetPro[1][0] = 0.666666667;
        TargetPro[1][1] = 12;
        TargetPro[2][0] = 12.16666667;
        TargetPro[2][1] = 50;
}
```

Source Code 4.5 Penentuan target

Pada penelitian ini, semua data yang digunakan merupakan data yang dinormalisasi, sehingga pada data target juga dilakukan proses normalisasi untuk menyeragamkan data awal. Proses yang dilakukan pada tahap normalisasi ini sama dengan proses yang dilakukan pada tahapan normalisasi data. *Source code* 4.6 merupakan implementasi dari proses normalisasi target pada bahasa pemrograman Java.

```
public double getTargetUmurNorm(int i, int j) {
return TargetUmur[i][j] = (TargetUmur[i][j] - View.norm.max_min[0][0]) /
(View.norm.max_min[0][1] - View.norm.max_min[0][0]);
}

public double getTargetBiliNorm(int i, int j) {
return TargetBili[i][j] = (TargetBili[i][j] - View.norm.max_min[1][0]) /
(View.norm.max_min[1][1] - View.norm.max_min[1][0]);
}

public double getTargetAlkNorm(int i, int j) {
return  TargetAlk[i][j]  =  (TargetAlk[i][j]-  View.norm.max_min[2][0])  /
(View.norm.max_min[2][1] - View.norm.max_min[2][0]);
}

public double getTargetSgotNorm(int i, int j) {
return TargetSgot[i][j] = (TargetSgot[i][j] - View.norm.max_min[3][0]) /
(View.norm.max_min[3][1] - View.norm.max_min[3][0]);
}

public double getTargetAlbuNorm(int i, int j) {
return TargetAlbu[i][j] = (TargetAlbu[i][j] - View.norm.max_min[4][0]) /
(View.norm.max_min[4][1] - View.norm.max_min[4][0]);
}

public double getTargetProNorm(int i, int j) {
return  TargetPro[i][j]  =  (TargetPro[i][j]  -  View.norm.max_min[5][0])  /
(View.norm.max_min[5][1] - View.norm.max_min[5][0]);
}
```

Source Code 4.6 Proses normalisasi target

4.1.4 Pelatihan *BPNN*

Pada tahapan pelatihan proses awal yang dilakukan adalah menetapkan Boolean pelatihan. Fungsi dari penetepan *Boolean* pelatihan adalah untuk menentukan apakah proses perulangan tetap dilakukan atau proses perulangan dihentikan. Nilai awal yang ditetapkan pada *Boolean* pelatihan adalah *true*.

Proses selanjutnya yang dilakukan adalah melakukan perulangan berdasarkan banyaknya data. Setelah dilakukan perulangan berdasarkan banyaknya data, selanjutnya dilakukan perulangan berdasarkan jumlah node dari *node hidden layer* untuk dilakukan perhitungan nilai *Z_in* dan setelah didapatkan nilai *Z_in*, nilai dari *Z-in* tersebut dilakukan aktivasi untuk mendapatkan nilai *Z_out.*

Proses perulangan selanjutnya yang dilakukan adalah berdasarkan banyaknya jumlah *node output layer*. Hasil *Z_out* yang telah didapatkan selanjutanya digunakan untuk dilakukan perhitungan untuk mendapatkan nilai *Y_in.* Nilai *Y_in* yang telah didapatkan tersebut selanjutnya dilakukan aktivasi untuk mendapatkan nilai *Y_out.* Tahapan yang dilakukan dari proses pencarian nilai *Z_in* hingga mendapatkan nilai *Y_out* merupakan tahapan *feedforward* dari proses pelatihan *backpropagation neural network.*

Setelah dilakukan proses *feedforward* proses selanjutnya yang dilakukan adalah melakukan tahapan *backpropagation*, dimana tahapan awal yang dilakukan adalah melakukan perulangan berdasarkan banyaknya *node* pada *output layer* untuk melakukan perhitungan nilai *errorOutY*, dan dari nilai *errorOutY* yang didapatkan tersebut selanjutnya dilakukan perhitungan untuk mendapatkan nilai *MSE*. Nilai *MSE* yang didapatkan memiliki pengaruh terhadap proses pelatihan yang dilakukann, jika nilai *MSE* lebih besar dari nilai *MSE* target maka proses pelatihan tetap dilakukan, namun jika tidak proses pelatihan dihentikan.

Tahapan yang dilakukan setelah mendapatkan nilai *MSE* adalah melakukan perulangan berdasarkan banyak jumlah *hidden layer* untuk mendapatkan nilai *errorOutZ.* Nilai *errorOutY* dan nilai *errorOutZ* yang telah didapatkan tersebut nantinya digunakan pada proses perhitungan bobot baru.

Proses selanjutnya yang dilakukan adalah melakukan perulangan berdasarkan banyaknya jumlah *node output layer* untuk melakukan perbaikan

bobot W, dimana pada tahapan ini dilakukan penjumlah bobot lama ditambah hasil perkalian dari *learning rate*, *errorOutY*, dan nilai *Z_out*. Sedangkan untuk bobot bias *W* dilakukan proses penjumlah bobot bias lama ditambah hasil perkalian dari *learning rate*, dan *errorOutY*.

Setelah dilakukan perbaikan bobot *W*, proses selanjutnya yang dilakukan adalah melakukan perbaikan bobot *V*, dimana pada tahapan ini dilakukan perulangan berdasarkan banyaknya jumlah *node* pada *hidden layer*. Proses yang dilakukan dalam perbaikan nilai bobot *V* adalah dengan melakukan penjumlahan bobot lama V dengan hasil perkalian dari *learning rate*, *errorOutZ*, dan nilai data. Sedangkan untuk perhitungan bobot bias *V* baru adalah dengan melakukan bobot lama V dengan hasil perkalian dari *learning rate*, dan *errorOutZ*. *Source code* 4.7 merupakan implementasi dari proses pelatihan pada bahasa pemrograman Java.

```
public void setPelatihan(double a, double msetarget, int l, int m) {
        int i, j, k;
        boolean pelatihan = true;
        int n = 0;

        while (pelatihan) {
            boolean cekPelatihan = false;
            for (k = 0; k < dok.size(); k++) {
                //**Alur Maju
                double target[] = bt.getTargetNorm(l, m);
                //**Z_in
                for (i = 0; i < bt.jml_node_hl; i++) {
                    for (j = 1; j < bt.jml_node_il+1; j++) {
      Z[i] = dok.get(k).getDataNormalisasi(l) * bt.getBobotV(l, m, j, i);
                    }
                    Z[i] += bt.getBobotV(l, m, 0, i);
                    //**Z_out
                    Z[i] = 1 / (1 + Math.exp(-Z[i]));
                }
                //**Y_in
                for (i = 0; i < bt.jml_node_ol; i++) {
                    Y[i]=0;
                    for (j = 1; j < bt.jml_node_hl+1; j++) {
                        Y[i] += Z[j-1] * bt.getBobotW(l, m, j, i);
                    }
```

```
            Y[i] += bt.getBobotW(l, m, 0, i);
            //**Y_out
            Y[i] = 1 / (1 + Math.exp(-Y[i]));
        }
        //**Alur Mundur & mse
        double[] errorOutY = new double[bt.jml_node_ol];
        double mse = 0;
        for (i = 0; i < bt.jml_node_ol; i++) {
        errorOutY[i] = (target[i] - Y[i]) * (Y[i]) * (1 - Y[i]);
            mse += Math.pow(errorOutY[i], 2);
        }
        mse /= bt.jml_node_ol;
        if (mse >= msetarget) {
            cekPelatihan = true;
        }

        double[] errorOutZ = new double[bt.jml_node_hl];
        for (i = 1; i < bt.jml_node_hl+1; i++) {
            errorOutZ[i-1] = 0;
            for (j = 0; j < bt.jml_node_ol; j++) {
    errorOutZ[i-1] += (errorOutY[j] * bt.getBobotW(l, m, i, j));
            }
        }
        //**perbaikan bobot W
        for (i = 0; i < bt.jml_node_ol; i++) {
            for (j = 1; j < bt.jml_node_hl+1; j++) {
      bt.setBobotW(l, m, j, i, (bt.getBobotW(l, m, j, i) + (a *
      errorOutY[i] *
      Z[j-1])));
            }
      bt.setBobotW(l, m, 0, i, bt.getBobotW(l, m, 0, i) + (a *
             errorOutY[i]));
        }
        //**perbaikan bobot V
        for (i = 0; i < bt.jml_node_hl; i++) {
            for (j = 1; j < bt.jml_node_il+1; j++) {
            }
        bt.setBobotV(l, m, 0, i, (bt.getBobotV(l, m, 0, i) + (a *
        errorOutZ[i])));
        }
    }
    n++;
```

```
            setEpoch(n);
             if (!cekPelatihan) {
                 pelatihan = false;
             }
        }
    }
```

Source Code 4.7 Pelatihan *BPNN*

4.1.5 Pengujian *BPNN*

Proses yang dilakukan pada tahapan pengujian *BPNN*, secara keseluruhan sama dengan tahapan *feedforward* yang dilakukan pada tahapan pelatihan. Perbedaaan dari kedua tahapan tersebut hanyalah pada nilai bobot yang digunakan, dimana pada tahapan pengujian *BPNN* bobot yang digunakan merupakan bobot hasil perbaikan yang sebelumnya telah dilakukan pada tahap pelatihan *BPNN*. . *Source code* 4.8 merupakan implementasi dari proses pengujian pada bahasa pemrograman Java.

```
public double[] getPengujian(int l, int m, int k){
         double zUji[] = new double[BobotTarget.jml_node_hl];
         double yUji[]=new double[BobotTarget.jml_node_ol];
             int i, j;
         for (i = 0; i < bt.jml_node_hl; i++) {
             for (j = 1; j < bt.jml_node_il + 1; j++) {
   zUji[i] = dok.get(k).getDataNormalisasi(l) * bt.getBobotV(l, m, j, i);
             }
             zUji[i] += bt.getBobotV(l, m, 0, i);
             zUji[i] = 1 / (1 + Math.exp(-zUji[i]));
         }
         //**Y_in
         for (i = 0; i < bt.jml_node_ol; i++) {
             yUji[i]=0;
             for (j = 1; j < bt.jml_node_hl + 1; j++) {
                 yUji[i] += zUji[j - 1] * bt.getBobotW(l, m, j, i);
             }
             yUji[i] += bt.getBobotW(l, m, 0, i);
             //**Y_out
             yUji[i] = 1 / (1 + Math.exp(-yUji[i]));
         }
         return yUji;
     }
```

Source Code 4.8 Pengujian *BPNN*

4.1.6 Denormalisasi

Proses yang dilakukan pada tahapan ini adalah merupakan nilai normalisasi ke dalam rentang nilai awal dari data tersbut. Pada tahap denormalisasi, nilai yang dilakukan proses denormalisasi adalah nilai *Y* rata-rata yang sebelumnnya didapatkan dari proses pengujian *BPNN*. Proses perulangan pada tahap ini dilakukan berdasarkan banyaknya jumlah *node* pada *output layer*, dan di dalam perulangan tersebut selanjutnya dilakukan perhitungan nilai denormalisasi. *Source code* 4.9 merupakan implementasi dari proses pengujian pada bahasa pemrograman Java.

```
public double [] getDNormalisasi(double a[], int l){
        int i;
        yUji=a;
        for(i=0;i<BobotTarget.jml_node_ol;i++){
        yUjiNorm[i] =
(yUji[i]*(norm.max_min[l][1]norm.max_min[l][0]))+norm.max_min[l][0];
        }
        return yUjiNorm;
    }
```

Source Code 4.9 Denormalisasi

4.1.7 Fungsi Gaussian

Proses yang dilakukan pada tahapan ini adalah melakukan perhitungan nilai derajat keanggotaan berdasarkan fungsi Gaussian. Nilai x pada tahapan ini merupakan nilai data yang dicari nilai derajat keanggotaannya, sedangkan nilai c dan s merupakan nilaia titik tengah dan standar deviasi yang sebelumnya telah didapatkan dari proses *backpropagation neural network. Source code* 4.10 merupakan implementasi dari proses pengujian pada bahasa pemrograman Java.

```
public static double getGaussian(double x, double c,double s) {
        double z = Math.exp(-0.5 * Math.pow((x - c) / s, 2));
        return z;
    }
```

Source Code 4.10 Fungsi Gaussian

4.1.8 Pembentukan Kurva Gaussian

Pada tahapan ini proses yang dilakukan adalah melakukan proses pembentukan grafik perdasarkan nilai standar deviasi dan titik tengah yang

didapatkan dari proses *backpropagation neural network.* Pada proses perulangan awal dilakukan berdasarkan banyaknya kelas, hal ini dimaksudkan untuk mendapatkan hasil keluaran berdasarkan kelas tertentu.

Proses perulangan selanjutnya adalah perulangan berdasarkan nilai batas bawa hingga batas atas. Pada proses perulangan tersebut selanjutnya dilakukan perhitungan nilai derajat keanggotaan untuk nilai x tertentu, dan proses tersebut dilakukan hingga semua data telah dihitung nilai derajat keanggotaanya. *Source code* 4.11 merupakan implementasi dari pembentukan kurva Gaussian pada bahasa pemrograman Java.

```
//**Grafik
double yDNorm[] = dnorm.getDNormalisasi(yAkhir, atribut_data.getSelectedIndex());
 grafik [atribut_data.getSelectedIndex()][fungsianggota.getSelectedIndex()] = yDNorm;
      XYSeriesCollection dataset = new XYSeriesCollection();

      for(int i=0;i<bt.jml_kelas;i++){
      XYSeries series = new
     XYSeries(getNamaMF(atribut_data.getSelectedIndex(), i));
      double batasAwal = grafik [atribut_data.getSelectedIndex()][i][1]-(3*grafik
      [atribut_data.getSelectedIndex()][i][0]);
      double batasAkhir = grafik [atribut_data.getSelectedIndex()][i][1]+(3*grafik
      [atribut_data.getSelectedIndex()][i][0]);

       for(double xy = batasAwal;xy<=batasAkhir;xy+=0.01){
       series.add(xy,
      gaus.getGaussian(xy,grafik[atribut_data.getSelectedIndex()][i][1],
       grafik [atribut_data.getSelectedIndex()][i][0]));
       }
      dataset.addSeries(series);
       }
            JFreeChart chart = ChartFactory.createXYLineChart(
            "Grafik " + getNamaAtribut(atribut_data.getSelectedIndex()) , // Title
             "Nilai Linguistik", // x-axis Label
              "Derajat Keanggotaan", // y-axis Label
              dataset, // Dataset
              PlotOrientation.VERTICAL, // Plot Orientation
```

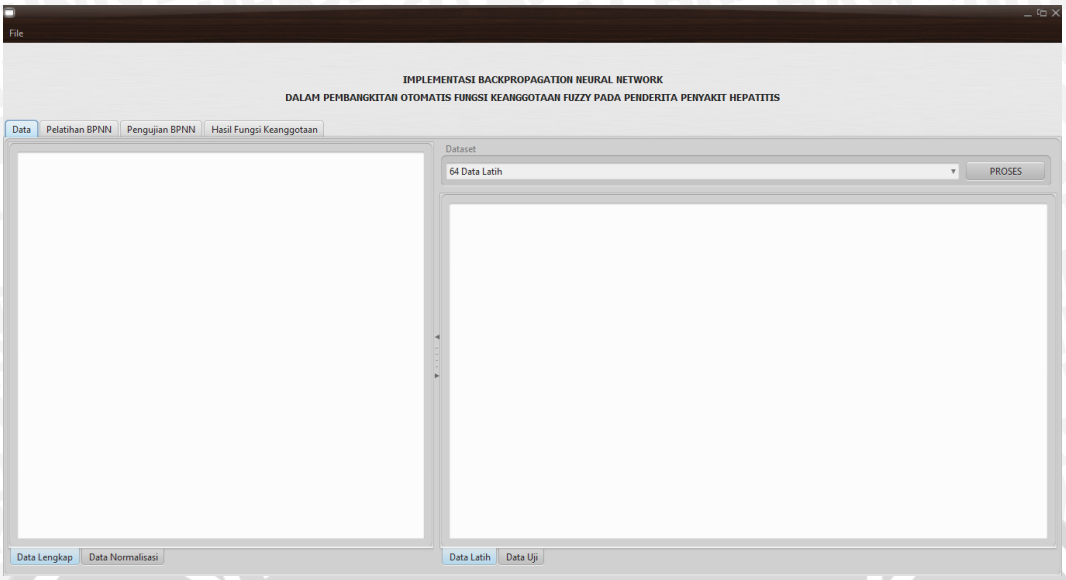
```
            true, // Show Legend
            true, // Use tooltips
           false // Configure chart to generate URLs?
           );
          XYPlot plot = chart.getXYPlot();
          XYLineAndShapeRenderer renderer = new
         XYLineAndShapeRenderer();
          renderer.setSeriesPaint(2, java.awt.Color.MAGENTA);
          renderer.setBaseShapesVisible(false);
          renderer.setBaseOutlineStroke(new BasicStroke(1.0f));
          plot.setRenderer(renderer);
          plot.setBackgroundAlpha(0.2f);
      ChartPanel CP = new ChartPanel(chart);
      fungsiKeanggotaan.removeAll();
      fungsiKeanggotaan.add(CP);
```

Source Code 4.11 Pembentukan Kurva Gaussian

4.2 Impementasi Antar Muka

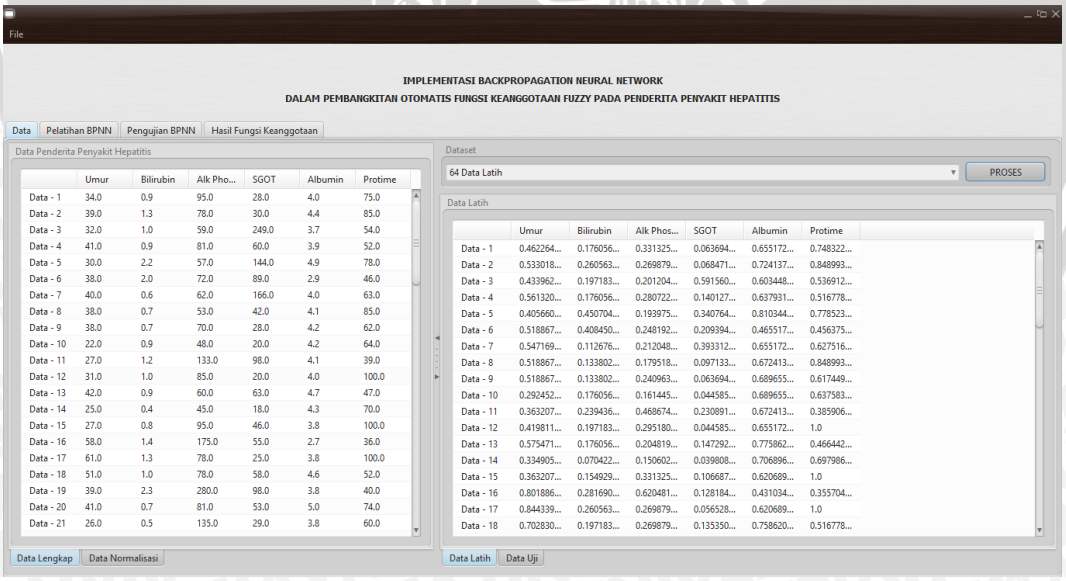
Proses yang dilakukan pada tahapan ini adalah menampiilkan antar mua yang digunakan pada sistem. Antar muka yang diimplementasikan pada tahapan ini sebelumnya telah dilakukan perancangan pada bab sebelumnya. Pada implementasi program pada penelitian ini, antar muka dari sistem terdiri 4 *tab* yaitu data, pelatihan *BPNN*, pengujian *BPNN*, dan Hasil Fungsi Keanggotaan.

Antar Muka data merupakan antar muka utama dari sistem, dimana pada antar muka data dilakukan proses *upload* data, normalisasi data, dan penentuan jumlah data latih yang digunakan. Gambar 4.1 merupakan antar muka awal dari antar muka data.



Gambar 4.1 Antar Muka Data

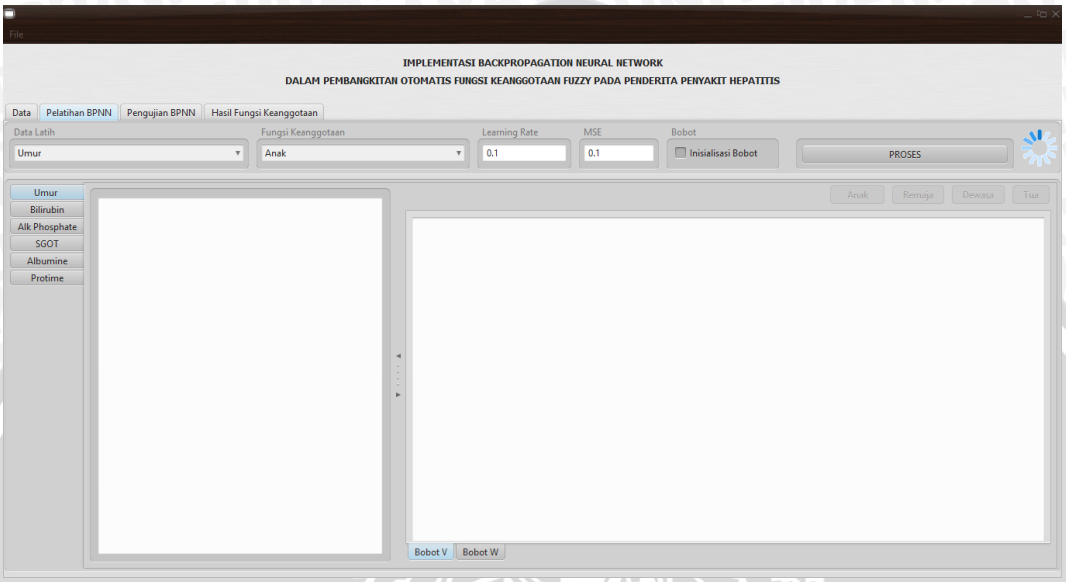
Saat pertama kali program dijalankan, pada antar muka data tidak menampilkan informasi apapun pada program. Proses yang perlu dilakukan untuk dapat menjalankan program adalah mengklik *button* proses pada program. Setelah *button* proses diklik, pada antar muka data akan menampilkan data secara keseluruhan, data hasil normalisasi, data latih, dan data uji. Gambar 4.2 merupakan antar muka data setelah menjalankan pemrosesan data.



Gambar 4.2 Antar Muka Data setelah pemrosesan Data

Pada gambar 4.2 ditampilkan informasi yang dibutuhkan pada sistem, seperti yang telah dijelaskan sebelumnya bahwa informasinya yang ditampilkan

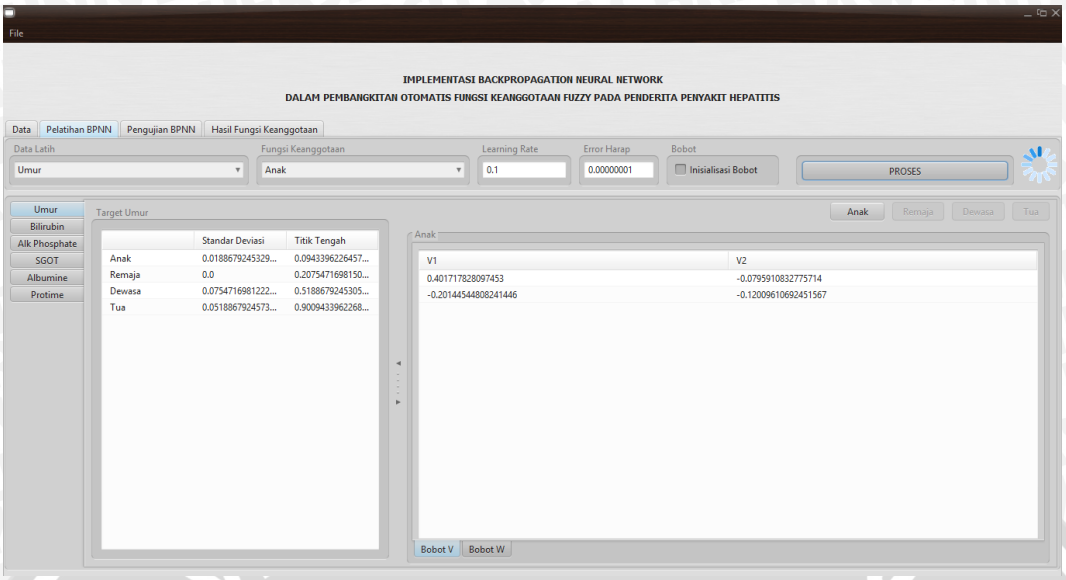
adalah data secara keseluruhan, data normalisasi, data latih, dan uji. Jumlah data latih yang ditampilkan pada sistem dipengaruhi jumlah data yang dipilih saat menjalankan *button* proses, sedangkan untuk data uji jumlah yang ditampilkan adalah tetap yaitu sebanyak 16 data.



Gambar 4.3 Antar Muka Pelatihan *BPNN*

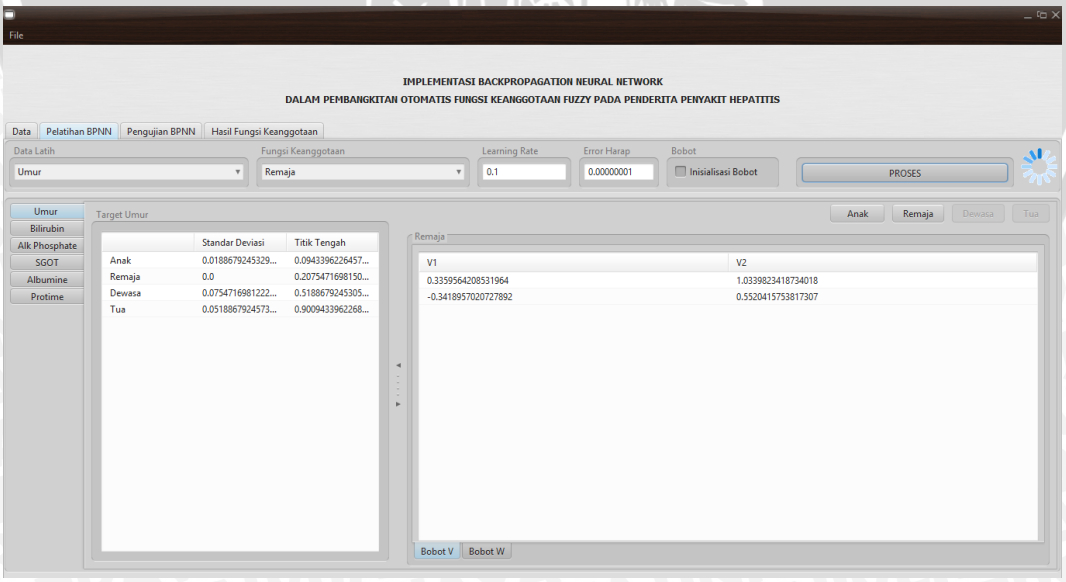
Pada gambar 4.3 merupakan antar muka awal dari pelatihan *BPNN*. Pada antar muka ini proses yang dilakukan adalah untuk melakukan proses pelatihan dan pengujian dari *Backpropagation Neural Network*. Informasi yang ditampilkan pada antar muka pelatihan *BPNN* adalah data target untuk semua atribut, bobot v yang telah diperbaharui, dan bobot w yang telah diperbaharui.

Pada antar muka pelatihan, *user* nantinya harus memasukan beberapa *inputan* untuk nantinya digunakan dalam proses pelatihan dan pengujian *BPNN*. Beberapa inputan yang dimasukkan antara lain nilai *MSE* dan *learning rate*. *MSE* dan *learning rate* memiliki pengaruh berkaitan tentang jumlah epoch yang dilakukan pada proses pelatihan, selain nilai *MSE* dan *learning rate* memiliki pengaruh pada bobot baru yang didapatkan serta nilai *Y* didapatkan. Selain 2 paramater inputan tersebut, user nantinya juga dapat memilih data, dan fungsi keanggotaan yang akan dihitung dengan menggunakan *Backpropagation Neural Network*.



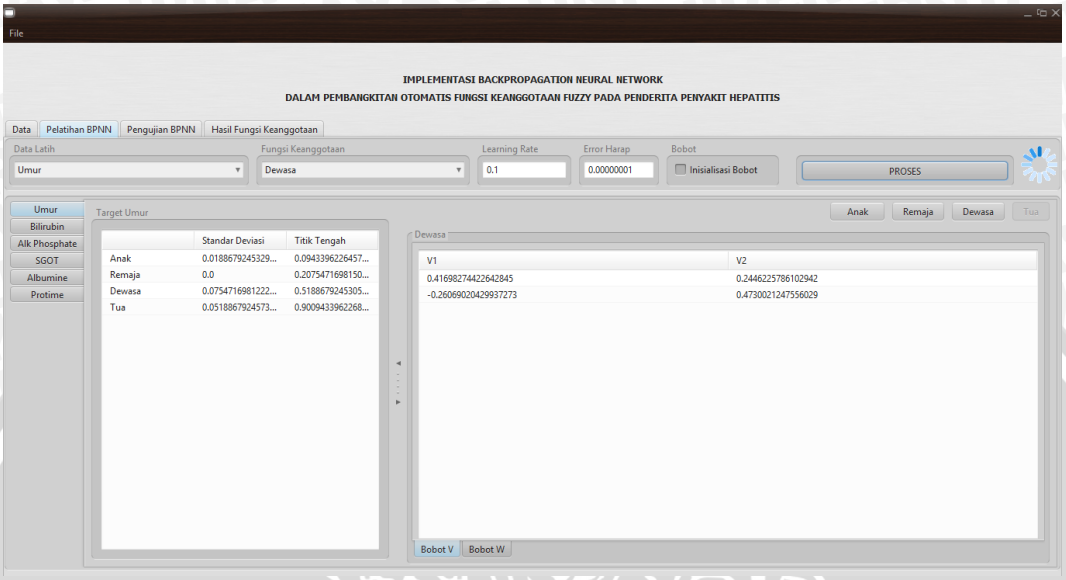
Gambar 4.4 Antar Muka Pelatihan *BPNN* setelah pelatihan *BPNN*

Gambar 4.4 merupakan antar muka pelatihan BPNN setelah dilakukan pemrosesan pelatihan dengan parameter yang digunakan yaitu data umur, fungsi keanggotaan anak, nilai *learning rate* 0.1, dan nilai *MSE* sebesar 10^{-8}. Pada gambar 4.4 dapat dilihat bahwa setelah dilakukan pemrosesan pelatihan, antar muka pelatihan akan menampilkan informasi antara lain nilai target, bobot v hasil pelatihan, dan bobot w pelatihan berdasarkan parameter yang dimasukkan pada sistem.



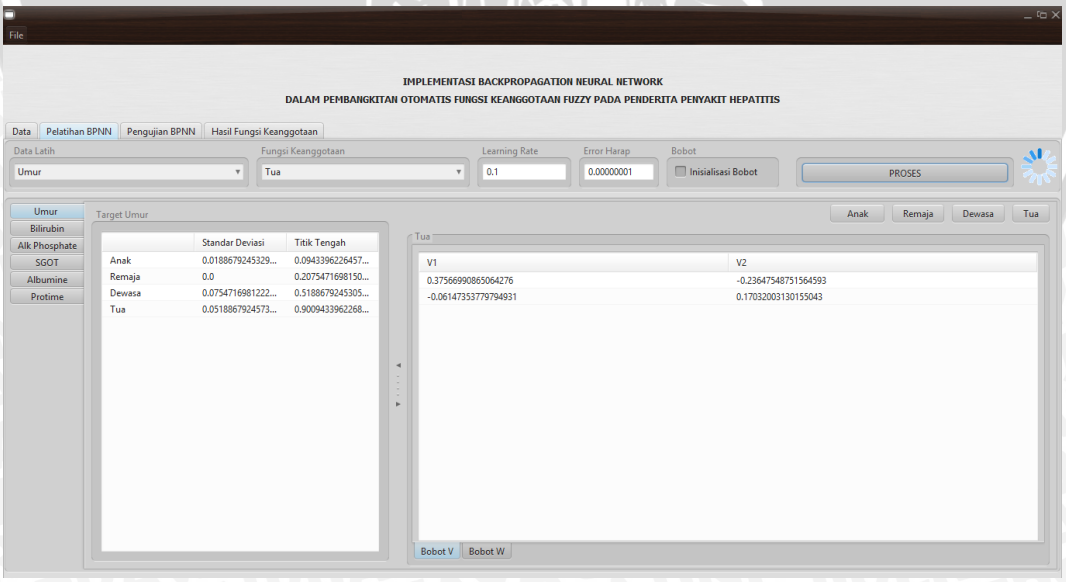
Gambar 4.5 Antar Muka Pelatihan *BPNN* setelah pelatihan *BPNN*

Gambar 4.5 merupakan antar muka pelatihan BPNN setelah pemrosesan data umur fungsi keanggotaan remaja. Seperti pada gambar 4.4, pada antar muka ini juga menampilkan bobot v dan bobot w terbaru dari hasil pelatihan BPNN.



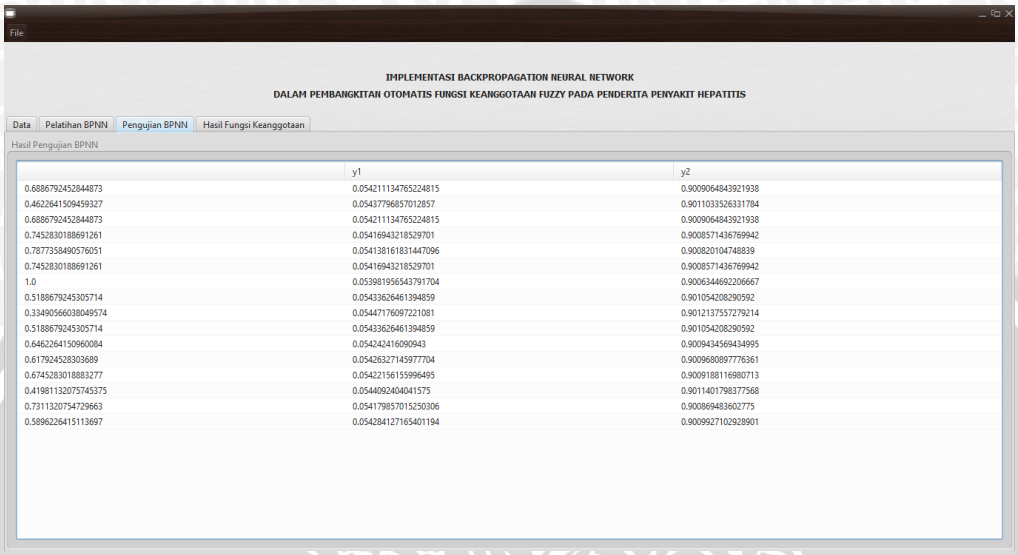
Gambar 4.6 Antar Muka Pelatihan BPNN setelah pelatihan *BPNN*

Gambar 4.6 merupakan antar muka pelatihan *BPNN* setelah pemrosesan data umur fungsi keanggotaan dewasa. Informasi yang ditampilkan pada antar muka ini sama seperti yang ditampilkan pada gambar 4.4 dan gambar 4.5.



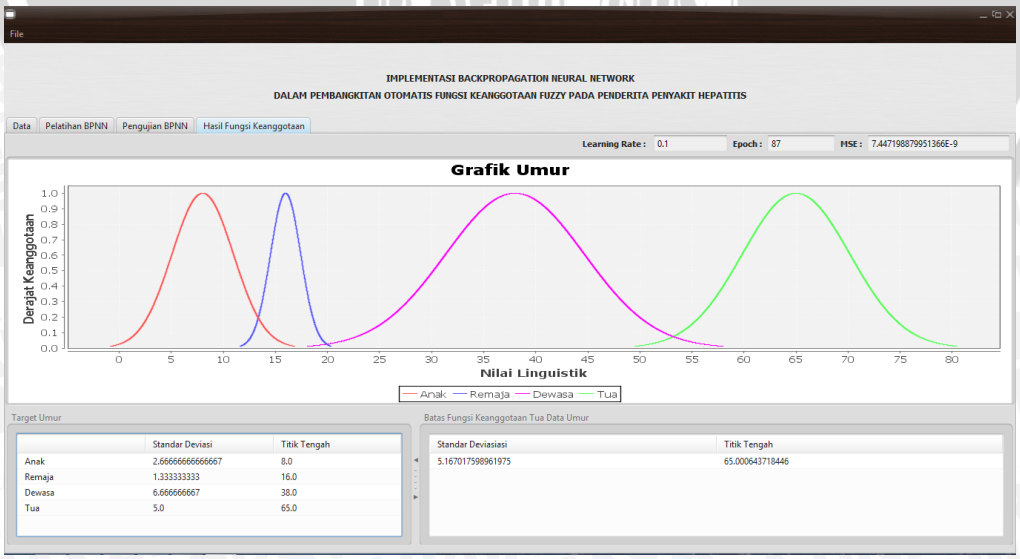
Gambar 4.7 Antar Muka Pelatihan BPNN setelah pelatihan *BPNN*

Gambar 4.7 merupakan antar muka pelatihan *BPNN* setelah pemrosesan data umur fungsi keanggotaan tua. Informasi yang ditampilkan pada antar muka ini sama dengan informasi yang ditampilkan pada gambar 4.4, gambar 4.5, dan gambar 4.6.



Gambar 4.8 Antar Muka Pengujian

Pada gambar 4.8 merupakan antar muka pengujian, pada antar muka pengujian *BPNN* ditampilkan informasi nilai Y untuk semua data uji yang digunakan. Pada penilitian ini jumlah data uji yang digunakan untuk semua parameter adalah sama, dimana jumlah data uji yang digunakan adalah sebanyak 16 data uji.



Gambar 4.9 Antar Muka Hasil Fungsi Keanggotaan

Pada gambar 4.9 merupakan antar muka hasil fungsi keanggotaan. Pada antar muka ditampilkan grafik untuk setiap fungsi keanggotaan, nilai target, nilai standar deviasi, dan nilai titik tengah. Kurva yang ditampilkan pada antar muka ini merepresentasikan batas fungsi keanggotaan dari data umur, dimana pada kurva tersebut ditampilkan informasi tentang nilai derajat keanggotaan pada nilai data tertentu,

Informasi lain yang ditampilkan pada halaman hasil fungsi keanggotaan adalah nilai target, dimana nilai target ini digunakan untuk mengetahui apakah nilai Y yang didapatkan telah mendekati nilai target yang diharapkan. Selain nilai target, informasi tentang nilai standar deviasi dan nilai titik tengah juga ditampilkan. Nilai standar deviasi dan titik tengah merupakan nilai *Y* rata-rata dari hasil pengujian *BPNN*, dan nilai inilah yang nantinya digunakan sebagai batas dalam pembentukan kurva Gaussian.

BAB V
PENGUJIAN DAN ANALISIS

Bab ini mambahas tentang tahapan pengujian dan analisis implementasi *Backpropagation Neural Network* dalam pembangkitan otomatis fungsi keanggotaan *fuzzy* pada penderita penyakit hepatits Proses pengujian dilakukan dengan mengevaluasi hasil uji coba sesuai dengan perancangan uji coba yang telah ditentukan sebelumnya.

5.1 Hasil dan Analisis Uji *Error* Harap

Pada tahapan ini proses pengujian yang dilakukan adalah berdasarkan nilai *Error* harap, dimana pada pengujian ini dilakukan kombinasi nilai *Error harap* untuk mengetahui *Error* harap terbaik yang didapatkan berdasarkan beberapa kombinasi tertentu. Nilai awal yang digunakan pada *Error harap* adalah 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4},10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}.

Penggunaan nilai *Error harap* pada rentang nilai tersebut dimaksudkan untuk mengetahui perubahan yang dihasilkan pada nilai *epoch* dan hasil keluaran yang didapatkan secara bertahap. Karena dengan menampilkan informasi secara jelas untuk setiap tahap yang diujikan, diharapkan dapat memudahkan proses analisa tentang pengaruh besarnya nilai *Error harap* dan memudahkan dalam menentukan *Error harap* terbaik..

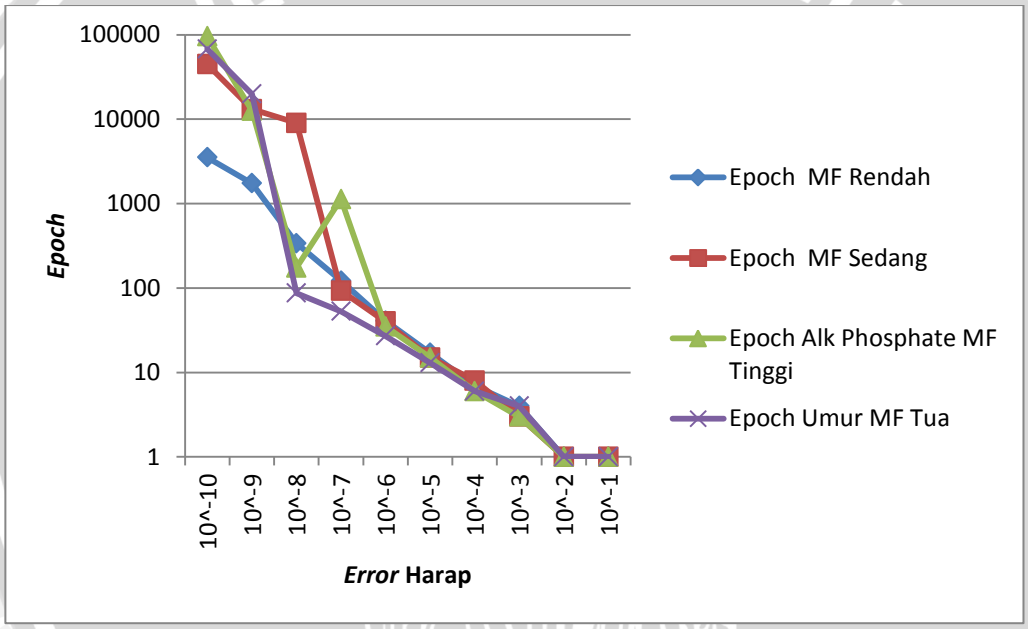
Parameter lain yang digunakan pada uji *Error harap* jumlah data latih dan nilai *learning rate.* Seperti yang telah dijelaskan pada bab perancangan untuk jumlah data yang digunakan adalah 64 data dan untuk *learning rate* yang digunakan adalah 0.1. Penggunaan jumlah data sebanyak 64 data karena diasumsikan bahwa semakin banyak data latih maka akan semakin memberikan data yang beragam, dan data yang beragam diasumsikan akan memberikan hasil yang lebih baik dalam proses pelatihan. Sedangkan nilai *learning rate* sebesar 0.1 karena diasumsikan bahwa semakin kecil nilai *learning rate* maka nilai *error* yang didapatkan akan semakin mendekati *error* target sehingga berpengaruh pada kondisi berhenti dan nilai epoch yang dihasilkan.

Tabel 5.1 Hasil Pengujian Uji Skenario *Error* Harap pada Bobot Acak

Pelatihan	Bilirubin *MF* Rendah			Sgot *MF* Sedang			Alk Phosphate *MF* Tinggi			Umur *MF* Tua		
Error Harap	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah
10^-1	1	1.513205	1.417219	1	116.7799	130.3917	1	80.32153	153.1756	1	22.97434	50.15369
10^-2	1	1.194821	1.144732	1	92.55183	127.9747	1	76.87136	162.245	1	17.735	45.98511
10^-3	4	0.77066	0.79358	3	72.32785	78.60244	3	52.90278	138.7252	4	12.91011	58.2534
10^-4	7	0.500047	0.570273	8	41.39066	50.12122	6	36.27197	147.0698	6	10.1585	61.29825
10^-5	17	0.335338	0.391351	15	28.83979	36.49527	15	24.57942	146.9445	13	7.510526	63.73324
10^-6	41	0.225255	0.277668	40	18.39813	28.82443	35	17.72026	147.0305	27	6.159745	64.77994
10^-7	122	0.158629	0.22413	93	12.37737	25.31105	1128	10.58451	146.9694	53	5.48451	64.96749
10^-8	338	0.121412	0.201415	8996	5.650635	24.95228	174	11.53288	147.0034	87	5.117622	65.00721
10^-9	1740	0.088789	0.199095	13095	5.666871	24.98494	12571	10.65192	147.0033	19830	4.995784	65.00932
10^-10	3525	0.082205	0.199944	44750	5.665422	25.00503	95338	10.65703	146.9985	67626	4.996236	64.99787
Target		0.066667	0.2		5.666667	25		10.66667	147		5	65

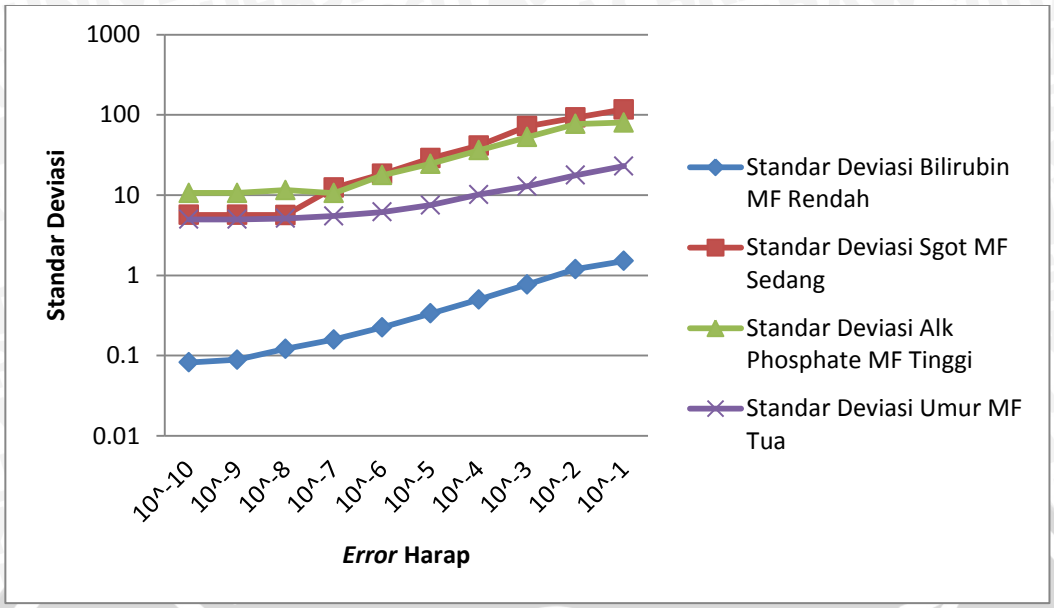
Tabel 5.1 merupakan hasil pengujian berdasarkan parameter *Error* harap pada data dengan nilai bobot dibangkitkan secara acak. Atribut data yang digunakan merupakan atribut data yang dipilih secara acak, dimana atribut data yang diguunakan yaitu pada data Bilirubin *MF* Rendah, Sgot *MF* Sedang, Alk Phosphate *MF* Tinggi, dan Umur *MF* Tinggi.

Penggunaan nilai *Error* harap pada atribut yang beragam dimasudkan untuk mengetahui secara lebih jelas pengaruh *Error* harap yang diberikan, karena dengan data yang beragam maka semakin banyak variasi data yang dihasilkan dan semakin banyak variasi data yang dihasilkan diharapkan akan memudahkan dalam proses analisa.

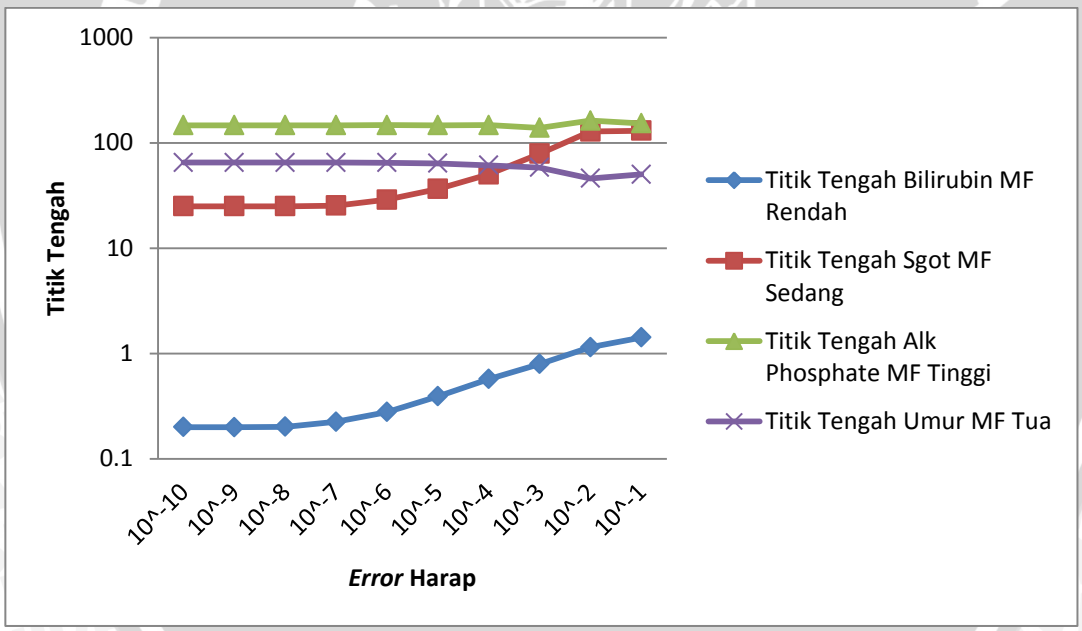


Gambar 5.1 Grafik pengaruh *Error* Harap terhadap *Epoch* pada bobot acak

Gambar 5.1 merupakan grafik pengaruh *Error* harap terhadap *Epoch* pada bobot acak, pada grafik tersebut ditampilkan bahwa semakin besar *Error* harap menghasilkan *epoch* yang semakin besar pula, hal tersebut dapat disebabkan oleh pengaruh *Error* harap sebagai kondisi berhenti pada saat proses pelatihan *BPNN*. Namun perlu digaris bawahi bahwa pada hasil pengaruh *Error* harap terhadap *epoch* pada bobot acak, terdapat nilai dimana *Error* harap yang besar menghasilkan epoch yang lebih kecil, sebagai contoh *Error* harap 10^{-8} pada atribut Alk Phosphate fungsi keanggotaan Tinggi.



Gambar 5.2 Grafik pengaruh *Error* Harap terhadap Standar Deviasi pada bobot acak



Gambar 5.3 Grafik pengaruh *Error* Harap terhadap Titik Tengah pada bobot acak

Gambar 5.2 dan 5.3 merupakan grafik pengaruh *Error* Harap terhadap hasil keluaran yang didapatkan, dari hasil keluaran yang didapatkan dapat diketahui bahwa pada rentang 10^{-8} hingga 10^{-10} menghasilkan nilai yang paling mendekati nlai target. Pada rentang nilai 10^{-1} hingga 10^{-4} merupakan kondisi dimana hasil keluaran belum mendekati nilai target, dan hasil yang didapatkan pada rentang nilai tersebut dapat lebih tinggi ataupun lebih kecil dari target yang

ditentukan berdasarkan bobot awal yang digunakan. Sedangkan rentang nilai 10^{-5} hingga 10^{-7} adalah rentang nilai dimana nilai keluaran sudah semakin mendekati nilai target yang ditentukan.

Berdasarkan pengujian yang telah dilakukan tersebut dapat diketahui bahwa pada *Error* Harap tertentu cenderung mengalami penurunan nilai *epoch*. Hal tersebut dapat disebakan oleh bobot yang digunakan pada proses pengujian. Apabila bobot yang digunakan memudahkan mencapai target yang ditentukan maka *epoch* yang dihasilkan adalah *epoch* dengan nilai yang kecil, sedangkan jika bobot yang didapatkan adalah bobot dengan nilai yang menyulitkan dalam mencapai nilai target maka *epoch* yang dihasilkan adalah *epoch* dengan nilai yang besar.

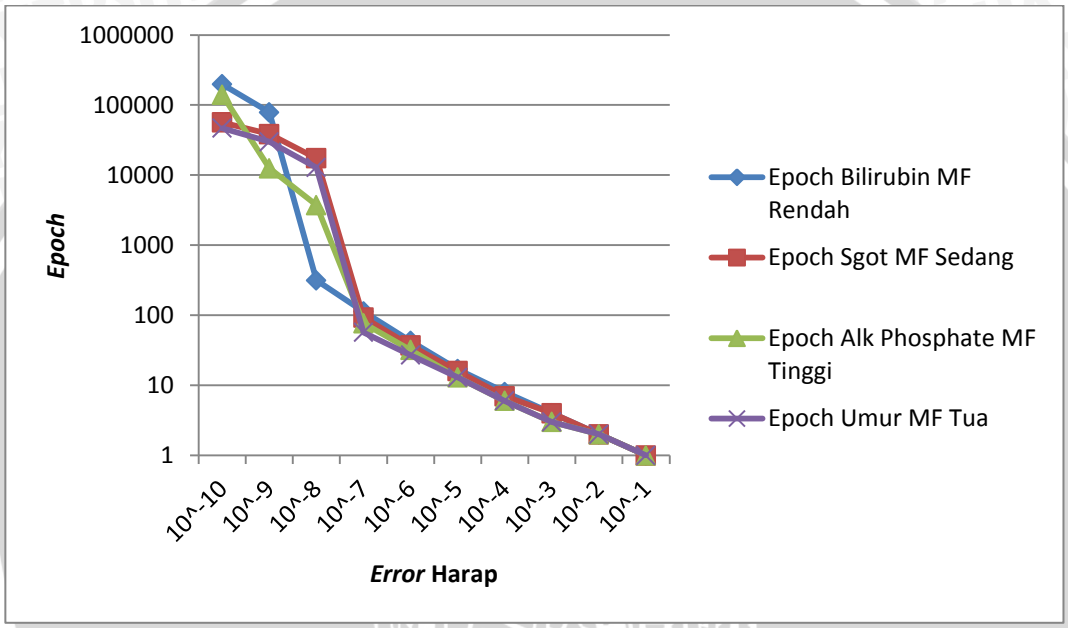
Hasil lain yang didapatkan dari proses pengujian ini adalah pengaruh nilai *Error* Harap terhadap nilai keluaran yang didapatkan. Pada proses pengujian, nilai *Error* Harap 10^{-8} hingga 10^{-10} merupakan nilai *Error* Harap dimana nilai keluaran yang didapatkan merupakan nilai yang paling mendekati nilai target. Namun untuk menetukan nilai *Error* Harap terbaik harus dibandingkan dengan *epoch* yang dihasilkan, sedangkan hasil *epoch* yang didapatkan masih belum dapat dilakukan analisa karena bobot acak yang digunakan menghasilkan nilai *epoch* yang fluktuatif. Sehingga dari permasalahan tersebut dibutuhkan parameter uji dengan nilai yang sama agar hasil yang didapatkan dapat lebih dipertanggung jawabkan dan dilakukan analisa secara ilmiah.

Tabel 5.2 merupakan tabel pengujian *Error* Harap dengan nilai bobot yang sama untuk semua parameter. Penggunaan nilai bobot yang sama dimaksudkan untuk menghindari faktor lain yang mungkin akan mempengaruhi hasil keluaran yang dihasilkan. Sehingga diharapkan dengan penggunaan nilai bobot yang sama, akan menghasilkan nilai keluaran yang dapat lebih dipertanggung jawabkan dan dilakukan analisa secara ilmiah.

Tabel 5.2 Hasil Pengujian Uji Skeanario *Error* Harap pada Bobot Inisialisasi

Pelatihan	Bilirubin *MF* Rendah			Sgot *MF* Sedang			Alk Phosphate *MF* Dewasa			Umur *MF* Tua		
Error Harao	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah
10^-1	1	1.88929	1.753988	1	164.1046	156.3121	1	112.8552	167.0073	1	30.10781	54.22697
10^-2	2	1.199694	1.182224	2	103.6458	106.4265	2	72.81047	157.4099	2	19.79436	57.3879
10^-3	4	0.756187	0.782422	4	64.43236	70.94409	3	56.04469	152.33	3	15.4685	59.24285
10^-4	8	0.499616	0.538746	7	45.15842	52.68388	6	37.27669	147.6708	6	10.66116	61.95022
10^-5	17	0.337185	0.382225	16	28.1359	36.7045	13	25.35411	146.9771	13	7.682564	63.94479
10^-6	43	0.223837	0.275245	37	18.4297	28.57821	32	17.65278	146.977	27	6.139874	64.84315
10^-7	113	0.158502	0.219873	93	12.32727	25.35304	77	13.5619	146.9794	57	5.32588	65.06559
10^-8	315	0.119738	0.199065	17427	5.679021	25.05028	3737	10.62218	146.9908	12813	4.986352	65.02995
10^-9	78696	0.069712	0.199289	38499	5.671397	25.01607	12479	10.62713	146.9989	29972	4.996256	65.00948
10^-10	197508	0.068594	0.199784	56586	5.668737	25.00549	139418	10.65784	147.0005	46641	4.999348	65.00313
Target		0.066667	0.2		5.666667	25		10.66667	147		5	65

Tabel 5.2 merupakan tabel hasil pengujian uji skenario *Error* Harap pada data umur dengan nilai bobot sama pada semua data yang diujikan. Dari data yang didapatkan dapat diketahui bahwa dengan menggunakan bobot dengan nilai yang sama hasil yang didapatkan dapat lebih stabil dan *epoch* yang dihasilkan juga cenderung menunjukan bahwa semakin besar *Error* Harap maka *epoch* yang dihasilkan juga cenderung semakin besar.

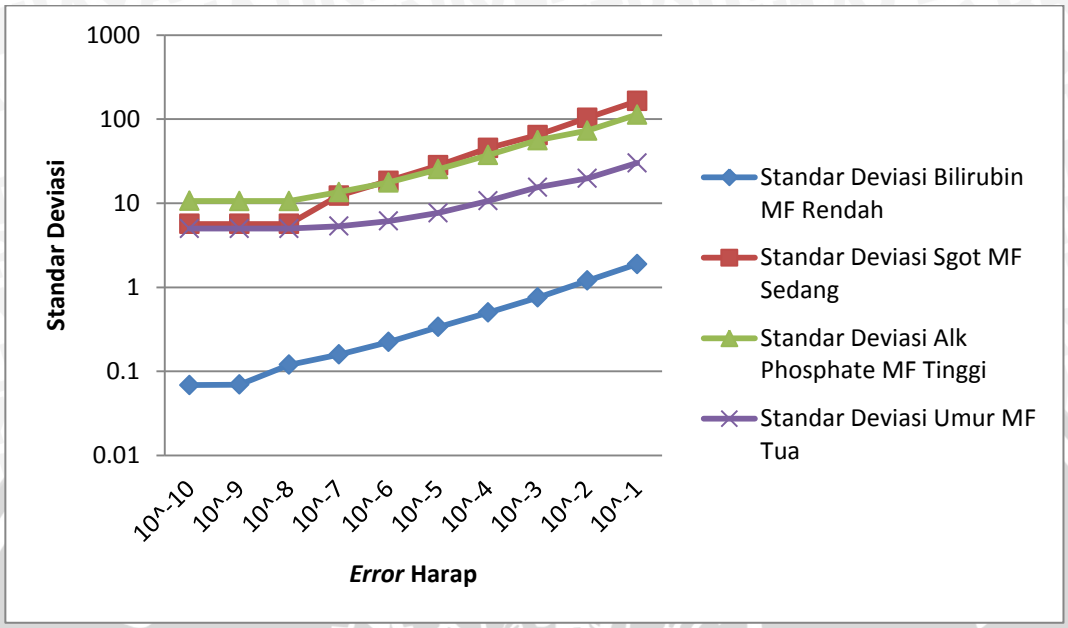


Gambar 5.4 Grafik pengaruh *Error* Harap terhadap *Epoch* pada bobot inisialisasi

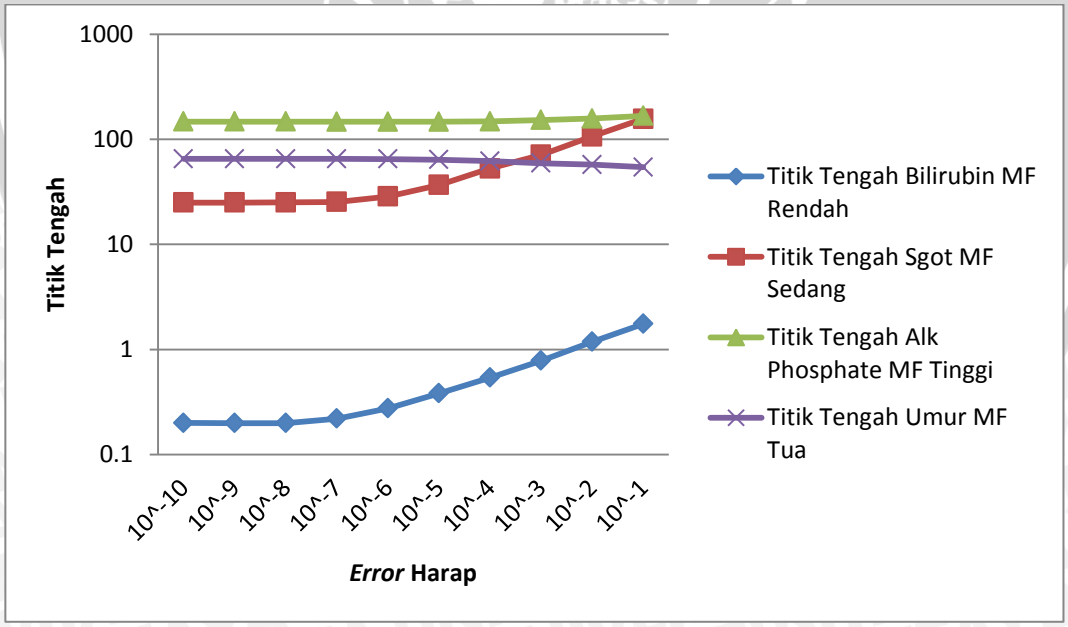
Pada gambar 5.4 merupakan grafik pengaruh *Error* Harap terhadap *epoch* pada bobot terinisialisasi, dari hasil yang didapatkan dapat diketahui bahwa hasil yang didapatkan berbeda dengan hasil yang didapatkan pada pengujian dengan bobot acak. Pada pengujian dengan nilai bobot yang sama menghasilkan nilai *epoch* yang lebih stabil, stabil dalam artian bahwa semakin besar *Error* Harap menghasilkan nilai *epoch* yang semakin besar pula.

Hal tersebut dikarenakan *Error* Harap mempengaruhi kondisi berhenti, dan kondisi berhenti mempengaruhi perbaikan bobot yang dilakukan. Hasil perbaikan bobot sendiri sangat berpengaruh terhadap nilai keluaran yang dihasilkan, karena pada *Error* Harap yang tinggi secara otomatis proses pelatihan telah dilakukan secara berulang kali, dan proses pelatihan yang berulang kali

menghasilkan nilai bobot baru yang semakin baik untuk mencapai hasil yang sesuai dengan nilai target.



Gambar 5.5 Grafik pengaruh *Error* Harap terhadap Standar Deviasi pada bobot inisialisasi



Gambar 5.6 Grafik pengaruh *Error* Harap terhadap Titik Tengah pada bobot inisialisasi

Gambar 5.5 dan 5.6 merupakan grafik pengaruh *Error* Harap terhadap nilai keluaran. Seperti pada pengujian dengan menggunakan bobot acak, pada pengujian dengan bobot yang sama menghasilkan nilai yang paling stabil adalah pada 10^{-8} hingga 10^{-9}. Berdasarkan hasil tersebut, maka selanjutnya dilakukan analisa bahwa, *Error* Harap terbaik dari hasil pengujian yang telah dilakukan adalah pada *Error* Harap 10^{-8}. Karena pada *Error* Harap 10^{-8}, epoch yang dihasilkan dan nilai keluaran yang didapatkan menghasilkan kombinasi yang paling stabil dibandingkan dengan *Error* Harap lain. Selain itu perbedaan nilai keluaran *Error* Harap 10^{-8} dibandingkan *Error* Harap 10^{-9} dan *Error* Harap 10^{-10} tidak meghasilkan perbedaan yang signifikan jika dibandingkan dengan perbedaan nilai *epoch* yang cukup signifikan. Sehingga 10^{-8} merupakan *Error* Harap berdasarkan pengujian yang telah dilakukan.

5.2 Hasil dan Analisis Uji *Learning Rate*

Proses yang dilakukan pada tahapan ini adalah melakukan pengujian berdasarkan nilai *learning rate,* alasan dilakukan pengujian dengan menggunakan nilai *learning rate* adalah untuk mengetahui apakah semakin besar nilai learning rate memiliki pengaruh terhadap nilai keluaran, *MSE* dan *epoch* yang dihasilkan.

Parameter lain yang digunakan pada penelitian ini adalah *Error* Harap, dan jumlah data latih. Nilai *Error* Harap yang digunakan adalah 10^{-8}, dimana nilai *Error* Harap tersebut didapatkan dari proses pelatihan yang dilakukan. Sedangkan jumlah data yang digunakan adalah 64 data. Penggunaan jumlah data sebanyak 64 data karena diasumsikan bahwa semakin banyak data latih maka akan semakin memberikan data yang beragam, dan data yang beragam diasumsikan akan memberikan hasil yang lebih baik dalam proses pelatihan

Proses pengujian pada tahapan ini dibagi dalam satu tahap yaitu proses pengujian dengan menggunakan nilai bobot yang telah dinisialisasi. Penggunaan nilai bobot yang telah diinisialisasi karena pada pengujian sebelumnya penggunaan nilai bobot acak menghasilkan nilai yang fluktuatif akibat pengaruh dari bobot yang digunakan, sehingga dari hasil yang didapatkan berpengaruh terhadap proses analisa yang nantinya dilakukan

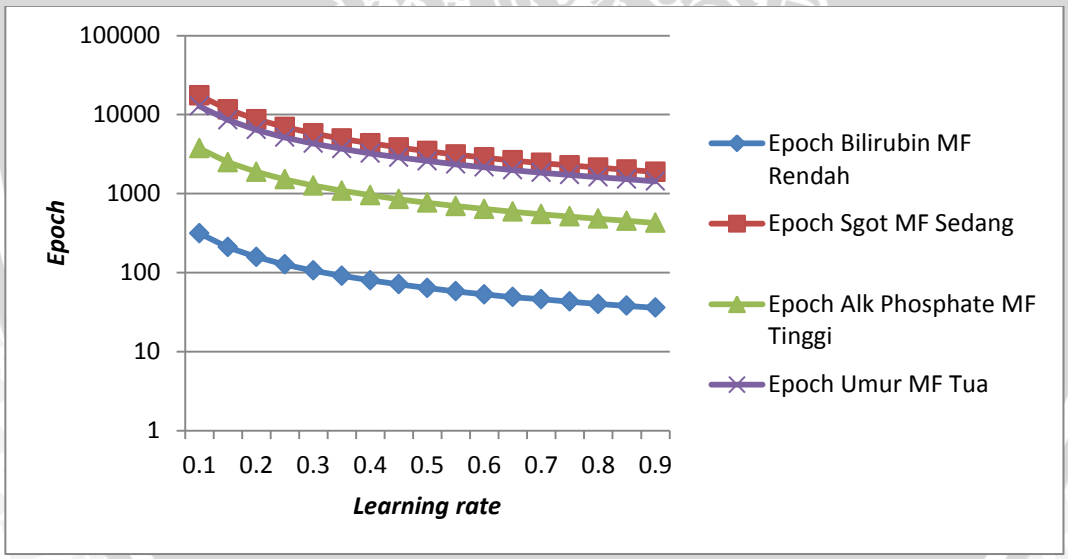
Tabel 5.3 Hasil Pengujian Uji Skeanario *Learning Rate* pada Bobot Inisialisasi

Pelatihan	Bilirubin *MF* Rendah				Sgot *MF* Sedang			
A	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE
0.1	315	0.119737773	0.1990653	7.74016E-09	17427	5.679020648	25.050282	2.06419E-11
0.15	210	0.119743449	0.1990663	7.74342E-09	11624	5.67902528	25.050542	2.08560E-11
0.2	158	0.119660633	0.1990471	7.69615E-09	8713	5.679027681	25.0508	2.10698E-11
0.25	127	0.119534114	0.1990183	7.62439E-09	6960	5.679029469	25.051062	2.12876E-11
0.3	106	0.119495529	0.199011	7.60256E-09	5789	5.679029139	25.051317	2.15008E-11
0.35	91	0.119456785	0.1990036	7.58069E-09	4950	5.679030217	25.051582	2.17229E-11
0.4	80	0.119331068	0.1989756	7.51022E-09	4320	5.679031562	25.051848	2.19480E-11
0.45	71	0.119378987	0.1989886	7.53695E-09	3831	5.679030718	25.052103	2.21646E-11
0.5	64	0.119340004	0.1989809	7.51511E-09	3439	5.679033002	25.052374	2.23955E-11
0.55	58	0.119431138	0.1990038	7.56608E-09	3120	5.679032074	25.052627	2.26124E-11
0.6	53	0.119522596	0.1990267	7.61753E-09	2854	5.679033694	25.052892	2.28409E-11
0.65	49	0.119482983	0.1990181	7.59517E-09	2629	5.67903727	25.053167	2.30786E-11
0.7	46	0.119184049	0.1989485	7.42829E-09	2437	5.679040011	25.053436	2.33128E-11
0.75	43	0.119145135	0.1989401	7.40676E-09	2271	5.679043432	25.053708	2.35505E-11
0.8	40	0.119364395	0.1989918	7.52856E-09	2127	5.67904381	25.053962	2.37744E-11
0.85	38	0.119110145	0.1989327	7.38744E-09	2000	5.679045953	25.054225	2.40069E-11
0.9	36	0.119028767	0.1989142	7.34272E-09	1887	5.679050509	25.054501	2.42517E-11
Target	Rendah	0.066666667	0.2		Sedang	5.666666667	25	

Tabel 5.4 Hasil Pengujian Uji Skeanario *Learning Rate* pada Bobot Inisialisasi

Pelatihan	Alk Phosphate *MF* Tinggi				Umur *MF* Tua			
A	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE
0.1	3737	10.62218	146.9908	4.31258E-11	12813	4.986352	65.02995	7.54581E-10
0.15	2501	10.62215	146.9917	3.64606E-11	8555	4.986384	65.02983	7.48612E-10
0.2	1883	10.62211	146.9926	3.06579E-11	6427	4.986418	65.0297	7.42656E-10
0.25	1512	10.62206	146.9935	2.56748E-11	5150	4.986453	65.02958	7.36748E-10
0.3	1264	10.62202	146.9944	2.14648E-11	4297	4.986486	65.02947	7.31167E-10
0.35	1087	10.62198	146.9952	1.79551E-11	3688	4.986523	65.02934	7.25242E-10
0.4	954	10.62197	146.996	1.51000E-11	3229	4.986556	65.02923	7.19907E-10
0.45	849	10.62196	146.9968	1.28546E-11	2872	4.986592	65.02912	7.14312E-10
0.5	766	10.62197	146.9975	1.11320E-11	2586	4.98663	65.029	7.08689E-10
0.55	697	10.62199	146.9982	9.89759E-12	2351	4.98666	65.0289	7.03412E-10
0.6	640	10.62202	146.9989	9.08634E-12	2155	4.9867	65.0288	6.98105E-10
0.65	591	10.62205	146.9995	8.65200E-12	1989	4.98673	65.0287	6.92809E-10
0.7	549	10.6221	147.0001	8.54294E-12	1847	4.98677	65.0285	6.87286E-10
0.75	513	10.62215	147.0007	8.71291E-12	1723	4.9868	65.0284	6.82385E-10
0.8	481	10.62221	147.0012	9.12138E-12	1615	4.98683	65.0283	6.77125E-10
0.85	452	10.62226	147.0018	9.73364E-12	1519	4.98686	65.0282	6.72425E-10
0.9	427	10.62233	147.0022	1.05054E-11	1434	4.98689	65.0281	6.67487E-10
Target	Tinggi	10.66667	147		Tua	5	65	

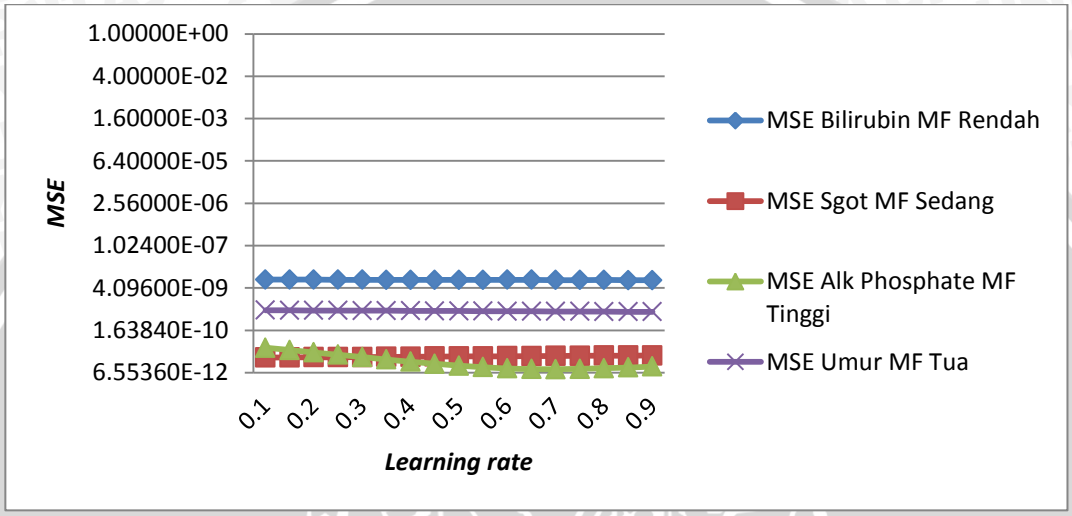
Tabel 5.3 dan Tabel 5.4 merupakan hasil pengujian nilai *learning rate*. Berdasarkan pengujian tersebut didapatkan hasil bahwa semakin besar nilai *learning rate* berpengaruh terhadap nilai *epoch* yang semakin kecil. Sedangkan untuk nilai standar deviasi dan nilai titik tengah yang dihasilkan didapatkan nilai yang bersifat fluktuatif. Salah satu faktor yang mempengaruhi terjadinya nilai keluaran yang naik turun adalah faktor nilai *learning rate* sebagai nilai yang digunakan dalam perbaikan nilai bobot. Pada *Backpropagation Neural Network* bobot memiliki pengaruh terhadap nilai *Y* yang dihasilkan dan nilai bobot yang digunakan dipengaruhi oleh nilai *learning rate*. Perbedaan dari hasil perbaikan bobot itulah yang menyebabkan nilai keluaran dari *Backpropagation Neural Network* menghasilkan nilai yang naik dan turun pada nilai *learning rate* tertentu. Namun perlu digaris bawahi bahwa perbedaan nilai yang dihasilkan tidak menghasilkan perbedaan nilai yang signifikan karena pengaruh utama dari *learning rate* adalah pada nilai *epoch* yang dihasilkan.



Gambar 5.7 Grafik pengaruh *Learning Rate* terhadap *Epoch*

Gambar 5.7 merupakan grafik pengaruh *learning rate* terhadap *epoch*. *Epoch* merupakan salah satu parameter yang digunakan untuk mengetahui sejauh mana efektifitas yang didapatkan dari pengujian parameter tertentu. Seperti yang telah dijelaskan sebelumnya bahwa pada pengujian pengaruh *learning rate* terhadap nilai *epoch* menghasilkan nilai bahwa semakin besar nilai *learning rate* yang digunakan berbanding lurus dengan semakin sedikit *epoch* yang dihasilkan.

Berdasarkan proses pengujian tersebut dapat dilihat bahwa nilai *epoch* terbaik didapatkan pada nilai *learning rate* tertinggi, yaitu nilai *learning rate* 0.9. Nilai *learning rate* 0.9 menghasilkan nilai epoch paling rendah dikarenakan pada penelitian ini nilai yang digunakan merupakan nilai hasil hasil normalisasi sehingga dengan semakin besarnya nilai *learning rate* yang digunakan maka akan semakin mempermudah proses dalam mendapatkan *Y* sesuai dengan nilai target.



Gambar 5.8 Grafik pengaruh *Learning Rate* terhadap *MSE*

Gambar 5.8 merupakan hasil yang didapatkan berdasarkan proses pengujian pengaruh *Learning Rate* terhadap nilai *MSE*. Berdasarkan pengujian yang dilakukan didapatkan hasil bahwa nilai *MSE* yang didapatkan bersifat fluktuatif, dan hasil tersebut dapat secara jelas dilihat pada tabel 5.3 dan tabel 5.4. Pada gambar 5.8 hanya hasil dari Alk Phosphate yang dapat dilihat secara jelas perubahannya, karena pada hasil Alk Phosphate terjadi perubahan nilai *MSE* yang cukup signifikan pada learning rate tertentu. Sedangkan pada data uji lainnya perubahan yang terjadi pada nilai yang relatif kecil sehingga pada grafik yang terlihat hanya seperti garis lurus.

Berdasarkan pengujian tersebut dapat dilihat bahwa nilai *learning rate* memiliki pengaruh terhadap *MSE* yang dihasilkan, karena pada proses pembaharuan bobot juga dilakukan perhitungan berdasarkan nilai *learning rate* yang digunakan. Bobot baru yang digunakan merupakan faktor yang mempengaruhi nilai keluaran yang dihasilkan, dan nilai keluaran yang dihasilkan juga memiliki pengaruh terhadap nilai *MSE* yang didapatkan. Namun berdasarkan

pengujian yang dilakukan belum dapat diketahui nilai *learning rate* terbaik berdasarkan pengujian nilai *MSE*, karena kecenderungan nilai *MSE terkecil* didapatkan pada beberapa sebaran data yang berbeda sehingga perlu dilakukan analisa berdasarkan semua parameter pengujian yang dilakukan untuk dapat mengetahui nilai *learning rate* terbaik.

Proses yang dilakukan dalam menentukan nilai *learning rate* terbaik didasarkan pada nilai *epoch* yang dihasilkan dan nilai hasil keluaran yang didapatkan serta berdasarkan nilai *MSE* yang didapatkan. Berdasarkan nilai epoch nilai keluaran, dan *MSE* yang didapatkan diketahui bahwa nilai *learning rate* terbaik adalah pada nilai *learning rate* 0.9, karena pada *learning rate* tersebut menghasilkan nilai *epoch* terkecil. Selain itu nilai keluaran yang didapatkan merupakan nilai yang paling relevan jika dibandingkan dengan *epoch* yang dihasilkan. Sedangkan hasil pengujian berdasarkan parameter *MSE* juga didapatkan hasil bahwa nilai *MSE* yang didapatkan pada masing-masing *learning rate* tidak menghasilkan nilai yang memiliki perbedaan nilai cukup jauh, sehingga perbedaan nilai *MSE* yang didapatkan tidak memberikan perubahan yang signifikan jika dibandingkan dengan pengaruhnya terhadap nilai epoch yang dihasilkan.

5.3 Hasil dan Analisis Uji Kombinasi Data Latih

Pengujian yang dilakukan pada tahapan ini adalah pengujian berdasarkan jumlah data latih yang digunakan. Pada pengujian ini, jumlah data latih yang digunakan adalah 64, 56, 48, 40 dan 32. Penggunaan jumlah data pada rentang nilai tersebut dimaksudkan untuk mengetahui perubahan nilai yang terjadai secara bertahap, sehingga diharapkan memudahkan proses analisa yang dilakukan. Data uji yang digunakan pada pengujian ini adalah berjumlah 16 dan jumlah yang sama untuk semua jumlah data latih, hal ini dimaksudkan untuk mengetahui pengaruh jumlah data latih yang digunakan terhadap data uji yang digunakan.

Parameter lain yang digunakan pada jumlah data adalah nilai *Error* Harap dan nilai *learning rate.* Nilai *Error* Harap yang digunakan pada penelitian ini adalah 10^{-8} dan nilai *learning rate* yang digunakan adalah 0.9. dimana nilai tersebut didapatkan dari proses pengujian yang telah dilakukan sebelumnya.

Tabel 5.5 Hasil Pengujian Uji Skenario Kombinasi Data Latih pada Bobot Inisialisasi

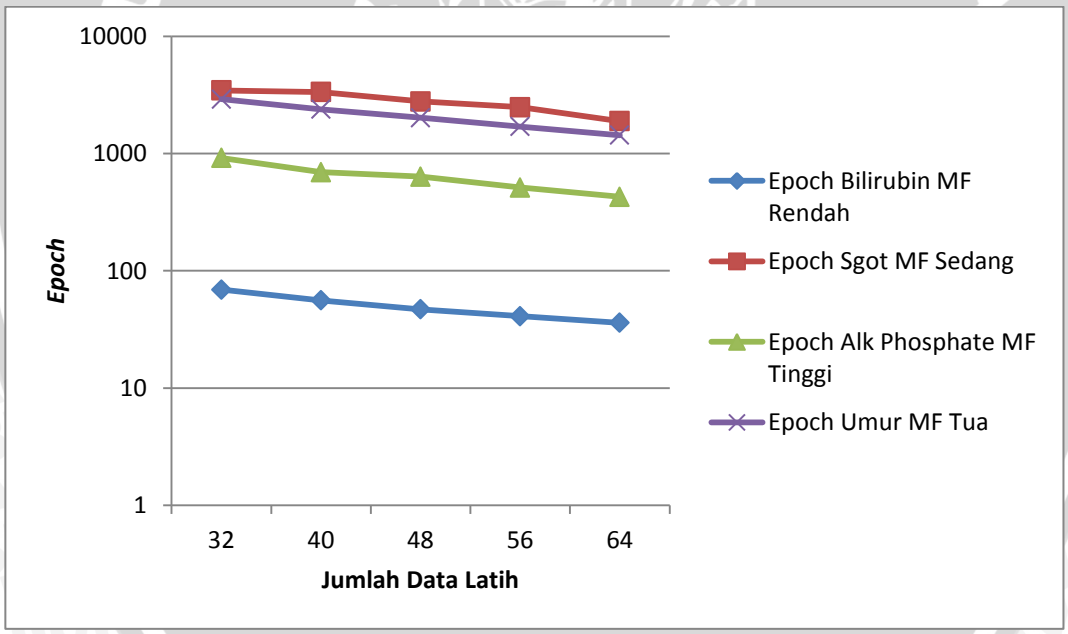
Pelatihan	Bilirubin *MF* Rendah				Sgot *MF* Sedang			
Data Latih	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE
64	36	0.119028767	0.1989142	7.34272E-09	1887	5.679050509	25.054501	2.42517E-11
56	41	0.119050967	0.1986259	7.36662E-09	2484	5.672731414	25.024354	4.83290E-12
48	47	0.119443676	0.1983474	7.60145E-09	2783	5.669731627	25.009441	7.26511E-13
40	56	0.119670907	0.1985205	7.72293E-09	3346	5.669876786	25.009786	7.80768E-13
32	69	0.120045095	0.1985077	7.94166E-09	3448	5.674812889	25.032339	8.52749E-12
Target	Rendah	0.066666667	0.2		Sedang	5.666666667	25	

Tabel 5.6 Hasil Pengujian Uji Skenario Kombinasi Data Latih pada Bobot Inisialisasi

Pelatihan	Alk Phosphate *MF* Tinggi				Umur *MF* Tua			
Data Latih	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE	Epoch	Nilai Standar Deviasi	Nilai Titik Tengah	MSE
64	427	10.62232713	147.00225	1.05054E-11	1434	4.986893609	65.028122	6.67487E-10
56	513	10.60889152	146.99037	5.20344E-11	1698	4.986624751	65.028688	6.94548E-10
48	635	10.6037168	146.98344	1.28591E-10	2016	4.985747674	65.031204	8.19166E-10
40	694	10.61519859	146.99605	1.77265E-11	2382	4.985459719	65.031953	8.58419E-10
32	918	10.61630508	146.99733	1.37904E-11	2903	4.985736997	65.03097	8.07760E-10
Target	Tinggi	10.66666667	147		Tua	5	65	

Tabel 5.5 dan 5.6 merupakan hasil pengujian kombinasi data latih. Pengujian yang dilakukan pada tahapan ini adalah pengujian dengan menggunakan bobot inisialisasi, penggunaan nilai bobot inisialisasi dimaksudkan untuk memudahkan proses analisa yang dilakukan. Karena berdasarkan pengujian yang telah dilakukan sebelumnya penggunaan bobot acak menghasilkan data dengan nilai yang lebih fluktuatif.

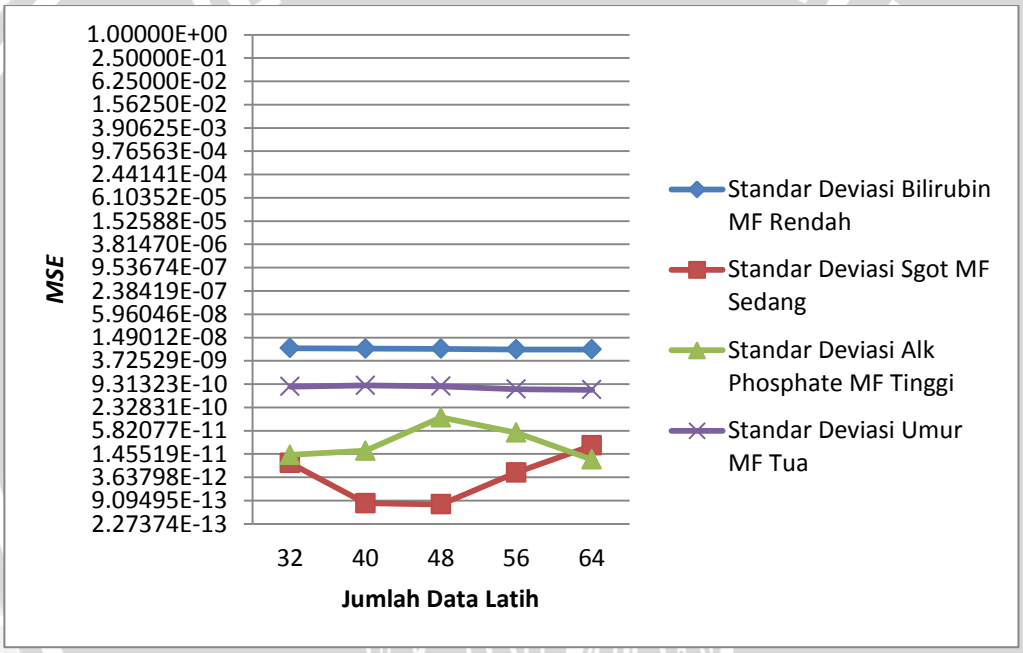
Berdasarkan pengujian yang dilakukan didapatkan hasil bahwa secara keseluruhan nilai standar deviasi dan nilai titik tengah yang didapatkan berada pada rentang nilai yang mendekati target untuk semua atribut data yang diujikan. Hal tersebut karena pengaruh dari penggunaan nilai *Error* Harap dan *learning rate* pada nilai yang sama, sehingga pada hasil yang didapatkan perbedaan nilai yang dihasilkan tidak menghasilkan perbedaan nilai yang terlalu signifikan pada jumlah data yang diujikan.



Gambar 5.9 Grafik pengaruh Data Latih terhadap *Epoch*

Gambar 5.9 merupakan grafik pengaruh jumlah data latih terhadap *epoch* Pada grafik tersebut didapatkan hasil bahwa jumlah data latih 64 menghasilkan nilai jumlah epoch terkecil, namun perlu ditekankan bahwa jumlah data latih bukan merupakan faktor utama yang menetukan jumlah epoch yang didapatkan. Sebagai contoh adalah pada data Sgot fungsi keanggotaan sedang. Pada atribut tersebut setiap terjadi penurunan jumlah data latih berdampak pada nilai epoch

yang dihasilkan namun perubahan nilai epoch yang dihasilkan tidak menghasilkan perubahan nilai epoch yang stabil. Pada pengujian tersebut, lonjakan nilai epoch yang dihasilkan sering bersifat naik turun, sebagai contoh selisih nilai epoch yang dihasilkan epoch pada jumlah data latih 64 dan 56 akan menghasilkan selisih nilai epoch yang berbeda dengan selisih epoch pada jumlah data 48 dan 40. Sehingga dapat diketahui bahwa epoch sangat berpengaruh terhadap ragam data yang didapatkan bukan jumlah data yang ada. Karena jika kombinasi ragam data yang digunakan memudahkan dalam mendapatkan nilai target yang diharapkan maka akan berpengaruh terhadap nilai *epoch* yang lebih sedikit dibandingkan jumlah data latih lain.



Gambar 5.10 Grafik pengaruh Data Latih terhadap *MSE*

Pada Gambar 5.10 merupakan hasil pengaruh data latih terhadap nilai *MSE* yang dihasilkan, dari pengujian ini didapatkan nilai bahwa jumlah data latih memiliki pengaruh terhadap nilai *MSE*. Proses pengujian dengan melakukan perhitungan nilai *MSE* digunakan untuk mengetahui secara lebih jelas nilai *error* rata-rata yang didapatkan, sehingga dari hasil tersebut dapat diketahui secara lebih jelas pengaruh dari *learning rate* yang digunakan.

Pada gambar diatas dapat dilihat bahwa pada atribut Alk Phosphate merupakan salah satu data yang memiliki nilai *MSE* yang fluktuatif. Selain itu

pada atribut Sgot juga menghasilkan nilai yang bersifat fluaktuatif karena terjadi perubahan nilai *MSE* yang signifikan dari proses pengujian yang dilakukan. Hal tersebut disebabkan karena pengaruh data yang digunakan dalam proses pengujian. Jika keragaman data yang digunakan memudahkan dalam mendapatkan nilai keluaran terbaik, maka data tersebut akan berdampak pada nilai *MSE* yang dihasilkan, karena ragam data yang terbaik akan menghasilkan nilai yang paling mendekati nilai target. Sehingga jika nilai keluaran yang didapatkan menghasilkan nilai terbaik, selisih nilai target dan nilai keluaran akan menghasilkan selisih nilai yang kecil dan selisih tersebut akan berdampak pada nilai *MSE* yang dihasilkan. Dari proses pengujian tersebut dapat dilihat bahwa jumlah data latih memiliki pengaruh terhadap *MSE* yang dihasilkan namun keragaman data yang digunakan merupakan faktor yang paling menetukan *MSE* yang dihasilkan.

Proses yang dilakukan selanjutnya adalah melakukan analisa untuk menentukan nilai yang digunakan sebagai kombinasi data latih terbaik dari penelitian ini. Berdasarkan pengujian yang dilakukan untuk dapat menentukan kombinasi data latih terbaik adalah dengan membandingakan nilai keluaran yang didapatkan dengan nilai epoch yang dihasilkan serta membandingkan dengan nilai *MSE* dari proses pengujian.

Pada proses pengujian ini, kombinasi data terbaik yang didapatkan adalah pada jumlah data latih 64 karena pada jumlah data latih 64 secara keseluruhan menghasilkan perbandingan nilai *epoch* dan nilai keluaran yang lebih baik dibandingkan dengan hasil yang didapatkan pada jumlah data lain. Faktor yang mempengaruhi suatu data latih menghasilkan nilai *epoch* dan nilai keluaran terbaik adalah pengaruh dari sebaran data yang digunakan, jika sebaran data yang digunakan merupakan data yang mencakup kebutuhan dari proses yang nantinya dilakukan maka data tersebut akan memberikan dampak yang signifikan dari proses yang nantinya dilakukan. Dampak yang diberikan dapat mencakup nilai *epoch* yang lebih sedikit ataupun nilai keluaran yang mendekati nilai target yang ditentukan. Selain itu nilai *MSE* yang didapatkan dari jumlah data latih 64 juga menghasilkan nilai paling stabil jika dibandingkan dengan jumlah data latih lain,

karena pada jumlah data latih 64, *MSE* yang dihasilkan memiliki pengaruh yang signifikan terhadap nilai epoch yang didapatkan.

Pada *Backpropagation Neural Network* data latih yang digunakan merupakan faktor yang sangat penting karena pada *Backpropagation Neural Network* data latih memiliki fungsi untuk menghasilkan bobot yang paling dibutuhkan pada saat dilakukan proses pengujian. Berdasarkan pengujian yang telah dilakukan maka jumlah data latih terbaik berdasarkan pengujian yang dilakukan adalah pada jumlah data latih 64.

5.4 Hasil Fungsi Keanggotaan

Proses yang dilakukan pada tahapan ini adalah menampilkan hasil akhir batas fungsi keanggotaan pada semua atribut berdasarkan nilai terbaik dari parameter yang sebelumnya telah dilakukan proses pengujian. Berdasarkan proses pengujian yang sebelumnya dilakukan, parameter terbaik dari penelitian ini adalah jumlah data latih 64, nilai *MSE* 10^{-8}, dan nilai *learning rate* 0.9.

Tabel 5.7 Hasil Akhir Fungsi Keanggotaan

Atribut Data	Fungsi Keanggotaan	Standar Deviasi	Titik Tengah
Umur	Anak	2.638909	7.969989
	Remaja	1.427407	15.98672
	Dewasa	6.6719	37.99181
	Tua	5.130759	65.01711
Bilirubin	Rendah	0.12076	0.200666
	Sedang	0.215125	0.899986
	Tinggi	0.566662	3.000039
Alk Phosphate	Rendah	6.467442	29.95637
	Sedang	18.62586	99.99941
	Tinggi	10.66873	146.998
Sgot	Rendah	5.64671	6.615444
	Sedang	5.735662	25.05476

Atribut Data	Fungsi Keanggotaan	Standar Deviasi	Titik Tengah
	Tinggi	5.330739	49.96802
Albumine	Rendah	0.666148	2.000026
	Sedang	0.204357	4.000221
	Tinggi	0.254385	5.944735
Protime	Rendah	1.013795	8.03371
	Sedang	0.807681	12.02012
	Tinggi	12.16799	49.99866

BAB VI
PENUTUP

6.1 Kesimpulan

Kesimpulan yang dapat diambil dari skripsi ini adalah:

1. *Backpropagation Neural Network* merupakan metode yang mampu digunakan secara baik dalam pembangkitan otomatis fungsi keanggotaan *fuzzy* pada penderita penyakit hepatitis berdasarkan tahapan proses yang dilalui, dimana tahapan proses yang dilalui antara lain proses pembacaan data, penentuan nilai bobot target, normalisasi data, pelatihan *BPNN*, pengujian *BPNN* berdasarkan skenario uji coba, dan proses denormalisasi yang diakhiri dengan pembentukan kurva berdasarkan fungsi keanggotaan Gaussian.
2. Hasil Fungsi Keanggotaan yang didapatkan dari penerapan metode *Backpropagation Neural Network* menghasilkan batas fungsi keanggotaan yang baik, karena *Backpropagation Neural Network* mampu menghasilkan fungsi keanggotaan seperti hasil fungsi keanggotaan yang dicapai berdasarkan penentuan oleh Pakar.
3. *Error* Harap terbaik dari hasil penelitian adalah pada nilai *Error* Harap 10^{-8}, karena pada nilai *Error* Harap tersebut menghasilkan perbandingan nilai *epoch* dan nilai keluaran yang lebih stabil dibandingkan dengan nilai *Error harap* lain. Nilai *Learning rate* terbaik pada penelitian ini adalah pada nilai 0.9, karena pada *Learning rate* tersebut menghasilkan jumlah epoch yang paling relevan terhadap nilai keluaran yang didapatkan. Sedangkan jumlah data latih terbaik adalah pada jumlah data latih 64 karena pada jumlah data tersebut memberikan keragaman data yang lebih baik sehingga berpengaruh terhadap *epoch* total yang lebih kecil dan menghasilkan nilai *MSE* yang paling relevan terhadap total epoch serta nilai keluaran yang dihasilkan.

6.2 Saran

Saran untuk penelitian berikutnya dari skripsi ini adalah :

1. Bobot awal yang digunakan pada penelitian dapat dicari dengan menggunakan metode tertentu, seperti Nguyen Widrow untuk menghasilkan bobot awal yang lebih seusai dengan nilai keluaran yang dihasilkan.

2. Fungsi keanggotaan lain yang dapat digunakan pada penelitian selanjutnya antara lain fungsi keanggotaan Bell, Trapesium, ataupun Sigmoid. Dari penerapan fungsi keanggotaaan tersebut diharapkan dapat memberikan informasi tentang fungsi keanggotaan yang paling relevan untuk digunakan pada data penderita penyakit hepatitis.
3. Batas fungsi keanggotaan yang didapatkan dari penelitian ini dapat diterapkan pada penelitian selanjutnya dengan menggunakan sistem inferensi *fuzzy* tertentu untuk mengetahui hasil keluaran sistem inferensi *fuzzy* tentang kemampuan bertahan hidup penderita hepatitis.
4. Pada penelitian selanjutnya dapat dilakukan pengembangan dengan menerapkan arsitektur jaringan secara keseluruhan pada semua atribut data dan fungsi keanggotaan yang digunakan.
5. Pada penelitian selanjutnya dapat dilakukan proses penggabungan metode backpropagation dengan metode tertentu, seperti LVQ untuk mengetahui hasil dari kombinasi metode *Backpropagation Neural Network* dengan metode tertentu.

DAFTAR PUSTAKA

[AGU-10] Agustina, I.D. 2010. Penerapan Metode *Extreme Learning Machine* untuk Peramalan Permintaan, Skripsi, Institut Teknologi Sepuluh November,Surabaya.

[BUD-13] Budianita, L., dan Widodo P. 2013," Penerapan *Learning Vector Quantization (LVQ)* untuk Klasifikasi Status Gizi Anak", IJCCS, Vol.7, No.2., ISSN: 1978-1520.

[FAU-94] Fausett, L. 1994,"*Fundamentals of Neural Network: Architectures, Algorithms, and Applications*", Prenctice-Hall, Inc., New Jersey, USA.

[IND-12] Indrabayu dkk. 2012," Prediksi Curah Hujan Dengan Jaringan Syaraf Tiruan", Prosiding 2012, Volume 6, ISBN:978-979-127255-0-6.

[JAN-97] Jang, J.S.R., Sun, C.T., Mizutani,E. 1997, "*Neuro-Fuzzy and Soft Computing*", Prentice-Hall International, New Jersey.

[JEF-08] Jeff Heaton.2008. "*Introduction to Neural Networks for Java, Second Edition"*. http://www.heatonresearch.com/node/707 [17 Juni 2014].

[KRU-14] Krucik, G. 2014. Bilirubin *Blood Test*. http : // www. Healthline .com / health/bilirubin-blood#AbnormalResults [7 April 2014].

[MED-14a] MedlinePlus. 2014. Alp *Blood Test*. http: // www . nlm . nih . gov / medlineplus / ency / article/003470.htm [7 April 2014].

[MED-14b] MedlinePlus. 2014. Albumin *Blood* Serum. http : // www . nlm . nih . gov/medlineplus/ency/article/003480.htm [7 April 2014].

[MED-14c] MedlinePlus. 2014. Prothrombin Time. http : // www . nlm . nih . gov / medlineplus/ency/article/003652.htm [7 April 2014].

[MED-14d] MedicineNet.2014.*Liver Blood Test.* http : // www . medicinenet . com / liver_blood_tests/page2.htm [7 April 2014].

[PUT-14] Putri,Ratna.2014," Implementasi Algoritma Particle Swarm Optimization Untuk Optimasi Fungsi Keanggotaan Pada Kondisi

Penderita Penyakit Hepatitis", Skripsi Program Studi Ilmu Komputer, Universitas Brawijaya, Malang.

[ROJ-96] Rojas, R. 1996. "*Neural Network*", Springer-Verlag, Berlin.

[ROS-04] Ross, T.J. 2004. "*Fuzzy Logic With Engineering Aplication*", Second edition, John Weley and Son.

[SED-06] Sedgewick, R., Wayne, K. 2006, "*Introduction to Programming in Java*", Addison-Wesley, New York.

[UCI-88] UCI machine learning repository.1988.Hepatitis Data *Set.* http :// www.ics.uci.edu/~mlearn/ MLRepository.html [21 Februari 2014].

[WOR-13] World Health Organization.2013.*What is* Hepatitis?. http : // www.who.int / feature /qa/76/en/.[21 Februari 2014].

[YAN-05] Yang, Bose.2005."*Generating Fuzzy Membership Function with Self Organizing Feature Maps*",Science Direct, Pattern Recognition Letters 27 (2006) 356–365.

[YUN-12] Yunizar,Zara.2012,"Pembangkitan Fungsi Keanggotaan Fuzzy Otomatis Menggunakan Neural Network",Thesis Program Studi Teknik Informatika, Program Pasca Sarjana Universitas Sumatra Utara, Medan.

[ZAD-65] Zadeh, L.A. 1965, *Fuzzy Sets*, Information Control, 8:338–353.