BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Salah satu penggunaan metode *backpropagation* adalah dalam analisis medikal, seperti pengidentifikasian suatu penyakit yang memiliki gejala yang bias antara jenis yang satu dengan yang lainya. Hal ini telah dibuktikan pada jurnal "Aplikasi jaringan syaraf tiruan untuk memprediksi penyakit saluran pernafasan dengan metode *backpropagation"* karya Novi Indah Pradasari, F.Trias Pontia W, dan Dedi Triyanto[2]. Pada jurnal ini, metode *backpropagation* digunakan untuk memprediksi penyakit saluran pernafasan berdasarkan gejala penyakit yang disusun menjadi 12 buah variabel masukan. Target keluaran sistem berupa jenis penyakit saluran pernafasan yang teridentifikasi oleh sistem, sistem memiliki batasan penyakit yang dapat diidentifikasi sebanyak lima buah penyakit yaitu penyakit *Asma, ISPA, Pneumonia, Bronkhitis, Sinusitis,* dan *Tuberkulosis.* Penelitian ini mendapatkan kesimpulan bahwa metode *backpropagation* dapat digunakan untuk memprediksi penyakit saluran pernafasan dengan keakuratan 91,66% yang diperoleh dari pelatihan 96 data dan pengujian 24 data menggunakan dua buah *hidden* layer, target *error* sebesar 0,0001, dan *learning rate* sebesar 0,1 [2].

Penyakit mata menular memiliki gejala yang hampir mirip antara jenis yang satu dan yang lainya. Gejala-gejala yang ditimbulkan penyakit mata menular dapat berjumlah tiga sampai lima gejala setiap jenis penyakitnya, jumlah ini merupakan jumlah yang sangat sedikit. Suatu penyakit akan sangat mudah diidentifikasi apabila memiliki gejala yang sangat berbeda, akan tetapi gejala yang ditimbulkan penyakit mata menular sangat bias antara jenis penyakit satu dan lainnya. Penyakit mata menular menjadi sulit untuk diidentifikasi karena minimnya gejala yang ditimbulkan dan banyaknya gejala yang mirip. Dari masalah yang telah dijelaskan, maka perlunya dilakukan analisis performansi metode *backpropagation* untuk mengetahui akurasi

dan parameter yang optimal dari metode *backpropagation* dalam mengidentifikasi penyakit mata menular.

3.2 Analisis Data Masukan

Data yang digunakan dalam mengidentifikasi penyakit mata menular pada sistem adalah data gejala dari beberapa penyakit mata menular seperti terlihat dalam tabel 2.1 penyakit mata menular dan gejalanya pada bab 2. Dari tabel 2.1 penyakit mata menular dan gejalanya, gejala-gejala dari penyakit mata tersebut disusun menjadi variabel X_1 hingga X_{19} yang menjadi data masukan pada sistem. Masing-masing variabel memiliki rentang nilai dari 0 sampai 1 diurutkan dari tidak terjadi gejala hingga terjadi gejala yang parah. Untuk nilai dari seluruh variabel dapat dilihat pada tabel 3.1 berikut ini.

Tabel 3.1 Variabel input gejala penyakit mata menular

Gejala-gejala	Variabel	Nilai
Mata Merah (X_1)	Tidak	0
	Ya	1
Frekuensi Waktu Mata Merah (X_2)	Tidak	0
	Ya	1
Mata Terasa Perih (X_3)	Tidak	0
	Ya	1
Mata Terasa Gatal (X_4)	Tidak	0
	Ya	1
Mata Berair (X_5)	Tidak	0
	Ya	1
Mata Mengeluarkan Kotoran (X_6)	Tidak	0
	Ya	1
Mengalami Gangguan Penglihatan (X_7)	Tidak	0
	Ya	1
Kelopak Mata Membengkak (X_8)	Tidak	0
	Ya	1
Bola Mata Menonjol (X_9)	Tidak	0

	Ya	1
Bola Mata Membengkak (X_{10})	Tidak	0
	Ya	1
Kelenjar Getah Bening Membengkak (X_{11})	Tidak	0
	Ya	1
Terdapat Bintik Putih Pada Kornea (X_{12})	Tidak	0
	Ya	1
Kornea Tampak Keruh(X_{13})	Tidak	0
	Ya	1
Terdapat Luka Pada Kelopak Mata (X_{14})	Tidak	0
	Ya	1
Sakit Kepala (X_{15})	Tidak	0
	Ya	1
Demam (X_{16})	Tidak	0
	Ya	1
Mata Terasa Panas (X_{17})	Tidak	0
	Ya	1
Bulu Mata Rontok (X_{18})	Tidak	0
	Ya	1
Sensitif Terhadap Cahaya (X_{19})	Tidak	0
	Ya	1

Data masukan yang digunakan untuk mengidentifikasi penyakit mata menular, memiliki batasan minimum gejala yang harus dipilih pada pilihan yang telah disediakan sistem. Minimum gejala yang bisa digunakan untuk pengidentifikasian penyakit mata menular adalah sebanyak empat buah gejala. Jika data masukan yang dipilih kurang dari empat buah gejala, maka sistem tidak dapat melakukan identifikasi.

3.3 Analisis Metode

Metode *backpropagation* adalah salah satu metode pembelajaran dari jaringan syaraf tiruan. Metode ini memiliki tingkat *error* yang sangat kecil, karena metode ini dapat memperkecil tingkat *error* dengan cara menyesuaikan perbedaan *output* dan

target yang diinginkan. Metode ini memiliki arsitektur jaringan *multilayer network* yang merupakan arsitektur jaringan dengan lapisan lebih dari dua lapisan. Metode *backpropagation* merupakan metode yang membutuhkan pembelajaran sebelum dapat mengidentifikasi suatu kasus, maka metode ini memiliki tahap pelatihan dan tahap pengujian.

a. Arsitektur Jaringan Metode *Backpropagation*

Arsitektur metode *backpropagation* terdiri dari tiga *layer* dalam proses pembelajarannya, yaitu *input layer, hidden layer,* dan *output layer.*

1) *Input layer*

Input layer memiliki jumlah *neuron* yang dipengaruhi oleh jumlah masukan pada suatu kasus, *input layer* (X_i) pada kasus ini memiliki 19 *neuron*, karena gejala penyakit mata secara keseluruhan berjumlah 19 gejala. *Input layer* berfungsi menerima nilai pola berupa sinyal *input* yang didapatkan dari terjadi atau tidak terjadinya gejala pada sebuah pola, sinyal *input* akan bernilai 1 bila gejala terjadi dan bernilai 0 bila gejala tidak terjadi. Pada *input layer* tidak terjadi proses komputasi, namun terjadi pengiriman sinyal input x ke *hidden layer.*

2) *Hidden layer*

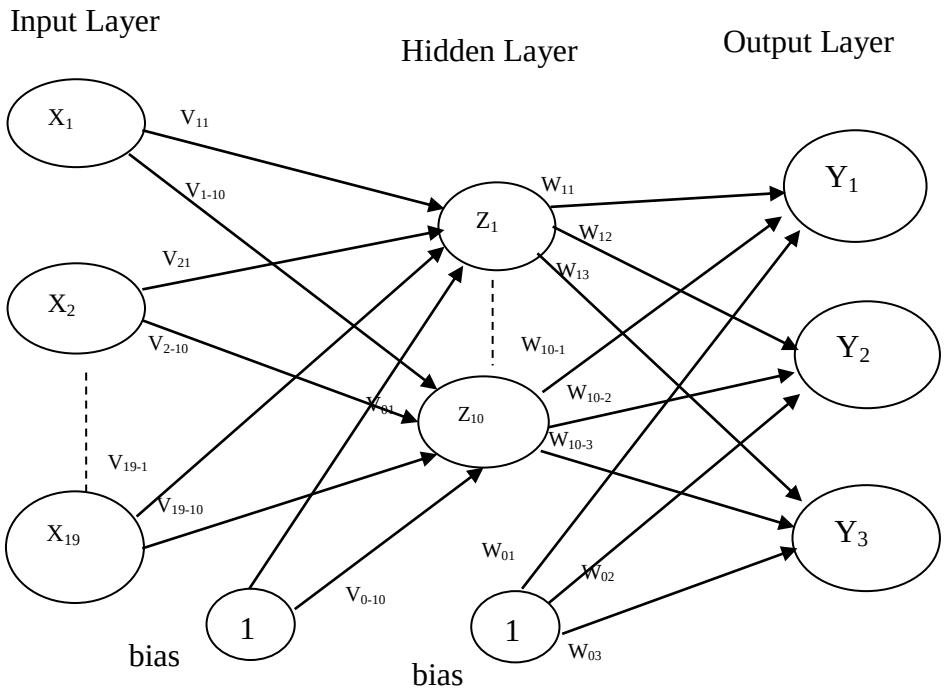
Tidak ada ketentuan yang pasti dalam menentukan jumlah *neuron* pada *hidden layer*, namun jumlah *hidden layer* berpengaruh terhadap akurasi dan laju pemebelajaran dari metode *backpropagation. Hidden layer* merupakan variabel yang dapat diubah-ubah untuk mengetahui akurasi dari metode ini, pada analisis metode ini digunakan 10 *neuron* pada *hidden layer* berdasarkan prosiding "Jaringan Syaraf Tiruan dengan Metode *Backpropagation* untuk Mendeteksi Gangguan Psikologi" karya Kiki dan Sri Kusumadewi [1]. Pada *hidden layer* terjadi perhitungan bobot dan bias dan juga aktivasinya. *Hidden*

layer pada *backpropagation* berfungsi sebagai tempat untuk mengoreksi dan menyesuaikan bobot, sehingga didapatkan nilai bobot yang baru yang bisa diarahkan mendekati dengan target *output* yang diinginkan.

3) *Output layer*

Output layer (Y_i) pada contoh kasus ini memiliki 3 *neuron,* karena jumlah keluaran pada kasus ini berupa kode penyakit yang berjumlah 3 digit. Kombinasi dari semua *neuron* tersebut digunakan untuk mengidentifikasi dan juga *learning* sebagai *output* yang seharusnya.

Arsitektur jaringan pada contoh kasus ini dapat dilihat pada gambar 3.1.



Gambar 3.1 Arsitektur jaringan metode backpropagation pada pengidentifikasi penyakit mata menular

Keterangan :

X_i : Lapisan Input terdiri dari 19 *neuron*

V_{ij} : Bobot pada lapisan tersembunyi.

Z_j : Lapisan Tersembunyi terdiri dari 10 *neuron*

W_{jk} : Bobot pada lapisan keluaran.

V_{0j} : Bias pada lapisan tersembunyi

W_{0k} : Bias pada lapisan keluaran.

i : Variabel yang mewakili *input layer*

j : Variabel yang mewakili *hidden layer*

k : Variabel yang mewakili *output layer*

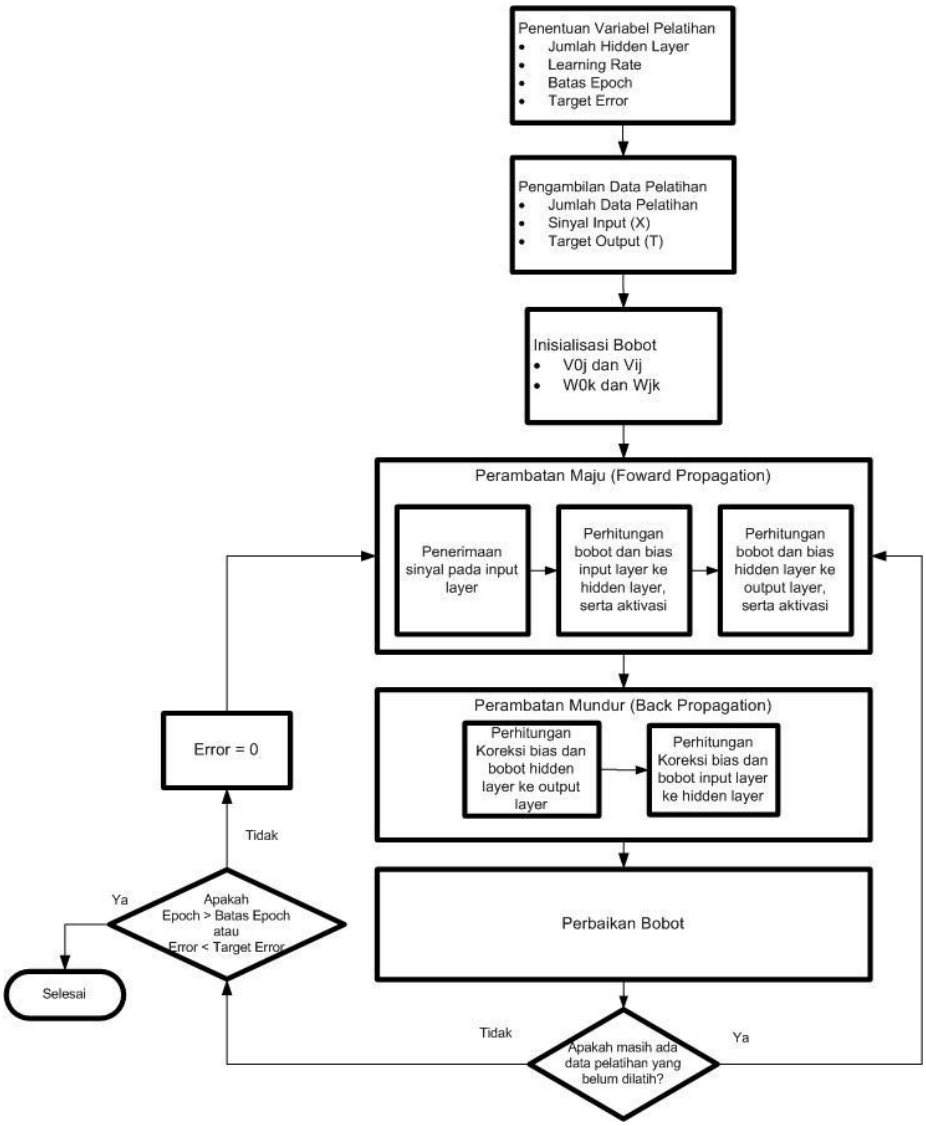
n : Jumlah *neuron* dalam suatu lapisan.

Y_k : Lapisan Output terdiri dari 3 *neuron*

1 : Konstanta bias.

b. Pelatihan Metode *Backpropagation*

Pelatihan dilakukan agar metode ini dapat mempelajari berbagai pola yang terjadi pada sebuah kasus sebelum dapat mengidentifikasi sebuah pola pada kasus tersebut. *Block* diagram menggambarkan setiap blok atau bagian dari alur kerja. *Block* diagram metode pelatihan *backpropagation* yang diimplementasikan pada sistem pengidentifikasi penyakit mata menular dapat dilihat pada gambar 3.2



Gambar 3.2 Block diagram metode pembelajaran backpropagation

Penjelasan dari *block diagram* tersebut akan diimplementasikan pada contoh kasus menggunakan data pasien yang menderita penyakit *konjungtivitis* dengan gejala mata merah (X_1), mata terasa perih (X_3), mata terasa gatal (X_4), mata berair (X_5), mata mengeluarkan kotoran(belekan) (X_6), serta penglihatan kabur (X_7).

1) Penentuan Variabel Pelatihan

Tahap pertama pada metode *backpropagation* yang diimplementasikan pada pengidentifikasi penyakit mata menular adalah penentuan variabel pelatihan. Terdapat empat variabel pelatihan yang ditentukan, yaitu jumlah *neuron* pada *hidden layer*, *learning rate*(α), batas epoch, dan batas iterasi.

Penentuan nilai variabel pelatihan dipengaruhi pemilihan jumlah lapisan tersembunyi (*hidden layer*) dan nilai *learning rate* (laju pembelajaran) yang sesuai. Nilai *learning rate* yang tidak sesuai berdampak pada hasil klasifikasi yang kurang optimal. Apabila nilai *learning rate* terlalu besar, jaringan akan mencapai konvergen dalam waktu yang singkat tetapi *error* klasifikasi menjadi besar. Sebaliknya bila *learning rate* terlalu rendah, hasil klasifikasi memiliki akurasi yang baik tetapi proses *training* membutuhkan waktu yang lama. Batas epoch dan target *error* berfungsi untuk membatasi pelatihan, dimana kedua variabel ini akan dibandingkan dengan kondisi pelatihan. Pada analisis metode ini digunakan variabel pelatihan berdasarkan prosiding "Jaringan Syaraf Tiruan dengan Metode *Backpropagation* untuk Mendeteksi Gangguan Psikologi" karya Kiki dan Sri Kusumadewi [1]. Berikut ini adalah variabel pelatihan yang akan digunakan pada analisis metode pada contoh kasus ini.

Jumlah *neuron* pada *hidden layer*	=	10
Learning Rate (α)	=	0.1
Batas Epoch	=	50000
Target Error	=	0.01

2) Pengambilan Data Pelatihan

Data pelatihan yang digunakan pada metode ini berupa pasangan sinyal *input* (X) dengan target *output* (T) yang membentuk sebuah pola. Sinyal *input* (X) memilki

nilai 0 atau 1, nilai ini didapatkan dari terjadinya gejala atau tidak terjadinya gejala. Sinyal *input* akan digunakan pada proses perambatan maju dan perbaikan bobot. Target *output* (T) merupakan pengkodean dari nama penyakit yang memiliki gejala yang sesuai dengan sinyal *input* yang menjadi pasangan dari target *output* ini, untuk pengkodean jenis penyakit mata menular dapat dilihat pada tabel 3.6. Target *output* akan digunakan untuk menghitung *error* dan juga digunakan pada tahap perambatan mundur. Contoh dari salah satu pola data pelatihan yang terdiri dari pasangan sinyal *input* dan target *output* berdasarkan contoh kasus dapat dilihat pada tabel 3.2 dan 3.3 berikut.

Contoh:

Sinyal *input*

Tabel 3.2 Sinyal input untuk gejala penyakit konjungtivitis

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
1	0	1	1	1	1	1	0	0	0
X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}	X_{19}	
0	0	0	0	0	0	0	0	0	

Target *output*

Tabel 3.3 Target output untuk penyakit konjungtivitis

T_1	T_2	T_3
0	0	1

3) Inisialisasi Bobot

Inisialisasi bobot dan bias dilakukan untuk seluruh bobot dan bias yang menghubungkan antara *input layer* ke *hidden layer* dan *hidden layer* ke *output* layer.

Bobot yang menghubungkan antara *input layer* ke *hidden layer* dilambangkan dengan V_{ij} sedangkan untuk biasnya dilambangkan dengan V_{0j}. Bobot yang menghubungkan antara *hidden layer* ke *output* layer dilambangkan dengan W_{jk} sedangkan untuk biasnya dilambangkan dengan W_{0k}. Inisialisasi ini dilakukan dengan memberi nilai acak terkecil antara 0,0 hingga 0,9 ke seluruh bobot dan bias.

Contoh :

$V_{11} = 0.2643$

$V_{01} = 0.4356$

$W_{11} = 0.4543$

$W_{01} = 0.3242$

4) Perambatan Maju (*Foward Propagation*)

Setelah dilakukan tahap inisialisasi, maka seluruh bobot dan bias yang menghubungkan *layer-layer* telah terisi oleh nilai acak kecil. Pada tahap perambatan maju ini dilakukan penjumlahan seluruh bobot dan bias tersebut dan dilakukan aktivasi pada hasil penjumlahannya. Pada tahap perambatan maju terdapat tiga langkah yaitu penerimaan sinyal pada *input layer*, perhitungan bobot dan bias *input layer* ke *hidden layer*, dan perhitungan bobot da bias *hidden layer* ke *output layer*.

a) Penerimaan sinyal pada *input layer*

Langkah awal pada tahap perambatan maju adalah penyebaran sinyal *input* yang telah didapatkan dari data pelatihan ke setiap *neuron* pada *input layer*. Penyebaran sinyal *input* ke *input layer* pada contoh kasus ini dapat dilihat pada tabel 3.4.

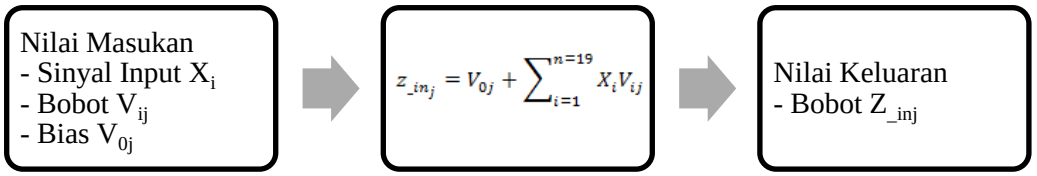
.

Tabel 3.4 Penerimaan sinyal pada input layer

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
1	0	1	1	1	1	1	0	0	0
X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}	X_{19}	
0	0	0	0	0	0	0	0	0	

b) Perhitungan bobot dan bias *input layer* ke *hidden layer,* serta aktivasi

Pada langkah ini digunakan nilai sinyal *input*(X_i), nilai bobot(V_{ij}), dan nilai bias($V0_j$) yang menghubungkan *input layer* pada *hidden layer,* untuk menghitung jumlah seluruh bobot *input layer* yang menuju ke *hidden layer* tertentu menggunakan persamaan (2.10). Proses perhitungan bobot dan bias *input layer* ke *hidden layer* dapat dilihat pada gambar 3.3 berikut.

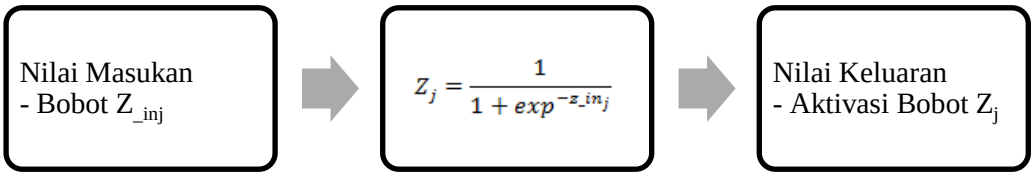


Gambar 3.3 Proses perhitungan seluruh bobot dan bias input layer ke hidden layer

Berikut adalah salah satu contoh perhitungan bobot dan bias *input layer* ke *hidden layer* dari analisis metode ini.

$z_{_in_1}$ = 0.4356 + (1 x 0.2643) + (0 x 0.3623) + (1 x 0.1385) + (1 x 0.4643) +
(1 x 0.6537) + (1 x 0.8933) + (1 x 0.6512) + (0 x 0.8246) + (0 x 0.2319) +
(0 x 0.5634) + (0 x 0.1745) + (0 x 0.9127) + (0 x 0.5246) + (0 x 0.5524) +
(0 x 0.7623) + (0 x 0.6723) + (0 x 0.2342) + (0 x 0.9248) + (0 x 0.3494)
=3.5009

Dari perhitungan tersebut dihasilkan jumlah seluruh bobot *input layer* yang menuju ke *hidden layer* ke j yang dilambangkan dengan Z_{in_j}. Nilai Z_{in_j} ini kemudian diaktivasi menggunakan rumus aktivasi sigmoid biner dengan persamaan (2.12). Proses perhitungan aktivasi dapat dilihat pada gambar 3.4 berikut.



Gambar 3.4 Proses aktivasi hasil perhitungan bobot dan bias input layer ke hidden layer

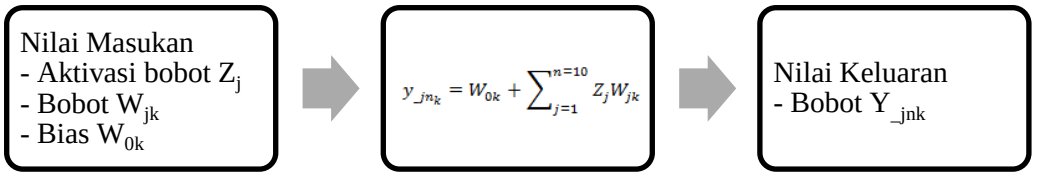
Berikut adalah salah satu contoh perhitungan aktivasi bobot dan bias *input layer* ke *hidden layer* dari analisis metode ini.

$$Z_1 = \frac{1}{1+exp^{-3.5009}} = \frac{1}{1+0.030170217} = \frac{1}{1.030170217} = 0.970713367$$

Dari perhitungan aktivasi nilai Z_{in_j} ini dihasilkan nilai aktivasi yang dilambangkan dengan Z_j. Nilai Z_j ini akan digunakan pengganti sinyal *input* pada langkah perhitungan bobot dan bias *hidden layer* ke *output layer.*

c) Perhitungan bobot dan bias *hidden layer* ke *output layer*, serta aktivasi

Pada langkah ini digunakan nilai aktivasi bobot *input layer* ke *hidden layer*(Z_j) sebagai sinyal pada *hidden layer*, bobot(W_{ij}), dan bias($W0_j$) yang menghubungkan *hidden layer* pada *output layer*, untuk menghitung jumlah seluruh bobot *hidden layer* yang menuju ke *output layer* ke k menggunakan persamaan (2.13). Proses perhitungan bobot dan bias *hidden layer* ke *output layer* dapat dilihat pada gambar 3.5 berikut.



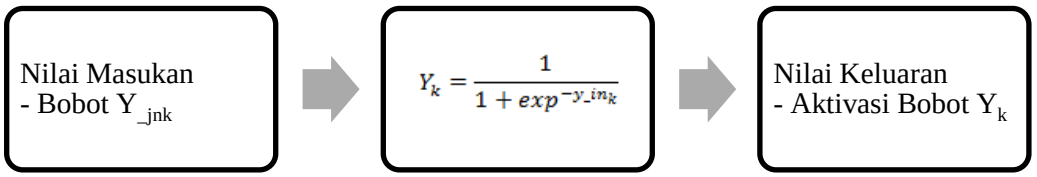
Gambar 3.5 Proses perhitungan seluruh bobot dan bias hidden layer ke output layer

Berikut adalah salah satu contoh perhitungan bobot dan bias *hidden layer* ke *output layer* dari analisis metode ini.

$y_{_jn_1}$ = 0.3242 + (0.970713367 x 0.4543) + (0.974032527 x 0.7645) + (0.950329616 x 0.2324) + (0.954034115 x 0.6575) + (0.950395657 x 0.1232) + (0.979411113 x 0.3425) + (0.930603697 x 0.2478) + (0.986094941 x 0.4358) + (0.975209696 x 0.4368) + (0.953562601 x 0.6573)

= 4.523606098

Dari perhitungan tersebut dihasilkan jumlah seluruh bobot *hidden layer* yang menuju ke *output layer* tertentu yang dilambangkan dengan Y_{jn_k}. Nilai Y_{jn_k} ini kemudian diaktivasi menggunakan rumus aktivasi sigmoid biner dengan persamaan (2.15). Proses perhitungan aktivasi dapat dilihat pada gambar 3.6 berikut.



Gambar 3.6 Proses aktivasi hasil perhitungan bobot dan bias hidden layer ke output layer

Berikut adalah salah satu contoh perhitungan aktivasi bobot dan *hidden layer* ke *output layer* dari analisis metode ini.

$$Y_1 = \frac{1}{1+exp^{-4.523606098}} = \frac{1}{1+0.010849825} = \frac{1}{1.010849825} = 0.98926663$$

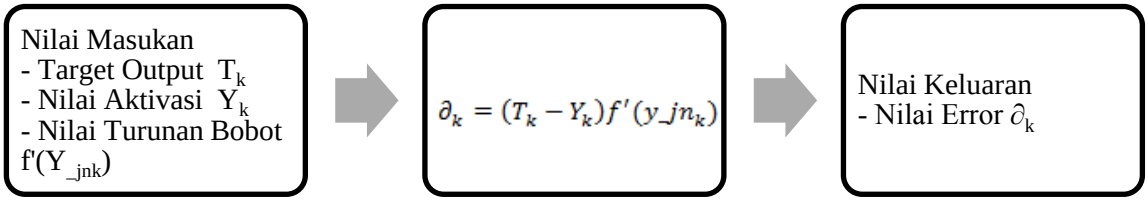
Dari perhitungan aktivasi nilai Y_{jn_k} ini dihasilkan nilai aktivasi yang dilambangkan dengan Y_k. Nilai Y_k ini digunakan untuk menghitung error yang digunakan untuk menghitung koreksi bobotpada tahap perambatan mundur.

5) Perambatan Mundur (*Back Propagation*)

Setelah melewati tahap perambatan maju (*foward propagation)*, maka tahap selanjutnya adalah tahap perambatan mundur (*back propagation*). Pada tahap ini dilakukan perhitungan error yang digunakan untuk menghitung koreksi untuk seluruh bobot awal. Pada tahap perambatan mundur terdapat dua langkah yaitu perhitungan bobot bias *hidden layer* ke *output layer*, dan perhitungan bobot dan bias *input layer ke hidden layer*.

a) Perhitungan koreksi bias dan bobot *hidden layer* ke *output layer*

Pada langkah ini digunakan nilai target *output*(T_k), nilai turunan dari hasil penjumlahan seluruh bobot *hidden layer* yang menuju ke *output layer*(Y_{jn_k}), dan nilai aktivasinya(Y_k), untuk menghitung nilai *error* bobot yang menghubungkan *hidden layer* dengan *output layer* menggunakan persamaan (2.16). Proses perhitungan *error* bobot yang menghubungkan *hidden layer* dengan *output layer* dapat dilihat pada gambar 3.7 berikut.

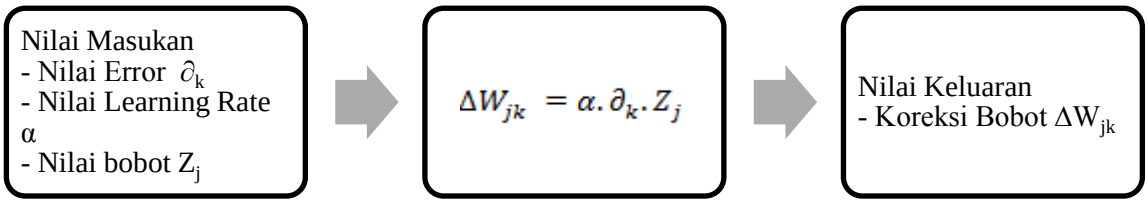


Gambar 3.7 Proses perhitungan error bobot hidden layer ke output layer

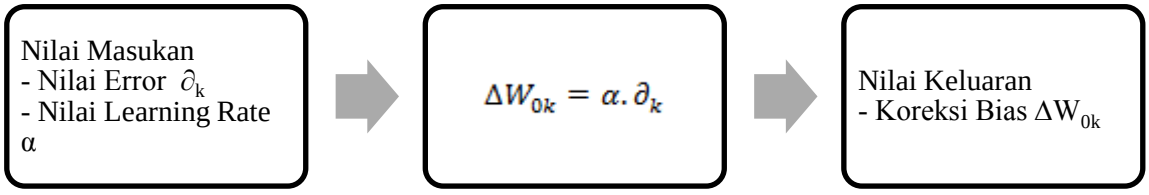
Berikut adalah salah satu contoh perhitungan *error* bobot *hidden layer* ke *output layer* dari analisis metode ini.

∂_1 = (0 - 0.98926663) x 0.98926663x (1 - 0.98926663) = -0.010504196

Dari perhitungan tersebut didapatkan nilai error pada bobot yang menghubungkan *hidden layer* dengan *output layer* yang dilambangkan dengan ∂_k. Setelah didapatkan nilai ∂_k, maka proses perhitungan dilanjutkan ke perhitungan koreksi bobot dan bias yang menghubungkan *hidden layer* ke *output layer* menggunakan persamaan (2.19) dan (2.20). Pada perhitungan koreksi bobot dan bias ini digunakan nilai *error* ∂_k, nilai *learning rate*(α), dan nilai hasil aktivasi penjumlahan seluruh bobot *input layer* yang menuju ke *hidden layer*(Z_j). Proses perhitungan koreksi bobot dan bias *hidden layer* ke *output layer* dapat dilihat pada gambar 3.8 dan 3.9 berikut.



Gambar 3.8 Proses perhitungan koreksi bobot hidden layer ke output layer



Gambar 3.9 Proses perhitungan koreksi bias hidden layer ke output layer

Berikut adalah salah satu contoh perhitungan koreksi bias *hidden layer* ke *output layer* dari analisis metode ini.

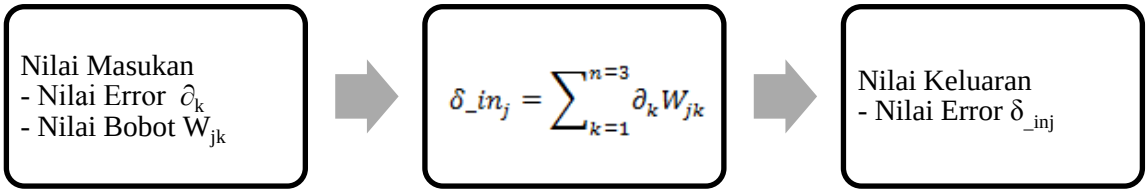
ΔW_{11} = 0.1 x -0.010504196 x 0.970713367 = -0.001019656

ΔW_{01} = 0.1 x -0.010504196 = -0.0010504196

Dari perhitungan tersebut didapatkan perbaikan bobot dan bias yang menghubungkan *hidden layer* ke *output layer* yang dilambangkan dengan ΔW_{jk} dan ΔW_{0k}. Nilai ini akan digunakan untuk memperbaiki bobot dan bias yang menghubungkan *hidden layer* ke *output layer* menjadi bobot baru yang digunakan untuk iterasi selanjutnya.

b) Perhitungan koreksi bias dan bobot *input layer* ke *hidden layer*

Setelah didapatkan nilai koreksi bobot dan bias *hidden layer* ke *output layer*, maka langkah selanjutnya adalah menghitung nilai koreksi bobot dan bias *input layer* ke *hidden layer*. Dibutuhkan nilai *error* pada bobot yang menghubungkan *input layer* ke *hidden layer* sebelum dapat menghitung koreksi bobotnya, untuk menghitung nilai *error* digunakan persamaaan (2.12). Proses perhitungan *error* bobot yang menghubungkan *input layer* dengan *hidden layer* dapat dilihat pada gambar 3.10 berikut.

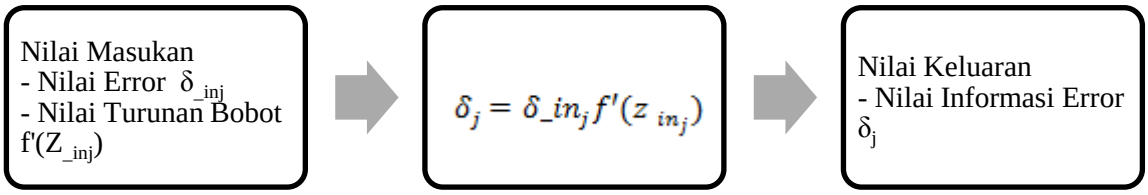


Gambar 3.10 Proses perhitungan error bobot input layer ke hidden layer

Berikut adalah salah satu contoh perhitungan *error* bobot *input layer ke hidden layer* dari analisis metode ini.

δ_in_1 = (-0.010504196 x 0.4543) + (-0.004430397 x 0.6574) +
(0.0000306743 x 0.6456)
= -0.007664796

Hasil dari perhitungan nilai *error* bobot *input layer ke hidden layer* kemudian dikalikan dengan turunan dari fungsi aktivasinya menggunakan persamaan (2.22) agar didapatkan nilai informasi *error* yang menghubungkan *input layer ke hidden layer*. Proses perhitungan informasi *error* bobot *input layer ke hidden layer* dapat dilihat pada gambar 3.11 berikut.

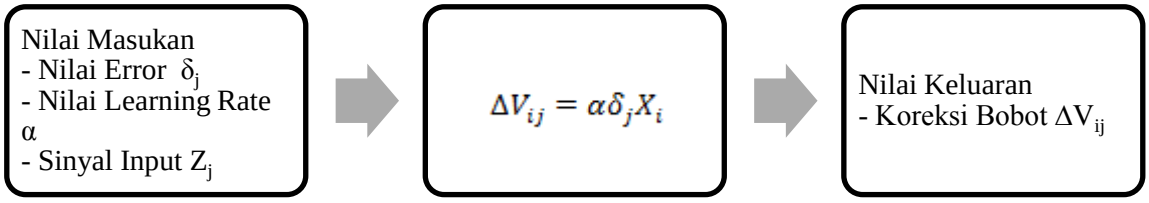


Gambar 3.11 Proses perhitungan informasi error bobot input layer ke hidden layer

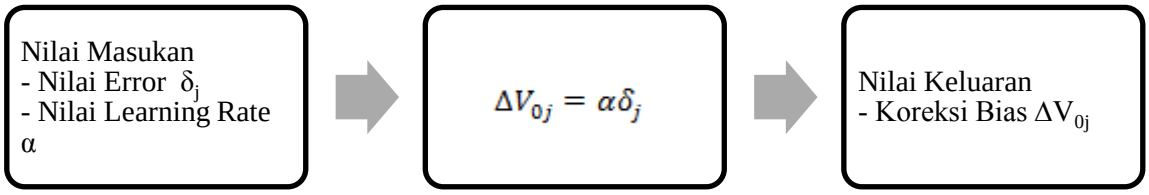
Berikut adalah salah satu contoh perhitungan informasi *error* bobot *input layer ke hidden layer* dari analisis metode ini.

δ_1 = (-0.007664796) x 0.970713367 x (1 - 0.970713367) = -0.000217902

Dari perhitungan tersebut didapatkan nilai informasi error pada bobot yang menghubungkan *input layer* dengan *hidden layer* yang dilambangkan dengan δ_j. Setelah didapatkan nilai δ_j, maka proses perhitungan dilanjutkan ke perhitungan perbaikan bobot dan bias yang menghubungkan *input layer* ke *hidden layer*. Pada perhitungan ini digunakan nilai *error* δ_j, nilai *learning rate*(α), dan nilai sinyal *input*(X_i), yang dihitung menggunakan persamaan (2.23) dan (2.24). Proses perhitungan koreksi bobot dan bias *input layer* ke *hidden layer* dapat dilihat pada gambar 3.12 dan 3.13 berikut.



Gambar 3.12 Proses perhitungan koreksi bobot input layer ke hidden layer



Gambar 3.13 Proses perhitungan koreksi bias input layer ke hidden layer

Berikut adalah salah satu contoh perhitungan koreksi bobot dan bias *input layer* ke *hidden layer* dari analisis metode ini.

ΔV_{11} = 0.1 x -0.000217902 x 1 = -0.00002179

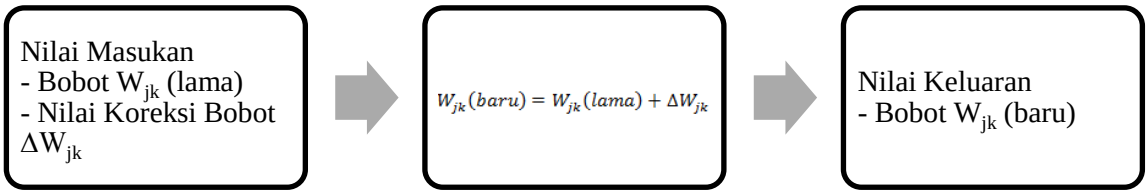
ΔV_{01} = 0.1 x -0.000217902 = -0.00002179

Dari perhitungan tersebut didapatkan perbaikan bobot dan bias yang menghubungkan *input layer* ke *hidden layer* yang dilambangkan dengan ΔV_{ij} dan

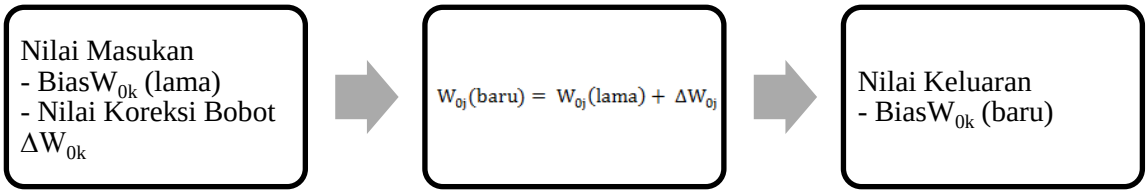
ΔV_{0j}. Nilai ini akan digunakan untuk memperbaiki bobot dan bias yang menghubungkan *input layer* ke *hidden layer* menjadi bobot baru yang digunakan untuk iterasi selanjutnya.

6) Perbaikan Bobot

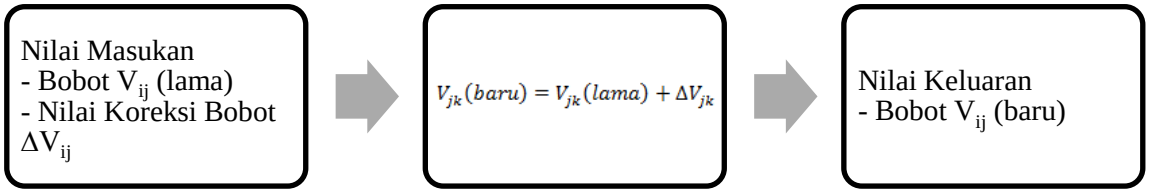
Setelah dilakukan tahap *back propagation* maka didapatkan nilai perbaikan seluruh bobot. Pada tahap perbaikan bobot ini digunakan seluruh nilai bobot dan bias yang lama (V_{ij}, V_{0j}, W_{jk}, dan W_{0k}) dan seluruh nilai perbaikan dari seluruh bobot dan bias (ΔV_{ij}, ΔV_{0j}, ΔW_{jk}, dan ΔW_{0k}) untuk menghitung nilai bobot dan bias yang baru. Nilai bobot dan bias yang baru akan dihasilkan dengan cara menambahkan bobot dan bias awal dengan perbaikan bobot dan bias yang dihitung dengan persamaan (2.25), (2.26), (2.27), dan (2.28). Proses perbaikan seluruh bobot dan bias dapat dilihat pada gambar 3.14, 3.15, 3.16, dan 3.17 berikut.



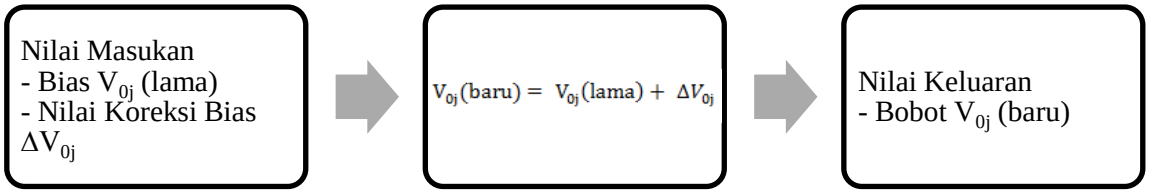
Gambar 3.14 Proses perhitungan bobot hidden layer ke output layer baru



Gambar 3.15 Proses perhitungan bias hidden layer ke output layer baru



Gambar 3.16 Proses perhitungan bobot input layer ke hidden layer baru



Gambar 3.17 Proses perhitungan bias input layer ke hidden layer baru

Berikut adalah salah satu contoh perbaikan bobot dan bias *hidden layer* ke *output layer* dan *input layer* ke *hidden layer* dari analisis metode ini.

$W_{11}(baru)$ = 0.4543 + (-0.001019656) = 0.453280344
$W_{01}(baru)$ = 0.3242 + (-0.00105042) = 0.32314958
$V_{11}(baru)$ = 0.2643 + (-0.0000217902) = 0.26427821
$V_{01}(baru)$ = 0.4356 + (-0.0000217902) = 0.43557821

Dari perhitungan tersebut didapatkan seluruh nilai bobot dan bias yang baru yang dilambangkan dengan lambang yang sama dengan bobot dan bias awal V_{ij}, V_{0j}, W_{jk}, dan W_{0k}. Nilai bobot dan bias yang baru ini digunakan untuk iterasi selanjutnya.

7) Cek Kondisi Data Pelatihan

Setelah didapatkan bobot yang baru, maka tahap selanjutnya adalah cek kondisi data pelatihan. Kondisi yang harus dipenuhi pada tahap ini adalah seluruh

data pelatihan sudah dilatih. Jika kondisi dipenuhi, maka proses pelatihan akan dilanjutkan kepada tahap selanjutnya, sedangkan jika kondisi belum dipenuhi maka proses pelatihan akan kembali ke tahap perambatan maju.

8) Cek Kondisi Epoch dan Error

Tahap akhir pada metode ini adalah pengecekan kondisi akhir dari epoch dan *error*. Terdapat dua kondisi yang harus dipenuhi. Kondisi pertama adalah jika jumlah iterasi(epoch) telah mencapai batas epoch yang telah ditentukan, dan kondisi kedua adalah jika nilai *error* telah mencapai target *error*. Nilai *error* yang akan dibandingkan didapatkan dari hasil pengurangan nilai target *output* (T_k) dengan hasil aktivasi bobot *hidden layer* ke *output layer* (Y_k). Berikut adalah contoh perhitungan *error* yang digunakan untuk perbandingan kondisi.

Error $= T_k - Y_k$
Error = 0 - 0.922375242
= -0.922375242

Setelah didapatkan hasil error tersebut, maka seluruh error dibagi dengan jumlah error dan hasilnya dikuadratkan. Hasil dari error kuadrat ini yang nantinya akan dibandingkan dengan target *error* yang sudah ditentukan. Pada tahap ini jika salah satu kondisi terpenuhi maka proses pelatihan akan berhenti, sedangkan jika tidak ada kondisi yang terpenuhi maka nilai error akan diinisialisasi menjadi 0 dan proses pelatihan akan kembali ke tahap perambatan maju.

c. Pengujian Metode *Backpropagation*

Setelah dilakukan pelatihan *backpropagation*, maka selanjutnya dilakukan pengujian *backpropagation* untuk mengidentifikasi pola yang sudah ditentukan untuk

diidentifikasi. Proses perhitungan pada tahap pengujian metode *backpropagation* hanya melewati proses perambatan maju. *Block* diagram menggambarkan setiap blok atau bagian dari alur kerja. *Block* diagram metode *backpropagation* tahap pengujian yang diimplementasikan pada sistem pengidentifikasi penyakit mata menular dapat dilihat pada gambar 3.18



Gambar 3.18 Block diagram metode pengujian backpropagation

1) Pengambilan Bobot dan Bias Hasil Pelatihan

Pada pengujian *backpropagation,* bobot dan bias yang digunakan adalah bobot dan bias hasil dari pelatihan yang sudah melewati iterasi sampai akhir pelatihan. Misal setelah epoch ke 7503 diperoleh nilai bobot V, bias V_0, bobot W, dan bias W_0 sebagai berikut.

Contoh :

$V_{11} = 1.6769$

$V_{01} = 0.4356$

$W_{11} = -1.5233$

$W_{01} = 0.3242$

2) Perambatan Maju (*Foward propagation*)

Tahap perambatan maju pada pengujian *backpropagation,* umumnya sama seperti tahap perambatan maju pada pelatihan *backpropagation.* Pada tahap perambatan maju terdapat tiga langkah yaitu penerimaan sinyal pada *input layer*,

perhitungan bobot dan bias *input layer* ke *hidden layer*, juga perhitungan bobot dan bias *hidden layer* ke *output layer*.

a) Penerimaan siyal pada *input layer*

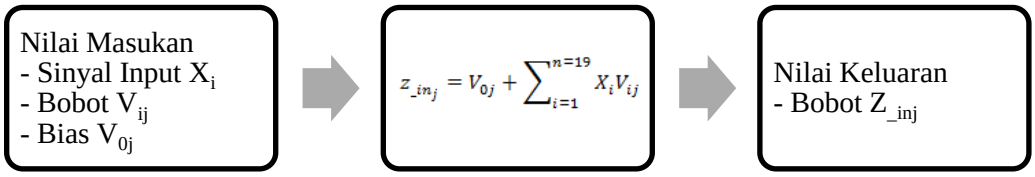
Pada langkah ini, sinyal *input* diterima dari pola yang dijadikan data uji. Sebagai contoh, data uji menggunakan data pasien yang menderita penyakit konjungtivitis dengan gejala mata merah (X_1), mata terasa perih (X_3), mata terasa gatal (X_4), mata berair (X_5), mata mengeluarkan kotoran(belekan) (X_6), serta penglihatan kabur (X_7). Penyebaran sinyal *input* ke *input layer* pada contoh kasus ini dapat dilihat pada tabel 3.5.

Tabel 3.5 Penerimaan sinyal pada input layer pengujian

X_1	**X_2**	**X_3**	**X_4**	**X_5**	**X_6**	**X_7**	**X_8**	**X_9**	**X_{10}**
1	0	1	1	1	1	1	0	0	0
X_{11}	**X_{12}**	**X_{13}**	**X_{14}**	**X_{15}**	**X_{16}**	**X_{17}**	**X_{18}**	**X_{19}**	
0	0	0	0	0	0	0	0	0	

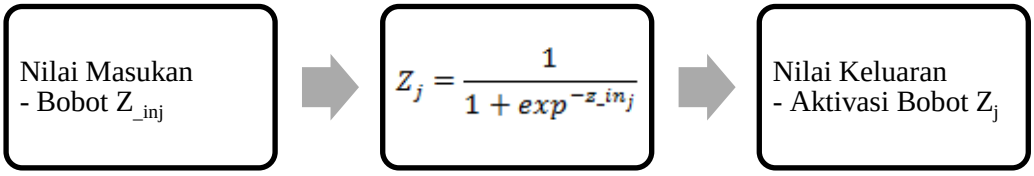
b) Perhitungan bobot dan bias *input layer* ke *hidden layer*, serta aktivasi

Langkah selanjutnya adalah perhitungan bobot dan bias *input layer* ke *hidden layer* menggunakan persamaan (2.10). Proses perhitungan bobot dan bias *input layer* ke *hidden layer* dapat dilihat pada gambar 3.19 berikut.



Gambar 3.19 Proses perhitungan seluruh bobot dan bias input layer ke hidden layer pada tahap pengujian

Setelah didapatkan nilai hasil perhitungan dari seluruh bobot dan bias *input layer* ke *hidden layer*, maka nilai tersebut diaktivasi menggunakan rumus aktivasi sigmoid biner dengan persamaan (2.12). Proses perhitungan aktivasi dapat dilihat pada gambar 3.20 berikut.

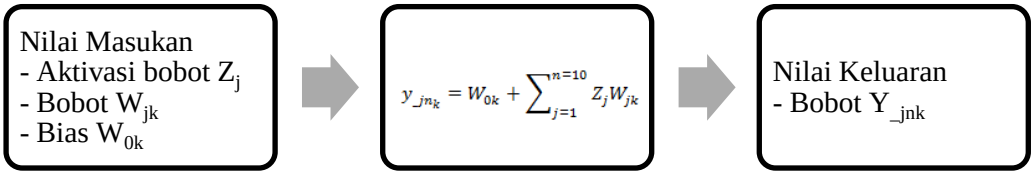


Gambar 3.20 Proses aktivasi hasil perhitungan bobot dan bias input layer ke hidden layer pada tahap pengujian

Nilai hasil aktivasi ini akan digunakan sebagai pengganti sinyal *input* pada langkah perhitungan bobot dan bias *hidden layer* ke *output layer*.

c) Perhitungan bobot dan bias *hidden layer* ke *output layer*, serta aktivasi

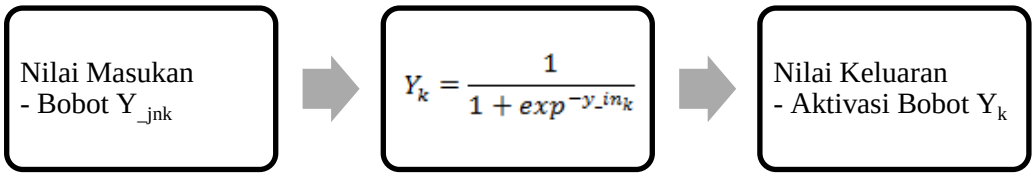
Pada langkah ini, digunakan nilai aktivasi bobot Z_j sebagai pengganti sinyal *input* pada langkah perhitungan bobot dan bias *hidden layer* ke *output layer* dengan persamaan (2.13). Proses perhitungan bobot dan bias *hidden layer* ke *output layer* dapat dilihat pada gambar 3.21 berikut.



Gambar 3.21 Proses perhitungan seluruh bobot dan bias hidden layer ke output layer pada tahap pengujian

Setelah didapatkan nilai hasil perhitungan dari seluruh bobot dan bias *hidden layer* ke *output layer*, maka nilai tersebut diaktivasi menggunakan rumus aktivasi

sigmoid biner dengan persamaan (2.15). Proses perhitungan aktivasi dapat dilihat pada gambar 3.22 berikut.



Gambar 3.22 Proses aktivasi hasil perhitungan bobot dan bias hidden layer ke output layer pada tahap pengujian

Dari perhitungan aktivasi nilai Y_{jn_k} ini dihasilkan nilai aktivasi yang dilambangkan dengan Y_k. Pada tahap pengujian ini nilai Y_k digunakan untuk menentukan hasil identifikasi yang didapatkan oleh metode *backpropagation*.

3) Identifikasi Hasil

Setelah didapatkan nilai aktivasi Y_k, maka nilai aktivasi dibandingkan dengan nilai ambang(*threshold*) yang bernilai 0.5 , yang berarti jika nilai *output*(Y_k) lebih besar dari 0.5 maka *output* yang diberikan adalah 1, jika nilai *output* lebih kecil dari 0,5 maka *output* yang diberikan adalah 0. Berikut ini adalah contoh nilai aktivasi Y_k yang dihasilkan dari perhitungan contoh kasus yang digunakan.

Contoh : Y_1 = 0.080374474

Y_2 = 0.071814849

Y_3 = 0.886581539

Dengan demikian hasil *output* dari X = [1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0] adalah Y = [0 0 1] (karena Y_1 = 0.080374474 < 0.5, Y_2 = 0.071814849 < 0.5, dan Y_3 = 0.886581539 > 0.5). Sesuai dengan target yang diharapkan yaitu [0 0 1] sebagai kode untuk penyakit *Konjungtivitis.*

3.4 Analisis Target Keluaran

Target keluaran atau *output* yang dihasilkan sistem berupa salah satu dari ke 7 jenis penyakit yang teridentifikasi oleh sistem yaitu *Konjungtivitis, Keratongjutivitas Vernalis, Endoftalmitis, Selulitis Orbitais, Trakoma, Blefaritis,* dan *Ulkus Kornea* yang telah dirubah menjadi sebuah kode biner untuk memudahkan pelatihan metode *backpropagation*. *Sample data* yang telah dikumpulkan, dijadikan sebagai bahan pelatihan untuk sistem. Sistem akan mengidentifikasi jenis penyakit yang diderita berdasarkan gejala yang muncul. Nama-nama penyakit beserta kode yang digunakan sebagai target keluaran pada sistem, dapat dilihat pada tabel 3.6.

Tabel 3.6 Target keluaran sistem penyakit mata menular

Jenis Penyakit	Kode
Konjungtivitis	0 0 1
Keratokonjungtivitas vernalis	0 1 0
Endoftalmitis	0 1 1
Selulitis Orbitalis	1 0 0
Trakoma	1 0 1
Blefaritis	1 1 0
Ulkus Kornea	1 1 1

3.5 Analisis Kompleksitas

Perhitungan kompleksitas waktu asimptotik metode *backpropagation* dapat dihitung dengan menelusuri setiap langkah algoritma metode *backpropagation* pada *pseudocode* metode ini. Proses yang menjadi inti dari algoritma metode *backpropagation,* adalah prosedure pelatihan jaringan syaraf tiruan. *Pseudocode* prosedure pelatihan jaringan syaraf tiruan yang menghasilkan nilai parameter jaringan yaitu nilai V_{ij}, W_{jk}, V_0, $W_{0,}$ dapat dilihat pada tabel 3.7.

Tabel 3.7 Perhitungan Big-O prosedure latihJST

No	**Algoritma**	Nilai Big-O
1	MSEepoch := maxMSE +1	O(1)
2	ee := 1	O(1)
3	**while** (ee <= Epoch) **and** (MSEepoch > MaxMSE) **do**	O(n)
4	MSEepoch := 0	O(1)
5	{pelatihan dari pola yang ada di database}	
6	**for** pp := 1 **to** Jpola **do**	O(n)
7	{inisialisasi Vtemp, Wtemp, dan Etemp}	
8	**for** zz := 1 **to** DimPola **do**	O(n)
9	Vtemp[zz] := 0	O(1)
10	**endfor**	
11	**for** zz := 1 **to** JHneuron **do**	O(n)
12	Wtemp[zz] := 0	O(1)
13	**endfor**	
14	**for** zz := 1 **to** Joutput **do**	O(n)
15	Etemp[zz] := 0	O(1)
16	**endfor**	
17	{pengisian pola yang akan dilatih }	
18	**for** ii := 1 **to** DimPola **do**	O(n)
19	CP[ii] := P[pp][ii]	O(1)
20	**endfor**	
21	{pengisian target output untuk pola yang dilatih}	
22	**for** ii := 1 **to** Joutput **do**	O(n)
23	CT[ii] := T[pp][ii]	O(1)
24	**endfor**	
25	{perhitungan bobot input ke hidden dan aktivasinya}	
26	**for** xx := 1 **to** JHneuron **do**	O(n)
27	**for** zz := 1 **to** DimPola **do**	O(n)
28	Vtemp[xx] := Vtemp[xx] + (W1[xx][zz]*CP[zz])	O(1)

29	**endfor**	
30	**endfor**	
31	**for** zz := 1 **to** JHneuron **do**	O(n)
32	V[zz] := Vo[zz] + Vtemp[zz]	O(1)
33	A1[zz] := 1/(1+(exp(-V[zz]))	O(1)
34	**endfor**	
35	{perhitungan bobot hidden ke output dan aktivasinya}	
36	**for** xx := 1 **to** Joutput **do**	O(n)
37	**for** zz := 1 **to** JHneuron **do**	O(n)
38	Wtemp[xx] := Wtemp[xx] + (W2[xx][zz] * V[zz])	O(1)
39	**endfor**	
40	**endfor**	
41	**for** zz := 1 **to** Joutput **do**	O(n)
42	W[zz] := Wo[zz] + Wtemp[zz]	O(1)
43	A2[zz] := 1/(1+(exp(-W[zz]))	O(1)
44	**endfor**	
45	{perhitungan error untuk perbandingan}	
46	**for** zz := 1 **to** Joutput **do**	O(n)
47	Error[zz] := CT[pp][zz] - A2[zz]	O(1)
48	**endfor**	
49	**for** kk := 1 **to** Joutput do	O(n)
50	MSEepoch := MSEepoch + (Error[kk]*Error[kk])	O(1)
51	**endfor**	
52	{perhitungan error untuk perbaikan bobot hidden ke output}	
53	**for** kk := 1 **to** Joutput **do**	O(n)
54	E2[kk] := A2[zz]*(1-A2[kk])*Error[kk]	O(1)
55	**endfor**	
56	{perhitungan bobot perbaikan hidden ke output}	
57	**for** jj := 1 **to** JHneuron **do**	O(n)
58	**for** kk := 1 **to** Joutput **do**	O(n)

59	DW2[jj][kk] := learning_rate * E2[kk] * A1[jj]	O(1)
60	**endfor**	
61	**endfor**	
62	{perhitungan bobot perbaikan bias hidden ke output}	
63	**for** jj := 1 **to** Joutput **do**	O(n)
64	DB2[jj] := learning_rate * E2[jj]	O(1)
65	**endfor**	
66	{perhitungan error untuk perbaikan bobot input ke hidden}	
67	**for** xx := 1 **to** JHneuron **do**	O(n)
68	**for** zz := 1 **to** Joutput **do**	O(n)
69	Etemp[xx] := Etemp[xx] + (W1[xx][zz]*E2[zz])	O(1)
70	**endfor**	
71	**endfor**	
72	**for** zz := 1 **to** JHneuron **do**	O(n)
73	Err2[zz] := Etemp[zz]	O(1)
74	**endfor**	
75	{aktivasi error}	
76	**for** zz := 1 **to** JHneuron **do**	O(n)
77	E1[zz] := A1[zz]*(1-A1[kk])*Err2[kk]	O(1)
78	**endfor**	
79	{perhitungan perbaikan bobot input ke hidden}	
80	**for** jj := 1 **to** DimPola **do**	O(n)
81	**for** kk := 1 **to** JHneuron **do**	O(n)
82	DW1[jj][kk] := learning_rate * E1[kk] * CP[jj]	O(1)
83	**endfor**	
84	**endfor**	
85	{perhitungan perbaikan bobot bias input ke hidden}	
86	**for** jj := 1 **to** JHneuron **do**	O(n)
87	DB1[jj] := learning_rate * E1[jj]	O(1)
88	**endfor**	

89	{perbaikan bobot input ke hidden}	
90	**for** ii := 1 **to** DimPola **do**	O(n)
91	**for** jj := 1 **to** JHneuron **do**	O(n)
92	W1[ii][jj] := W1[ii][jj] + DW1[ii][jj]	O(1)
93	**endfor**	
94	**endfor**	
95	{perbaikan bobot bias input ke hidden}	
96	**for** jj := 1 **to** JHneuron **do**	O(n)
97	B1[jj] := B1[jj] + DB1[jj]	O(1)
98	**endfor**	
99	{perbaikan bobot hidden ke output}	
100	**for** ii := 1 **to** JHneuron **do**	O(n)
101	**for** jj := 1 **to** Joutput **do**	O(n)
102	W2[ii][jj] := W2[ii][jj] + DW2[ii][jj]	O(1)
103	**endfor**	
104	**endfor**	
105	{perbaikan bobot bias input ke hidden}	
106	**for** jj := 1 **to** Joutput **do**	O(n)
107	B2[jj] := B2[jj] + DB2[jj]	O(1)
108	**endfor**	
109	**endfor**	
110	MSE[ee] := MSEepoch/Jpola	O(1)
111	ee = ee + 1	O(1)
112	**endwhile**	

Algoritma metode backpropagation terdiri dari banyak perulangan *for* atau lebih dikenal sebagai *nested for loop.* Perhitungan jumlah T(n) pada *nested for loop* memiliki dua aturan, yaitu :

1. Dianalisis dari *loop* terdalam kemudian keluar.
2. Waktu eksekusi total sebuah *statement* adalah waktu eksekusi *statement* tersebut dikalikan hasil kali dari banyaknya iterasi semua *loop* yang ada di luarnya

Dari perhitungan pada tabel 3.9 didapatkan hasil $f(n) = 30 + n + n^2 + 23n^3 + 7n^4$, maka dengan aturan *nested for loop* didapatkan hasil $T(n) = O(n^4)$. Setelah didapat hasil perhitungan big-O pada tabel 3.9, maka dapat ditentukan waktu terburuk dan waktu terbaik yang diperlukan untuk mengeksekusi algoritma *backpropagation* dengan menentukan jumlah masukan data paling sedikit dan jumlah masukan data paling banyak. Jumlah masukan data pelatihan paling sedikit adalah 100 buah data, dan paling banyak adalah 1000 buah data.

1. Waktu terbaik = $100n + n^4$ Didapatkan kompleksitas waktu asimptotik dari proses pembelajaran yaitu $O(100) + O(n^4) = O(n^4)$
2. Waktu terburuk = $1000n + n^4$ Didapatkan kompleksitas waktu asimptotik dari proses pembelajaran yaitu $O(1000) + O(n^4) = O(n^4)$

Algoritma *backpropagation* termasuk algoritma *polinominal* pada kasus terbaik dan terburuknya, karena algoritma ini memiliki kompleksitas asimptotik $O(n^4)$ untuk kedua kasus tersebut. Algoritma yang memiliki kasus *polinominal* kasus terburuk termasuk algoritma yang mangkus, karena algoritma akan menghasilkan solusi dalam waktu yang lebih pendek.

3.6 Analisis Kebutuhan Sistem

Spesifikasi kebutuhan sistem didapat berdasarkan kebutuhan perangkat lunak sistem pengidentifikasi penyakit mata menular untuk penerapan analisis metode *backpropagation*. Analisis kebutuhan perangkat lunak terbagi menjadi dua bagian, yaitu :

1. Kebutuhan non fungsional.
2. Kebutuhan fungsional.

3.6.1 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional adalah langkah dimana seorang pembangun perangkat lunak (*software developer*) menganalisis sumber daya yang akan digunakan dan menggunakan perangkat lunak yang dibangun. Perangkat keras dan perangkat lunak yang dimiliki harus sesuai dengan kebutuhan atau permintaan, sehingga dapat ditentukan kompabilitas aplikasi yang dibangun terhadap sumber daya yang ada.

Analisis kebutuhan non fungsional yang dilakukan dibagi dalam tiga tahap, yaitu:

1. Analisis perangkat keras (*hardware*).
2. Analisis perangkat lunak (*software*).
3. Analisis pengguna *(user*)

3.6.1.1 Analisis Kebutuhan Perangkat Keras (hardware)

Spesifikasi perangkat keras minimum yang butuhkan untuk menjalankan sistem pengidentifikasi penyakit mata menular ini, dapat dilihat pada tabel 3.8

Tabel 3.8 Spesifikasi perangkat keras yang dibutuhkan

No	Perangkat Keras	Spesifikasi
1	Prosesor	Kecepatan 2.00 GHz
2	Monitor	Monitor 14.1’’
3	Memori	1024 MB

3.6.1.2 Analisis Kebutuhan Perangkat Lunak (software)

Komponen perangkat lunak pendukung yang dibutuhkan untuk menjalankan sistem pengidentifikasi penyakit mata menular ini, dapat dilihat pada tabel 3.9

Tabel 3.9 Perangkat lunak yang dibutuhkan

No	Perangkat Lunak
1	Sistem operasi Windows 7
2	Visual Studio 2010
3	MySQL ver 5
4	MySQL connector ver 6

3.6.1.3 Analisis Kebutuhan Perangkat Pikir

Analisis kebutuhan perangkat pikir merupakan uraian mengenai siapa saja yang akan menggunakan sistem dan terlibat dalam pengolahan data beserta karakteristiknya sehingga dapat diketahui tingkat pengalaman dan pemahaman pengguna terhadap sistem. Adapun karakteristik dari pengguna adalah sebagai berikut:

1. Pengguna dapat membaca tulisan.
2. Pengguna minimal mampu menggunakan *keyboard* dan *mouse* sebagai untuk berinteraksi dengan sistem.
3. Pengguna sudah terbiasa menggunakan komputer.

3.6.2 Analisis Kebutuhan Fungsional

Spesifikasi kebutuhan fungsional adalah spesifikasi tentang hal-hal yang akan dilakukan sistem ketika diimplementasikan. Analisis kebutuhan ini diperlukan untuk menentukan keluaran yang akan dihasilkan sistem, masukan yang diperlukan sistem, lingkup proses yang digunakan untuk mengolah masukan menjadi keluaran.

3.6.2.1 Use Case Diagram

Use case merupakan gambaran umum dari perancangan sistem yang akan dibangun. Terdapat tiga buah poin yang merincikan *use case* diagram ini , yaitu :

1. Identifikasi *Use Case*
2. Identifikasi Aktor

3. Skenario *Use Case*

Adapun rincian dari *use case diagram* sistem pengidentifikasi penyakit mata menular dapat dilihat pada lampiran A-1.

3.6.2.2 Activity Diagram

Diagram aktivitas (*Activity Diagram*) memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Adapun rincian dari activity diagram sistem pengidentifikasi penyakit mata menular dapat dilihat pada lampiran A-2.

3.6.2.3 Class Diagram

Class diagram membantu menggambarkan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling sering ditemui dalam pemodelan sistem berbasis *object-oriented.* Adapun rincian dari class diagram sistem pengidentifikasi penyakit mata menular dapat dilihat pada lampiran A-3.

3.6.2.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antara sejumlah objek dalam urutan waktu. Adapun rincian dari sequence diagram sistem pengidentifikasi penyakit mata menular dapat dilihat pada lampiran A-4.

3.6.3 Perancangan Sistem

Perancangan merupakan bagian dari metodologi pembangunan suatu perangkat lunak yang harus dilakukan setelah melalui tahapan analisis. Pada bagian ini akan dijelaskan perancangan sistem yang dimaksudkan yang meliputi

1. perancangan basis data
2. perancangan struktur menu
3. perancangan antarmuka

Semua perancangan yang telah disebutkan diatas harus saling berhubungan agar menciptakan suatu sistem yang baik.

3.6.3.1 Perancangan basis data

Tahapan perancangan basis data dalam sistem pengidentifikasi penyakit mata menular ini berisi perancangan struktur tabel. Struktur tabel merupakan isi data yang berada dalam satu *record* pada satu tabel. Tabel-tabel yang digunakan dalam *database* aplikasi sistem pengidentifikasi penyakit mata menular adalah tabel untuk mengolah data penderita penyakit, tabel data tentang jenis penyakit mata menular serta tabel nilai MSE dan Epoch. Adapun rincian dari perancangan basis data sistem pengidentifikasi penyakit mata menular dapat dilihat pada lampiran A-5.

3.6.3.2 Perancangan Struktur Menu

Perancangan struktur menu menggambarkan keterkaitan setiap menu yang bisa diakses oleh user yang digambarkan sebagai berikut. Adapun rincian dari perancangan struktur menu sistem pengidentifikasi penyakit mata menular dapat dilihat pada lampiran A-5.

3.6.3.3 Perancangan Antarmuka

Perancangan antarmuka dibuat untuk memberi konsep pada implementasi pembuatan suatu antarmuka sistem yang akan dibangun. Perancangan antarmuka terdiri dari 3 bagian, yaitu :

1. Perancangan Form
2. Perancangan Pesan
3. Jaringan Semantik

Adapun rincian dari perancangan perancangan antarmuka sistem pengidentifikasi penyakit mata menular dapat dilihat pada lampiran A-5.