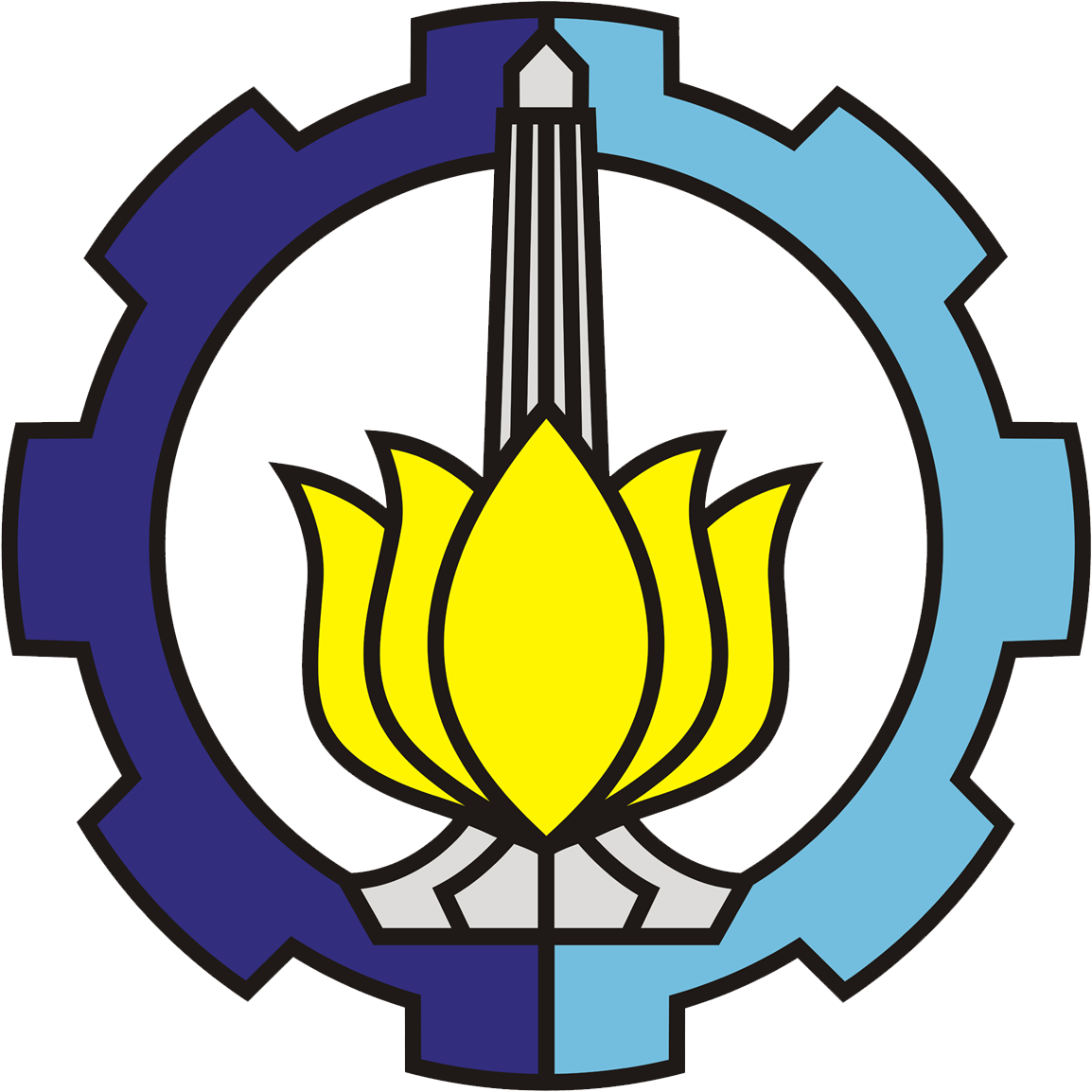
**LAPORAN TUGAS**

***5-CLASS CLASSIFIER* DENGAN *MULTI LAYER PERCEPTRON* DAN METODE PEMBELAJARAN *ERROR BACK PROPAGATION***

****

Mata Kuliah:

Dasar Sistem Cerdas EB184603

Dosen Pengampu:

Dr. Achmad Arifin, S.T., M.T.

Penyusun:

Steven Seaver Wiarta

07311640000028

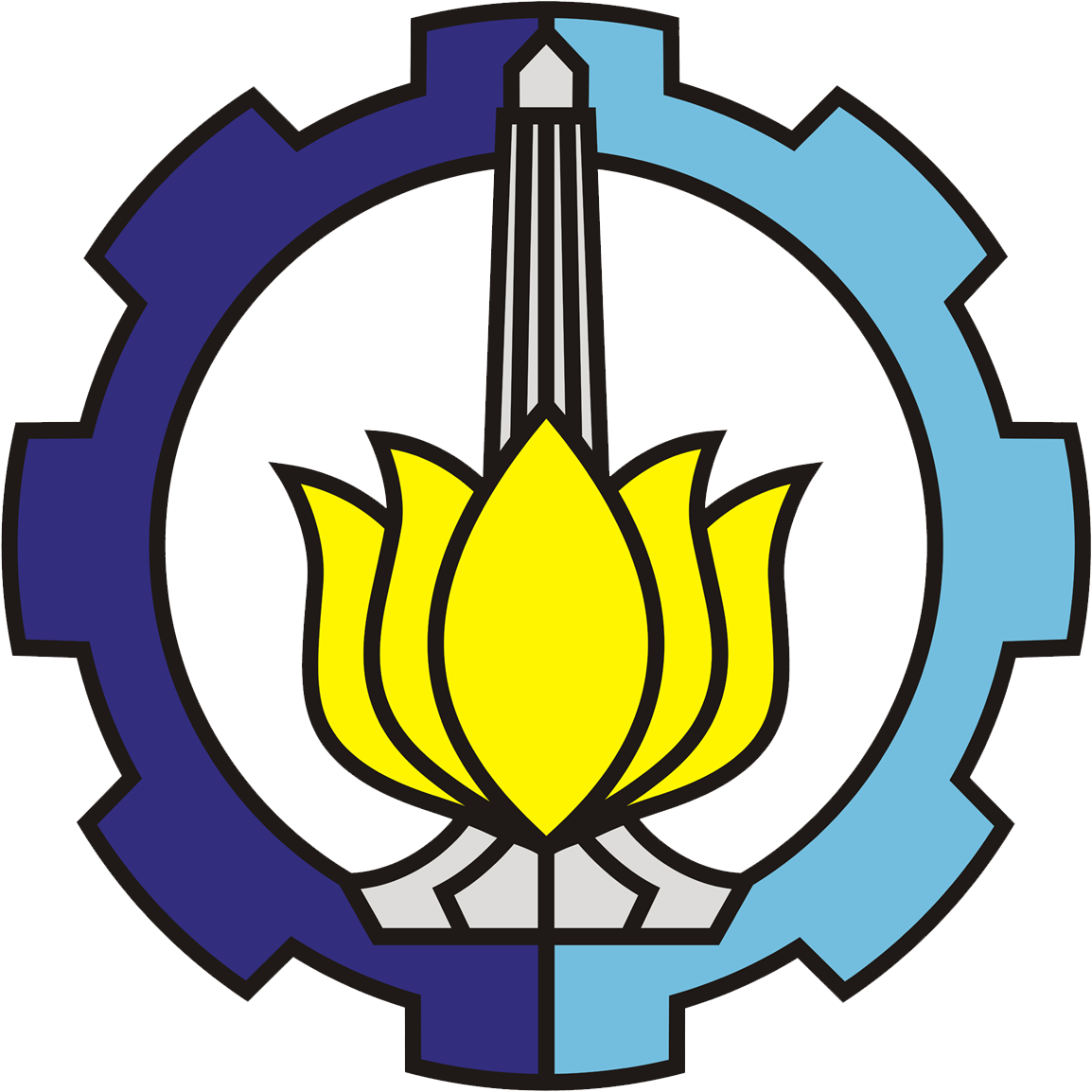
Departemen Teknik Biomedik

Fakultas Teknologi Elektro

Institut Teknologi Sepuluh Nopember

**LAPORAN TUGAS**

***5-CLASS CLASSIFIER* DENGAN *MULTI LAYER PERCEPTRON* DAN METODE PEMBELAJARAN *ERROR BACK PROPAGATION***

****

Mata Kuliah:

Dasar Sistem Cerdas EB184603

Dosen Pengampu:

Dr. Achmad Arifin, S.T., M.T.

Penyusun:

Steven Seaver Wiarta

07311640000028

Departemen Teknik Biomedik

Fakultas Teknologi Elektro

Institut Teknologi Sepuluh Nopember

## KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa karena berkat dan perkenanNya Laporan Tugas mata kuliah Dasar Sistem Cerdas: *5-Class Classifier* dengan Multi Layer Perceptron dan Metode Pembelajaran *Error Back Propagation* ini dapat diselesaikan dengan baik. Terima kasih juga disampaikan kepada dosen pengampu dan teman-teman yang telah berkontribusi menyelesaikan laporan tugas ini. Laporan Tugas mata kuliah Dasar Sistem Cerdas ini terdiri dari dasar teori, permasalahan, hasil dan analisa, serta kesimpulan yang dapat diambil dari hasil tersebut.

Melalui laporan resmi ini diharapkan pembaca juga dapat memahami pemaparan materi mata kuliah Dasar Sistem Cerdas sebagai salah satu materi dasar di Departemen Teknik Biomedik dalam lingkup Fakultas Teknologi Elektro. Laporan ini juga bersisi mengenai pembahasan mengenai teori-teori yang diuji kebenarannya melalui implementasi program yang bertumpu pada teori yang diajarkan pada perkuliahan.

Akhir kata, tiada gading yang tak retak, laporan ini masih jauh dari kata sempurna. Oleh karena itu, seluruh kritik dan saran diterima dengan baik demi memperbaiki kualitas dari laporan resmi ini. Semoga Laporan Resmi ini bermanfaat bagi pembaca.

Penulis

## DAFTAR ISI

Kata Pengantar iii

Daftar Isi iv

Bab I : Dasar Teori 5

Bab II : Permasalahan 8

Bab III : Metode 9

Bab IV : Hasil dan Analisis 11

Bab V : Kesimpulan 18

Lampiran 19

Referensi 28

## BAB I

**DASAR TEORI**

Model neuron *multi layer perceptron* atau MLP adalah pengembangan dari *single layer perceptron*, yaitu suatu kelas dari *feedforward artificial* *neural network* yang memiliki minimal tiga *layer*, yaitu *input layer, hidden layer,* dan *output layer* seperti pada Gambar 1.1. Jumlah dari *hidden layer* tidak dibatasi, sehingga dapat ditentukan dengan bebas, namu terbatas pada beban komputasi yang akan dialami oleh suatu pemodelan neuron. Metode pembelajaran atau *learning* yang digunakan adalah *supervised learning* (Rosenblatt, 1961), yaitu dengan menggunakan suatu algoritma *error back propagation*. Pada dasarnya, MLP menggunakan fungsi aktivasi non-linear pada setiap *node*-nya kecuali pada *input nodes*. Dengan jumlah *layer* yang banyak dan fungsi aktivasi yang tidak linear, MLP dapat menyelesaikan permasalahan klasifikasi untuk data yang tidak dapat dipisahkan secara *linear*.

Fungsi aktivasi *non-linear* yang digunakan adalah pemodelan dari rangsangan frekuensi ‘tembakan’ atau *firing* dari *action potential* pada suatu neuron biologis. Dua fungsi aktivasi yang paling umum digunakan adalah bersifat sigmoid, seperti pada Persamaan 1.1 dan Persamaan 1.2. Persamaan 1.1 adalah fungsi aktivasi *binary sigmoid function* dan Persamaan 1.2 adalah fungsi aktivasi *hyperbolic tangent sigmoid* atau *bipolar sigmoid function*.



**Gambar 1.1** Ilustrasi *Multi Layer Perceptron* dengan Jumlah *Layer* Minimal

1. Feedforward Multi Layer Perceptron

Keluaran dari MLP dilakukan dengan metode *feedforward multilayer perceptron*, yang mana keluaran pada *layer* ke-i akan mejadi masukan bagi *layer* ke-i+1 dan seterusnya sampai mencapai *layer* terluar. Persamaan *feedforward* yang digunakan dapat digambarkan pada Persamaan 1.1.1 sebagai keluaran *layer*pertama, Persamaan 1.1.2 untuk keluaran *layer* kedua, dan Persamaan 1.1.3 untuk *layer output* dari MLP pada Gambar 1.1.

Sehingga dari ketiga persamaan didapatkan keluaran dari MLP pada Gambar 1 adalah seperti pada Persamaan 1.1.4,

dengan persamaan *f(v)* yang digunakan tergantung dari jenis fungsi aktivasi yang dipilih antara Persamaan 1.1 dan Persamaan 1.2.

1. Error Back Propagation Algorithm

Metode pembelajaran yang digunakan adalah *error back propagation algorithm* atau EBPA dengan memanfaatkan perbandingan hasil yang diinginkan atau *desired ouput* dengan hasil keluaran model neuron yang sesungguhnya. Informasi tersebut akan digunakan untuk memperaharui nilai *weight* menggunaan Persamaan 2.1.1dan *threshold* dari *output layer* kemudian mundur ke *layer* sebelumnya sampai mencapai *layer* pertama atau *input layer*.

Persamaan 2.1.1 dapat digunakan dengan menggunakan parameter pada Persamaan 2.1.2 dan 2.1.3 sebagai berikut.

Proses pembelajaran dilakukan dengan mengikuti suatu proses sampai nilai *mean square error* atau *sum square error* dari keluaran MLP dibawah suatu nilai yang telah ditentukan sebagai prasyarat penyelesaian pembelajaran atau suatu nilai epsilon () tertentu. Algoritma EBP adalah sebagai berikut:

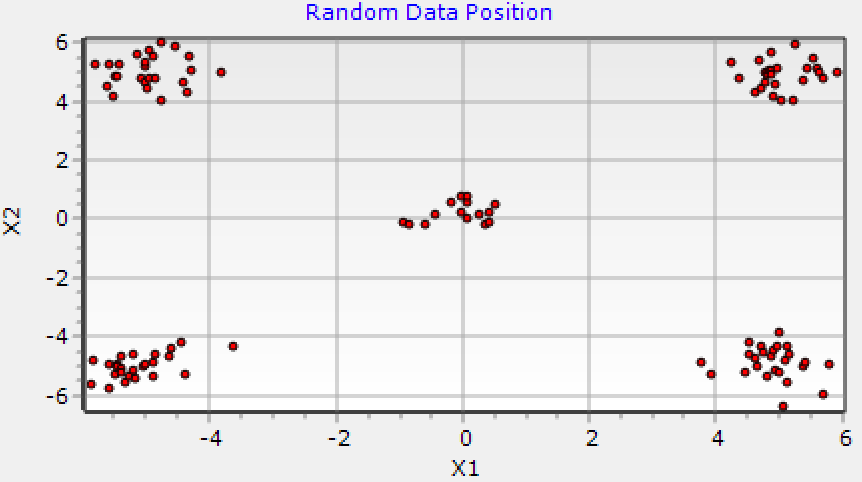
1. Tentukan suatu nilai *random* yang kecil untuk *initial weight* dan *threshold* untuk tiap-tiap koneksi node.
2. Dari data set *input-output,* lakukanperhitungan keluaran dari jaringan *neuron*.
3. Keluaran jaringan *neuron* dibandingkan dengan suatu nilai keluaran yang diinginkan dengan menggunakan Persamaan 2.1.2 dan 2.1.3.
4. Pembaharuan *weight* dan *threshold* dari jaringan dengan Persamaan 2.1.1.
5. Lakukan langkah dua sampai empat berulang sampai nilai *error* yang dihasilkan jaringan kurang dari suatu nilai *error* yang telah ditentukan sebelumnya.

## BAB II

**PERMASALAHAN**

Buatlah suatu program *classifier* untuk data *random* sebanyak 115 data dengan sebaran data seperti pada Gambar 2.1. Program yang dimaksud adalah suatu implementasi MLP dengan metode pembelajaran EBPA secara *offline*. Setelah mendapatkan *weight* dan *threshold* yang sesuai dengan hasil pembelajaran yang diinginkan dengan nilai MSE kurang dari suatu nilai yang ditentukan, maka akan dapat diuji dengan suatu data *random* berupa koordinat dari data tersebut untuk mendapatkan keluaran berupa klasifikasi dari data yang diujikan.

Perhitungan *weight* dan *threshold* ditampilkan dalam bentuk gambar sebagai representasi dari data matriks 2D untuk dapat dibandingkan dengan sebaran nilai *weight* dan *threshold* sebelum dilakukan prose EBPA. Untuk nilai *weight* dan *threshold* yang baru, dilakukan pengujian dengan data *random* yang berbeda dengan data pembelajaran kemudian nilai galat yang dihasilkan jaringan neural yang ada dapat dihitung berdasarkan kombinasi keluaran sesuai yang diinginkan.



**Gambar 2.1**  Sebaran Data Masukan untuk Program *Classifier*

**BAB III**

**METODE**

Model MLP yang digunakan akan terdiri dari total empat (4) *layer,* sehingga memiliki satu pasang *input-output layer* dan dua *hidden layer*. Jumlah node dari *input layer* adalah dua untuk mengakomodasi parameter masukan seperti pada Gambar 2.1 yaitu parameter X1 dan X2 pada diagram *Cartesian*. Sementara desain yang diinginkan untuk keluaran *Y* adalah mengikuti Tabel 2.1, dengan jumlah *output node* atau dilambangkan *j* sebanyak lima (*j* =5) buah, sebagai bagian dari *desired output* yang digunakan untuk proses pembelajaran.

Metode pembelajaran yang akan digunakan adalah EBPA secara *offline,* artinya digunakan suatu algoritma pembelajaran terhadap suatu set data sejumlah 115 data terlebih dahulu untuk menemukan nilai *weight* dan *threshold* yang secara *random* juga ditentukan sebelumnya. Penentuan *weight* dan *threshold* dilakukan menggunakan suatu fungsi *randg*(0,0.1) pada bahasa pemrograman Pascal dengan IDE Delphi 7, yaitu suatu nilai *random* dengan nilai rata-rata 0 dan standar deviasi sebesar 0.1. Penentuan *weight* dan *threshold* setelahnya mengikuti algoritma EBP pada pembahasan Bab I sub-bab B.

Nilai *α* dan *µ* yang digunakan adalah 0,5 dan 2 secara berurutan dengan parameter nilai konvergen MSE yang diharapkan adalah , proses EBPA akan dihentikan jika mencapai nilai konvergen. Pengujian dilakukan untuk menemukan korelasi dari jumlah *node* dari *hidden layer* yang digunakan dan penggunaan data *random* untuk jaringan neural yang sudah melakukan pembelajaran dengan nilai *weight* dan *threshold* baru

**Tabel 2.1** Keluaran yang Diharapkan dari Klasifikasi Data X1 dan X2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Keluaran (*y[j]*)** | **Kelas 1** | **Kelas 2** | **Kelas 3** | **Kelas 4** | **Kelas 5** |
| *y[1]* | 1 | 0 | 0 | 0 | 0 |
| *y[2]* | 0 | 1 | 0 | 0 | 0 |
| *y[3]* | 0 | 0 | 1 | 0 | 0 |
| *y[4]* | 0 | 0 | 0 | 1 | 0 |
| *y[5]* | 0 | 0 | 0 | 0 | 1 |

Pengujian program akan mengikuti poin-poin sebagai berikut:

1. Pengujian dengan data masukan *random* sebanyak 115 data dengan nilai *weight* dan *threshold* yang ditentukan sebelumnya secara *random* dengan jumlah node pada *hidden layer 1* dan *hidden layer 2* berganti dari 8, 32, dan 50 untuk mengukur perbedaan waktu konvergensinya. Hal ini dilakukan untuk mengetahui korelasi dari morfologi jaringan neural terhadap performa dari algoritma pengelompokan dan waktu EBPA yang diperlukan.
2. Dari ketiga kombinasi jumlah node, digunakan nilai jumlah node *hidden layer* sebesar 50 untuk mengukur nilai *error* yang dihasilkan jaringan neural *classifier* terhadap nilai yang diinginkan dari node keluaran, dilakukan proses *sampling* sebanyak 3 data tiap kelasnya.
3. Dilakukan pengujian dengan jumlah node *hidden layer* sebanyak 32 dengan mengganti nilai *α* menjadi 0,8 dan diukur pengaruhnya terhadap waktu yang digunakan untuk proses EBPA.
4. Nilai *weight* dan *threshold* dapat ditampilkan dalam bentuk grafik 2D untuk melihat sebaran nilainya sebelum dan sesudah training. Sampel yang digunakan adalah untuk jumlah node 8 dan 50.

**BAB IV**

**HASIL DAN ANALISIS**

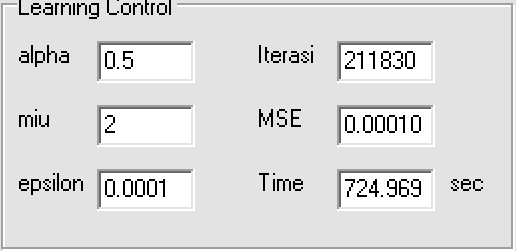
Berdasarkan algoritma EBPA yang digunakan dan program *classifier* yang dibuat, didapatkan hasil keluaran dari jaringan neural yang terdiri dari parameter waktu yang dibutuhkan (iterasi), lima node keluaran dari jaringan neural yang mewakili kelas dari data masukan, pengaruh pergantian nilai *alpha* atau koefisien pembelajaran, dan penggambaran *weight* dan *threshold* dalam gambar 2D.

1. Pengaruh Topologi Jaringan Neural terhadap Waktu yang Dibutuhkan untuk Pemrosesan EBPA

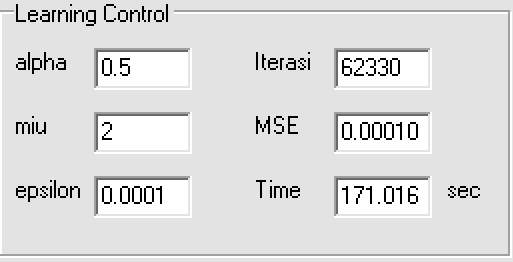
Hasil pengujian menunjukkan secara umum bahwa semakin banyak jumlah node yang digunakan pada *hidden layer 1* dan *hidden layer 2*, maka terjadi penurunan jumlah iterasi yang diperlukan untuk mencapai persyaratan MSE yang telah ditentukan. Hal ini terangkum pada Tabel 4.1 yang menujukkan bahwa iterasi yang dibutuhkan untuk jumlah node sebanyak 8 adalah 339.480 iterasi pada percobaan pertama. Anomali terjadi pada jumlah iterasi percobaan pertama dari penggunan 32 node untuk kedua *hidden layer*, jumlah iterasi yang dibutuhkan untuk menyelesaikan EBPA meningkat dari 339.480 menjadi 537.970 iterasi untuk penggunaan topologi jaringan yang lebih kompleks. Nilai ini kembali turun saat digunakan total 50 node untuk tiap *hidden layer* menjadi 155.480 iterasi. Pada percobaan kedua, didapatkan hasil yang lebih konsisten, yang mana penggunaan jaringan neural yang lebih kompleks memperpendek waktu yang diperlukan untuk proses EBPA, hal ini dapat terjadi karena semakin kompleks suatu jaringan, maka semakin cepat jaringan tersebut mencapai titik konvergensi.

**Tabel 4.1** Pengaruh topologi terhadap waktu pembelajaran

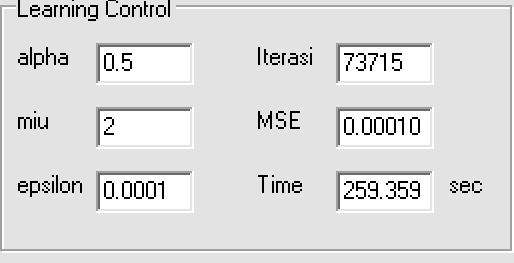
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Jumlah *hidden layer 1*** | **Jumlah *hidden layer 2*** | **Iterasi** | **Waktu yang dibutuhkan (sekon)** |
| 1 | 8 | 8 | 211.830 | 724,696 |
| 2 | 32 | 32 | 62.330 | 171,016 |
| 3 | 50 | 50 | 73.715 | 259,359 |



**Gambar 4.1.1** Jumlah iterasi yang diperlukan untuk penggunaan node sebanyak 8 untuk tiap *hidden layer*



**Gambar 4.1.2** Jumlah iterasi yang diperlukan untuk penggunaan node sebanyak 32 untuk tiap *hidden layer*



**Gambar 4.1.3** Jumlah iterasi yang diperlukan untuk penggunaan node sebanyak 50 untuk tiap *hidden layer*

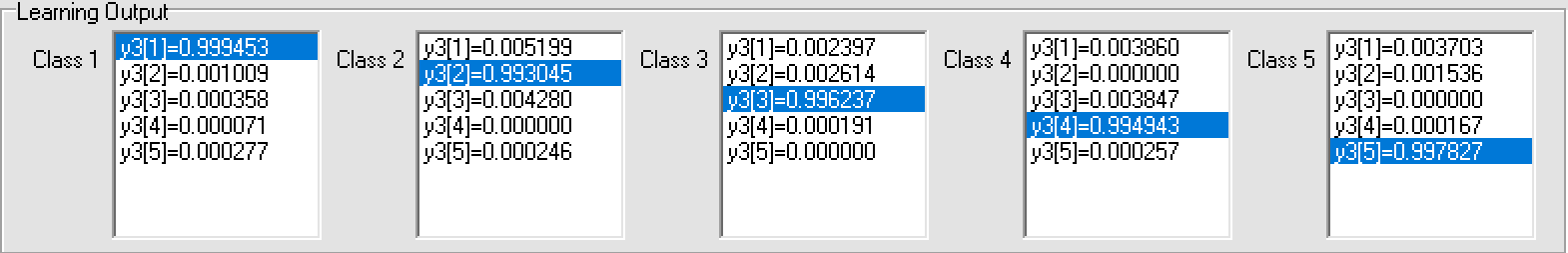
Hasil waktu atau iterasi yang diperlukan untuk EBPA dapat dilihat pada Gambar 4.1.1, 4.1.2, dan 4.1.3 yang dirangkum pada Tabel 4.1 pada bagian percobaan kedua. Anomali yang terjadi pada bagian percobaan pertama dapat diakibatkan oleh penggunaan nilai masukan dan inisialisasi *weight* dan *threshold* yang dilakukan secara *random*; sehingga memiliki parameter awal atau *intial condition* yang berbeda-beda untuk tiap-tiap pembelajaran sehingga tidak selalu mendapatkan nilai iterasi yang sama untuk tiap proses pembelajaran jaringan.

1. Keluaran Jaringan Neural dan Perhitungan Galat

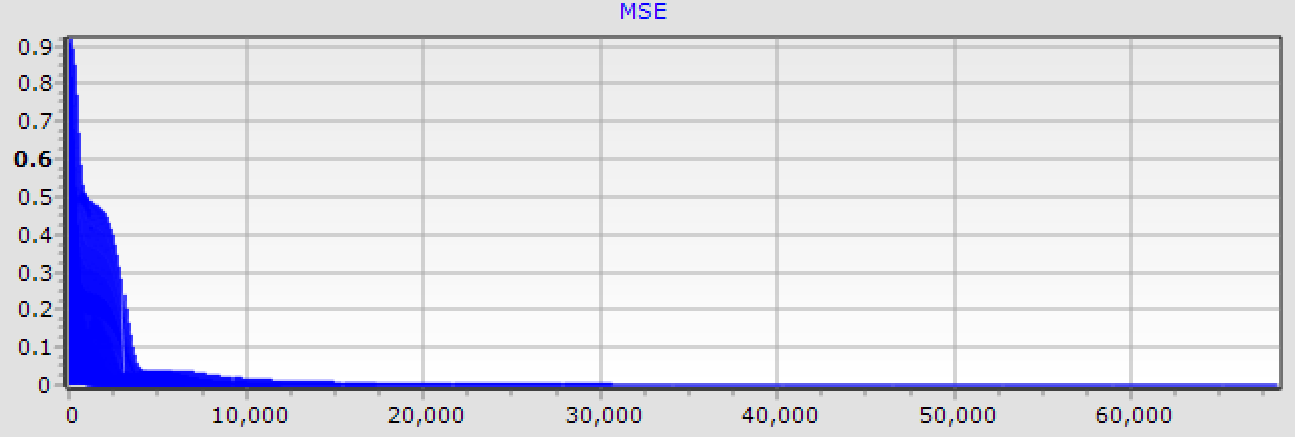
Hasil keluaran jaringan neural untuk keseluruhan kelas pada Gambar 4.2.1 mendekati nilai keluaran yang diinginkan seperti yang dijelaskan pada Tabel 2.1. Hal ini membuktikan jaringan neural yang didesain dengan proses pembelajaran EBPA adalah benar, dengan nilai *mean square error* dari masing-masing node keluaran kurang dari seperti pada Gambar 4.2.2, sehingga didapatkan nilai keluaran dan galat rata-rata dari kelima node keluaran pada Tabel 4.2. Nilai pengujian jaringan dengan data masukan yang bukan data pembelajaran memberikan hasil yang tidak jauh berbeda dengan keluaran jaringan hasil pembelajaran, hal ini membuktikan jaringan dapat mengklasifikasikan data masukan dengan benar sesuai kelas yang ditentukan dalam desain jaringan neural.

**Tabel 4.2** Hasil keluaran jaringan neural pada pengujian jaringan

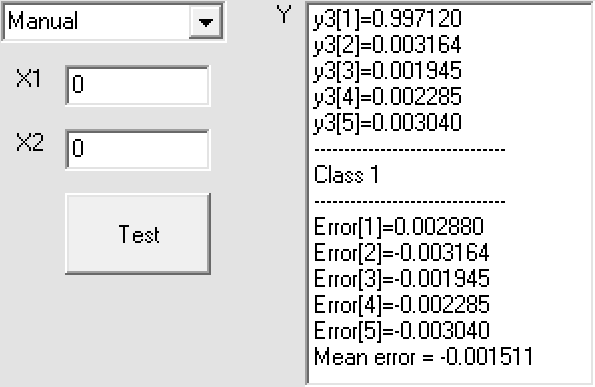
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Kelas** | **X1** | **X2** | **Keluaran** | | | | | **Galat rata-rata** |
| **Y1** | **Y2** | **Y3** | **Y4** | **Y5** |
| 1 | 0 | 0 | 0,9971 | 0,0031 | 0,0019 | 0,0023 | 0,0030 | -0,0015 |
| 0,7169 | -1,5817 | 0,9946 | 0,0020 | 0,0011 | 0,0038 | 0,0061 | -0,0015 |
| 1,3109 | -1,8863 | 0,9889 | 0,0019 | 0,0007 | 0,0045 | 0,0123 | -0,0016 |
| 2 | 5 | 5 | 0,0005 | 0,9970 | 0,0022 | 0 | 0,0012 | -0,0002 |
| 5,3815 | 7,3448 | 0,0084 | 0,9922 | 0,0057 | 0 | 0,0001 | -0,0013 |
| 7.7161 | 2,6159 | 0,0013 | 0,9244 | 0,0021 | 0 | 0,0892 | -0,0029 |
| 3 | -5 | 5 | 0,0011 | 0,0018 | 0,9977 | 0,0004 | 0 | -0,0002 |
| -7,1511 | 5,8303 | 0,0003 | 0,0003 | 0,9973 | 0,0059 | 0 | -0,0008 |
| -9,5892 | 7,6524 | 0,0002 | 0,0002 | 0,9974 | 0,0128 | 0 | -0,0021 |
| 4 | -5 | -5 | 0,0049 | 0 | 0,0016 | 0,9953 | 0,0005 | -0,0005 |
| -3,2984 | -4,6988 | 0,0138 | 0 | 0,0001 | 0,9886 | 0,0036 | -0,0012 |
| -2,4719 | -3,0915 | 0,0038 | 0 | 0,0003 | 0,9665 | 0,0017 | -0,0013 |
| 5 | 5 | -5 | 0,0034 | 0,0013 | 0 | 0,0002 | 0,9981 | -0,0006 |
| 5,4015 | -6,1197 | 0,0032 | 0,0006 | 0 | 0,0004 | 0,9988 | -0,0009 |
| 4,1638 | -4,6501 | 0,0058 | 0,0007 | 0 | 0,0003 | 0,9897 | -0,0008 |



**Gambar 4.2.1** Keluaran jaringan neural pada proses pembelajaran EBPA

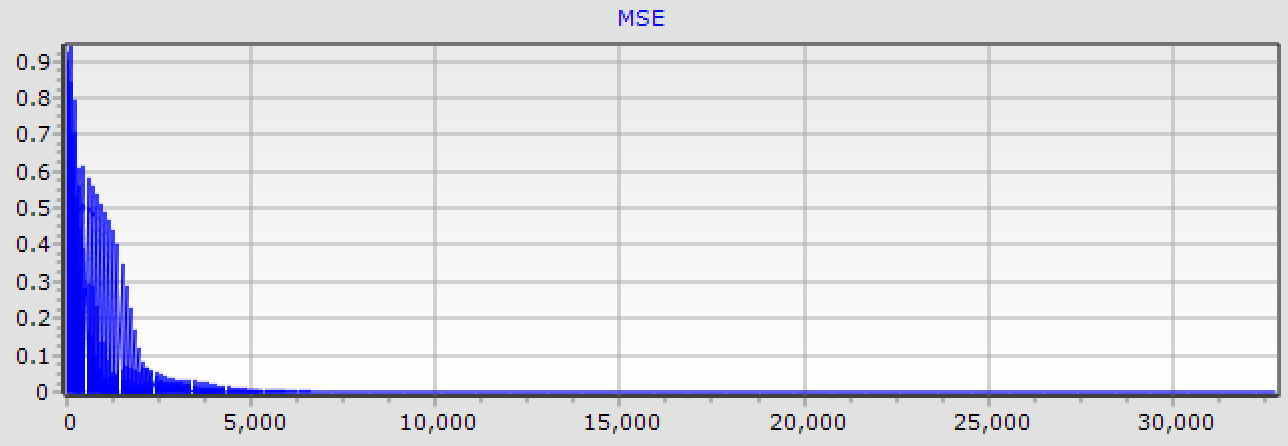


**Gambar 4.2.2** Grafik *error* hasil pembelajaran EBPA

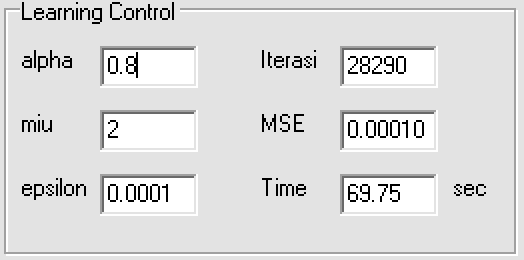


**Gambar 4.2.3** Keluaran jaringan neuron untuk pengujian data masukan kelas 1 (0,0)

Keluaran jaringan neuron ditampilkan dalam bentuk keluaran tiap node, galat tiap node, kelas dari data masukan, dan rata-rata galat yang dihitung berdasarkan galat tiap node seperti pada Gambar 4.2.3 yang menunjukkan data masukan X1,X2 bernilai 0,0 yang merupakan data kelas 1, sehingga keluaran jaringan menunjukkan nilai mendekati satu pada node pertama dan mendekati nol pada node lainnya dari kelima node keluaran. Tabel 4.2 merangkum keseluruhan data percobaan dari jaringan neural dan didapatkan hasil keluaran yang memiliki karakteristik sesuai dengan keluaran yang diinginkan dan memiliki nilai rata-rata galat total sebesar -0,00164, sesuai dengan parameter MSE yang merupakan kuadarat dari perhitungan galat sebesar . Hal ini mendukung bahwa algoritma EBPA dan jaringan neural yang digunakan untuk permasalahan *classifier* adalah benar dan dapat bekerja sesuai dengan parameter yang telah ditentukan sebelumnya. Ketika digunakan topologi jaringan yang berbeda, yaitu dengan 32 dan 50 node, keluaran galat total yang didapatkan tidak sebanding dengan kompleksitas topologi jaringan. Total galat dari jaringan ber-node 8 lebih kecil dari jaringan dengan jumlah node 32, sementara jaringan dengan node 50 memiliki total galat paling kecil dari ketiganya (Lih. Tabel B.3.).



**Gambar 4.3.1** Grafik *error* untuk penggunaan nilai alpha 0,8



**Gambar 4.3.2** Parameter pembelajaran dengan nilai alpha 0,8

1. Pengaruh Penggantian Koefisien Pembelajaran

Pengaruh koefisien pembelajaran atau *alpha* berbanding terbalik terhadap waktu yang dibutuhkan untuk proses pembelajaran EBPA, hal ini dikarenakan nilai koefisien sebagai pengali dalam proses pembaharuan *weight* dan *threshold*, sehingga semakin besar nilai *alpha*, maka semakin cepat nilai *weight* dan *threshold* berubah menyesuaikan dengan MSE untuk menapai titik konvergensi lebih cepat dari penggunaan nilai koefisien yang lebih rendah.

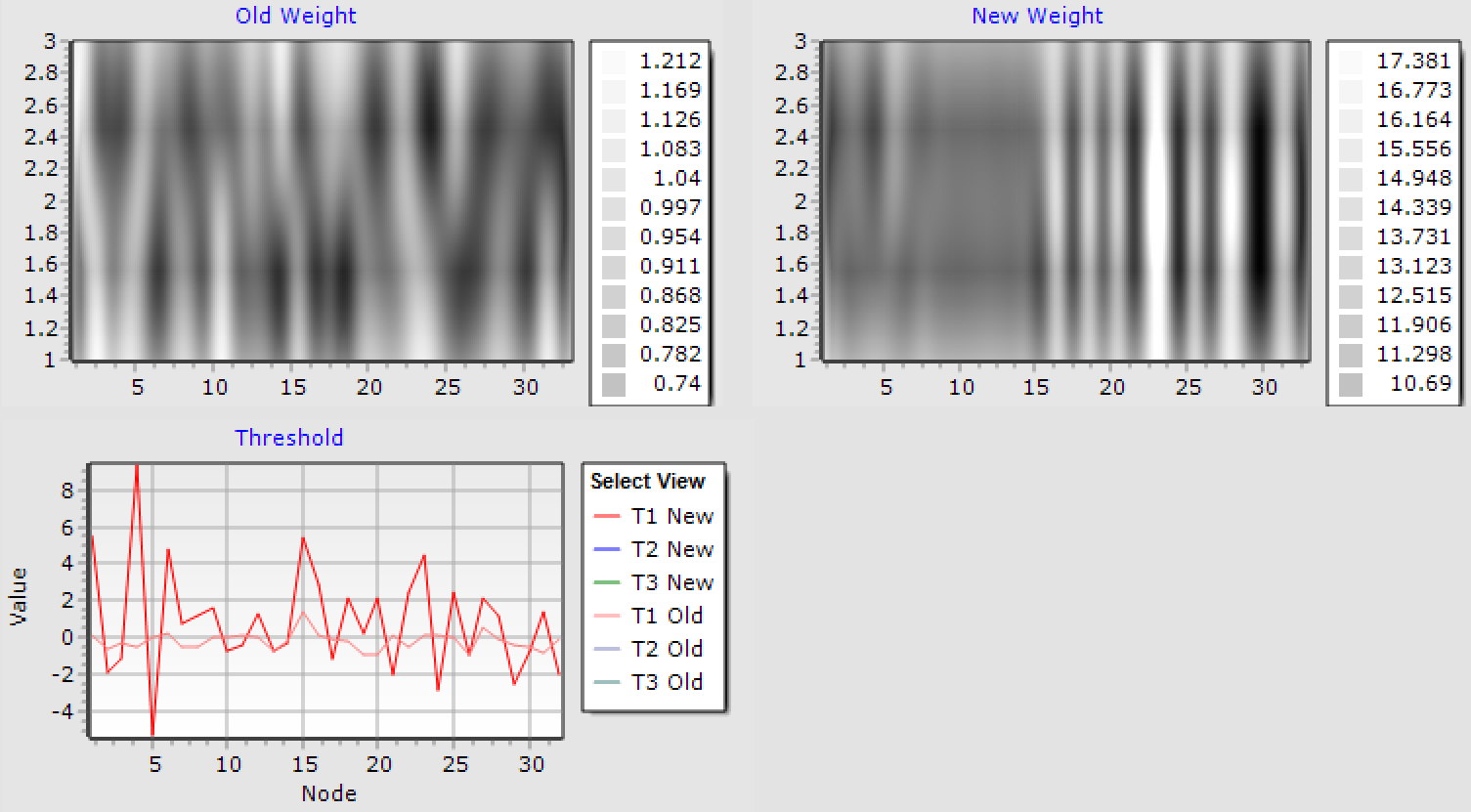
Grafik MSE pada penggunaan nilai alpha 0,8 tampak seperti Gambar 4.3.1 yang mana terjadi penurunan MSE yang lebih cepat daripada penggunaan nilai alpha 0,5; diakibatkan oleh perubahan nilai *weight* dan *threshold* yang lebih cepat. Jumlah iterasi seperti pada Gambar 4.3.2 untuk proses pembelajaran dengan nilai *alpha* sebesar 0,8 juga 54,61% lebih sedikit dibandingkan dengan jumlah iterasi pada nilai *alpha* 0,5 seperti pada Tabel 4.1. Waktu yang dibutuhkan untuk menyelesaikan pembelajaran dengan nilai *alpha* 0,8 hanya 69,75 sekon, dibandingkan ketika menggunakan *alpha* 0,5 yang membutuhkan total waktu 171,016 detik.

1. Penggambaran Parameter *Weight* dan *Threshold*

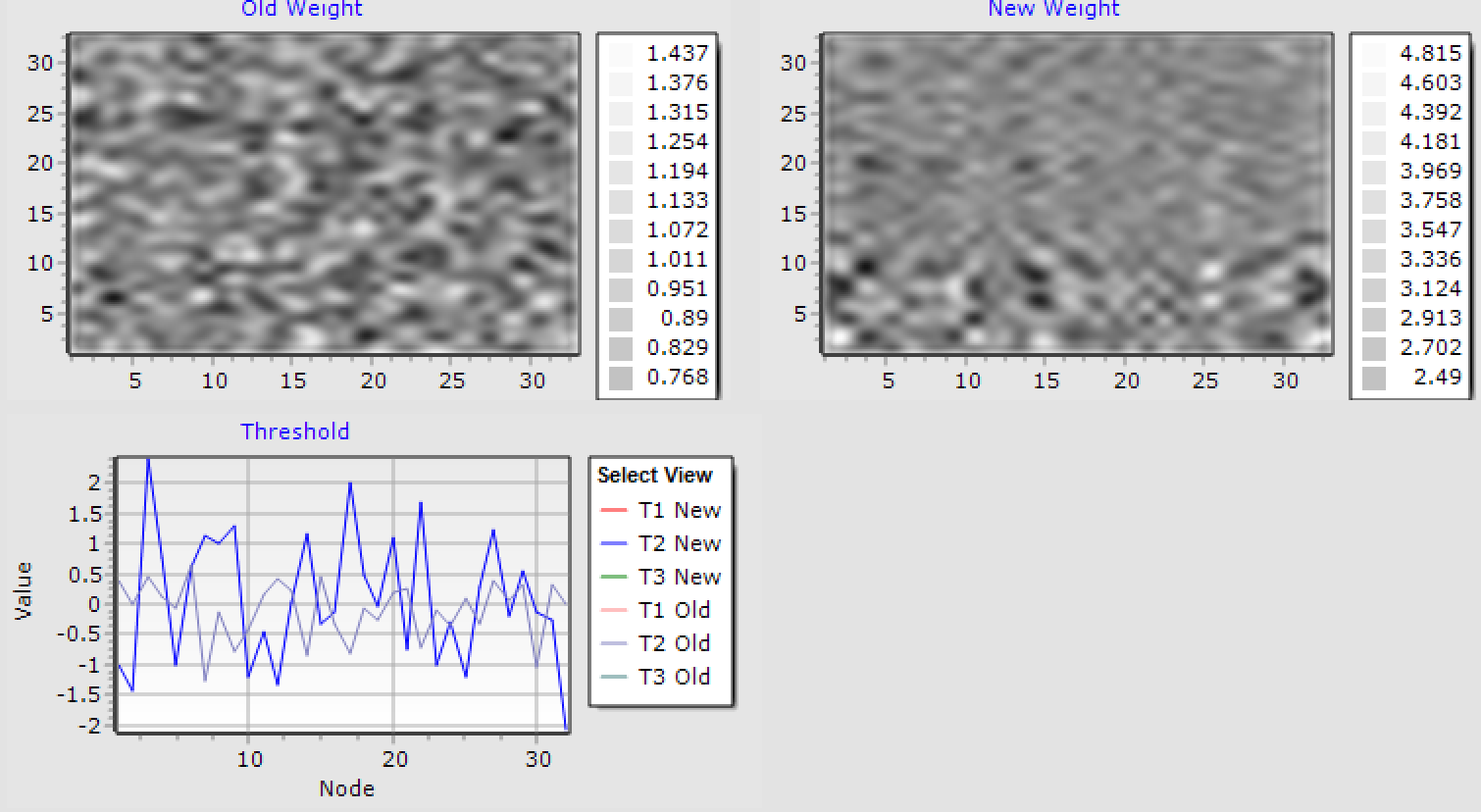
Nilai *weight* yang digunakan merupakan matriks dua dimensi yang menyesuaikan dengan jumlah node pada *layer* asal dan node pada *layer* tujuan; sementara nilai *threshold* adalah matriks satu dimensi yang bergantung pada jumlah node pada *layer* yang bersangkutan. Secara topologi, jaringan neural yang digunakan memiliki empat *layer*, yaitu *layer* masukan dengan dua node, dua buah *hidden layer* dengan jumlah node yang dapat diatur, dan *layer* keluaran dengan lima node. *Weight* dan *threshold* yang menghubungkan *layer* masukan ke *hidden layer* pertama diberi nama W1 dan T1, sementara *weight* dan *threshold* yang menghubungkan *hidden layer* pertama ke *hidden layer* kedua diberi nama W2 dan T2, dan sementara *weight* dan *threshold* yang menghubungkan *hidden layer* kedua ke *layer* keluaran diberi nama W3 dan T3. Perubahan dari nilai *weight* dan *threshold* sebelum dan sesudah pembelajaran dapat diamati kesesuaiannya dan ditampilkan seperti pada Gambar 4.4.1 untuk W1 dan T1, Gambar 4.4.2 untuk W2 dan T2, dan Gambar 4.4.3 untuk W3 dan T3 untuk jumlah node *hidden layer* sebanyak 32.

Pada perbandingan nilai W1 dan T1, didapatkan suatu matriks berukuran 2x32, merepresentasikan jumlah node *layer* masukan dan *hidden layer* pertama yang diatur memiliki 32 node. Nilai *weight* berubah sesuai representasi warna yang ada pada Gambar 4.4.1 dan perubahan nilai *threshold* tampak pada bagian bawah kiri gambar, yang menunjukkan perbedaan morfologi sebelum dan sesudah pembelajaran. Terlihat komponen *weight* mengalami perubahan menjadi lebih teratur seperti pada Gambar 4.4.1, dengan representasi matriks *weight* yang memiliki suatu pola terang dan gelap secara bergantian.

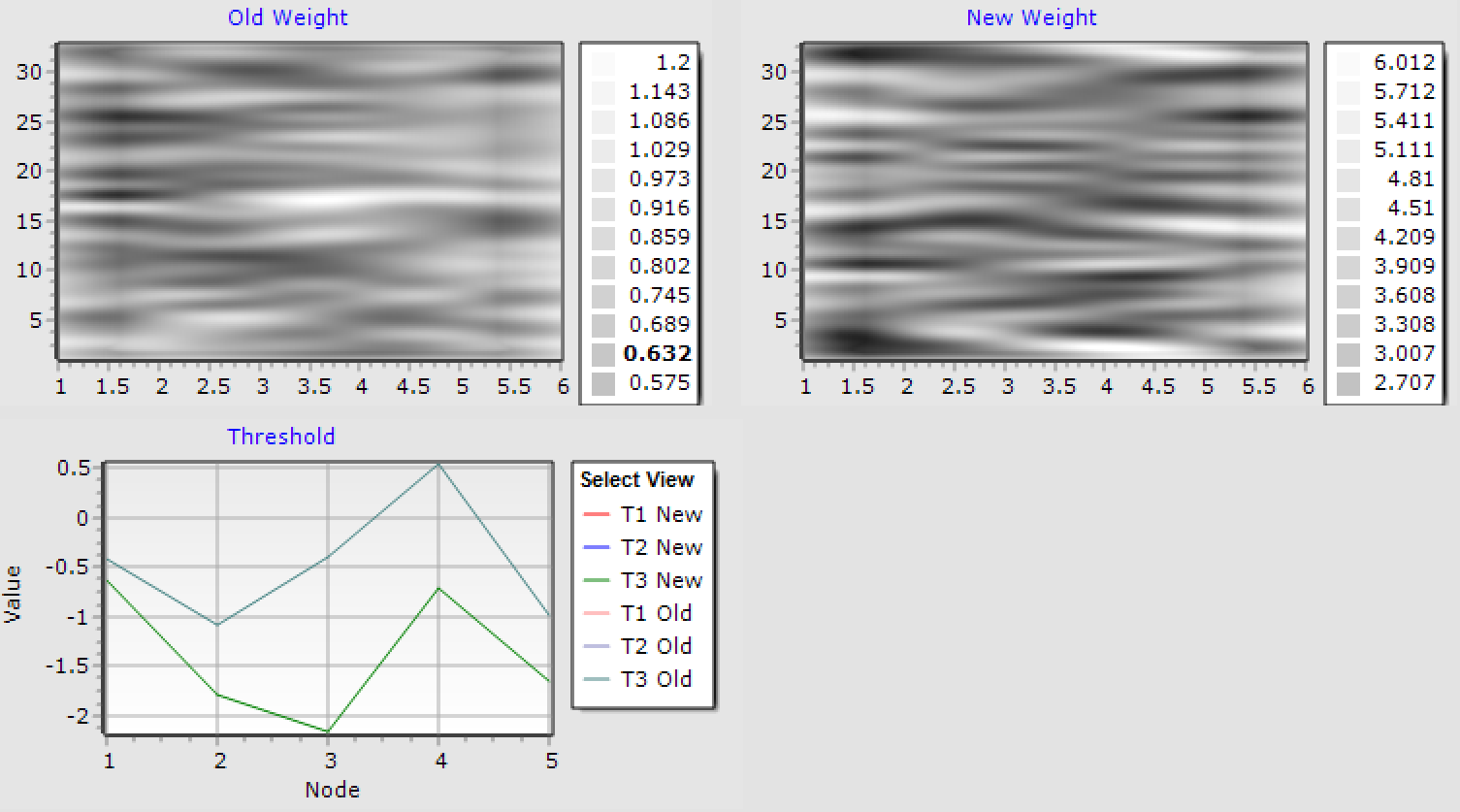
Pada Gambar 4.4.2 didapatkan perbandingan nilai W2 dan T2 sebelum dan sesudah pembelajaran dengan ukuran matriks W2 sebesar 32x32, merepresentasikan jumlah node *hidden layer* pertama dan kedua. Ukuran *array* dari threshold adalah sesuai dengan jumlah node *layer* yang yaitu sebanyak 32 buah node, seperti pada Gambar 4.4.2.



**Gambar 4.4.1** Nilai W1 dan T1 sebelum dan sesudah pembelajaran



**Gambar 4.4.2** Nilai W2 dan T2 sebelum dan sesudah pembelajaran



**Gambar 4.4.3** Nilai W3 dan T3 sebelum dan sesudah pembelajaran

Sementara untuk representasi W3 dan T3 ditunjukkan pada Gambar 4.4.3 dengan representasi nilai *weight* pada matriks berukuran 5x32 dan representasi nilai *threshold* sebanyak 5 node sesuai jumlah node pada *layer* keluaran.

**BAB V**

**KESIMPULAN**

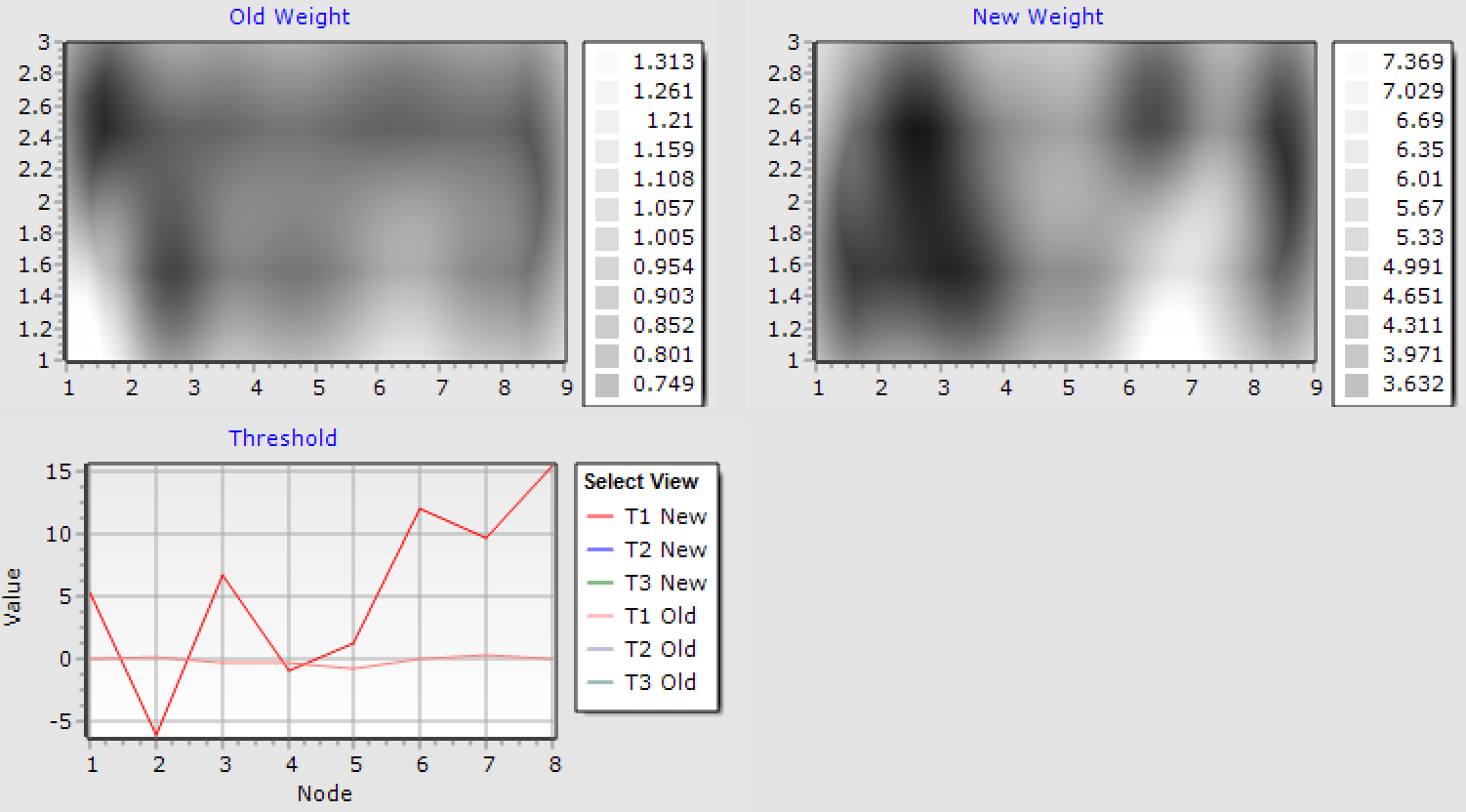
Berdasarkan pembahasan dan analisis tersebut, maka dapat diambil kesimpulan mengenai efek topologi jaringan pada proses pembelajaran EBPA, performa dari jaringan neural *classifier* untuk mengklasifikasikan data koordinat X1 dan X2, pengaruh dari koefisien pembelajaran, dan penggambaran nilai *weight* dan *threshold* dalam bentuk grafik.

Semakin kompleks dan besar topologi jaringan *neural* yang digunakan, maka semakin kompleks sistem ANN yang dihasilkan. Jumlah iterasi yang merepresentasikan waktu pmebelajaran untuk mencapai nilai konvergensi MSE yang ditentukan lebih cepat pada jaringan yang memiliki topologi yang lebih kompleks, karena sistem ANN yang lebih kompleks memiliki performa yang lebih baik daripada ANN yang memiliki topologi lebih sederhana. Hal ini tampak pada berkurangnya nilai iterasi dan waktu yang dibutuhkan untuk proeses *learning* seiring meningkatnya jumlah *node* untuk tiap-tiap *hidden layer* yang digunakan. Selain itu, koefisien pembelajaran yang semakin besar mengakibatkan perubahan nilai *weight* dan *threshold*, sehingga mampu mengurangi baik jumlah iterasi maupun waktu yang dibutuhkan untuk menyelesaikan proses pembelajaran.

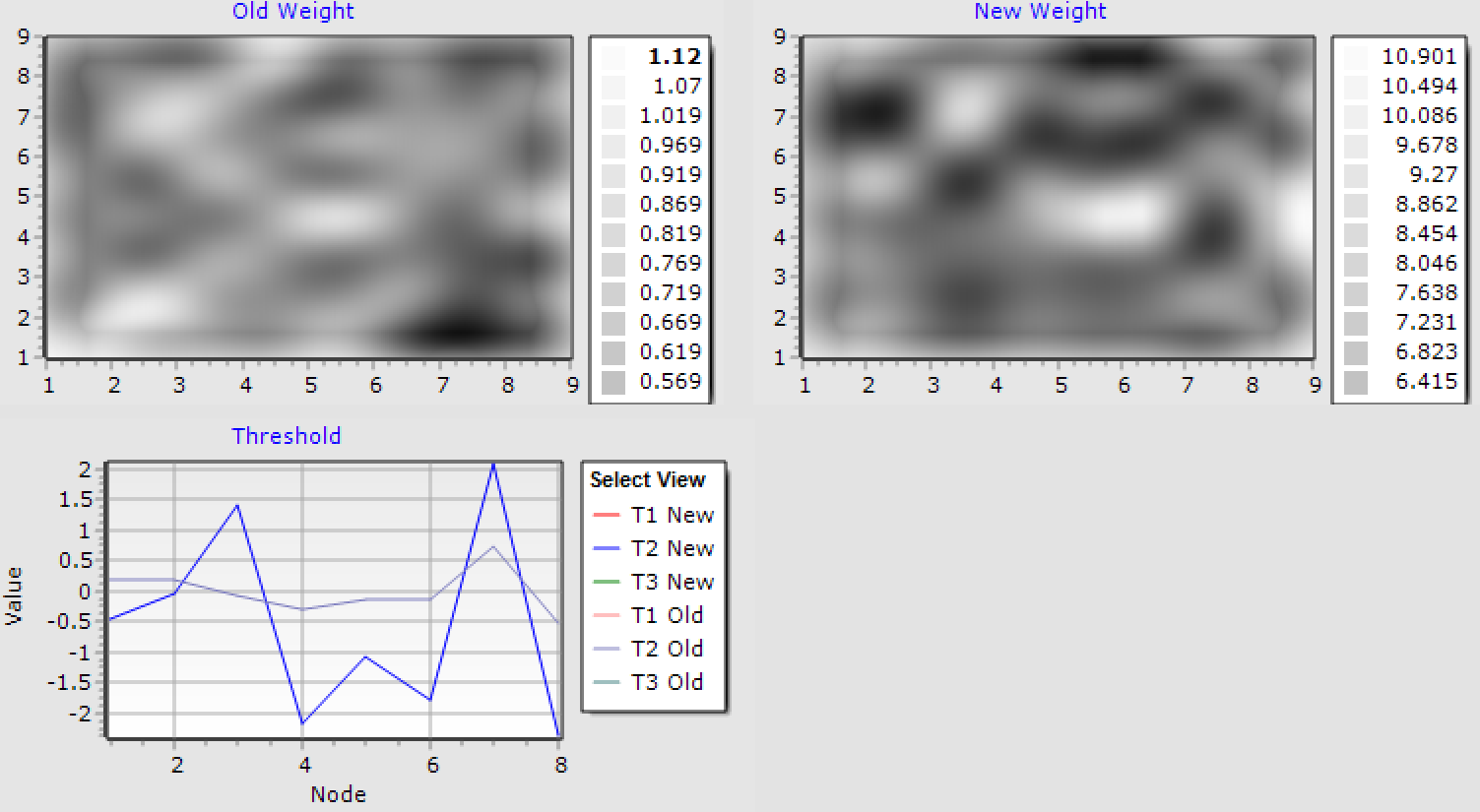
Performa dari jaringan neural diuji menggunakan data diluar data pembelajaran dan didapatkan hasil rata-rata galat total sebesar 0,00164 untuk kelima node dengan nilai MSE dari kelima node 0,00000269 atau sebesar . Hal ini menunjukkan sistem ANN dapat memberikan keluaran pada node-node *layer* keluaran sesuai yang telah ditetapkan sebelumnya dalam *desired output*. Nilai galat dapat diturunkan dengan menurunkan parameter konvergen MSE minimal yang harus dicapai, namun waktu yang dibutuhkan untuk pembelajaran akan semakin lama. Pengaruh topologi jaringan tidak menunjukkan perbedaan yang signifikan dari galat keluaran ANN; semua keluaran dari hasil pengujian masih dalam batas nilai galat maksimal.

Topologi *weight* dan *threshold* yang digambarkan dalam bentuk grafik adalah seperti pada Gambar 4.4.1, 4.4.2, dan 4.4.3 dengan perbedaan yang Nampak bahwa terjadi perubahan pada nilai *weight* (dalam bidang 2D) dan *threshold* dalam bidang 1D.

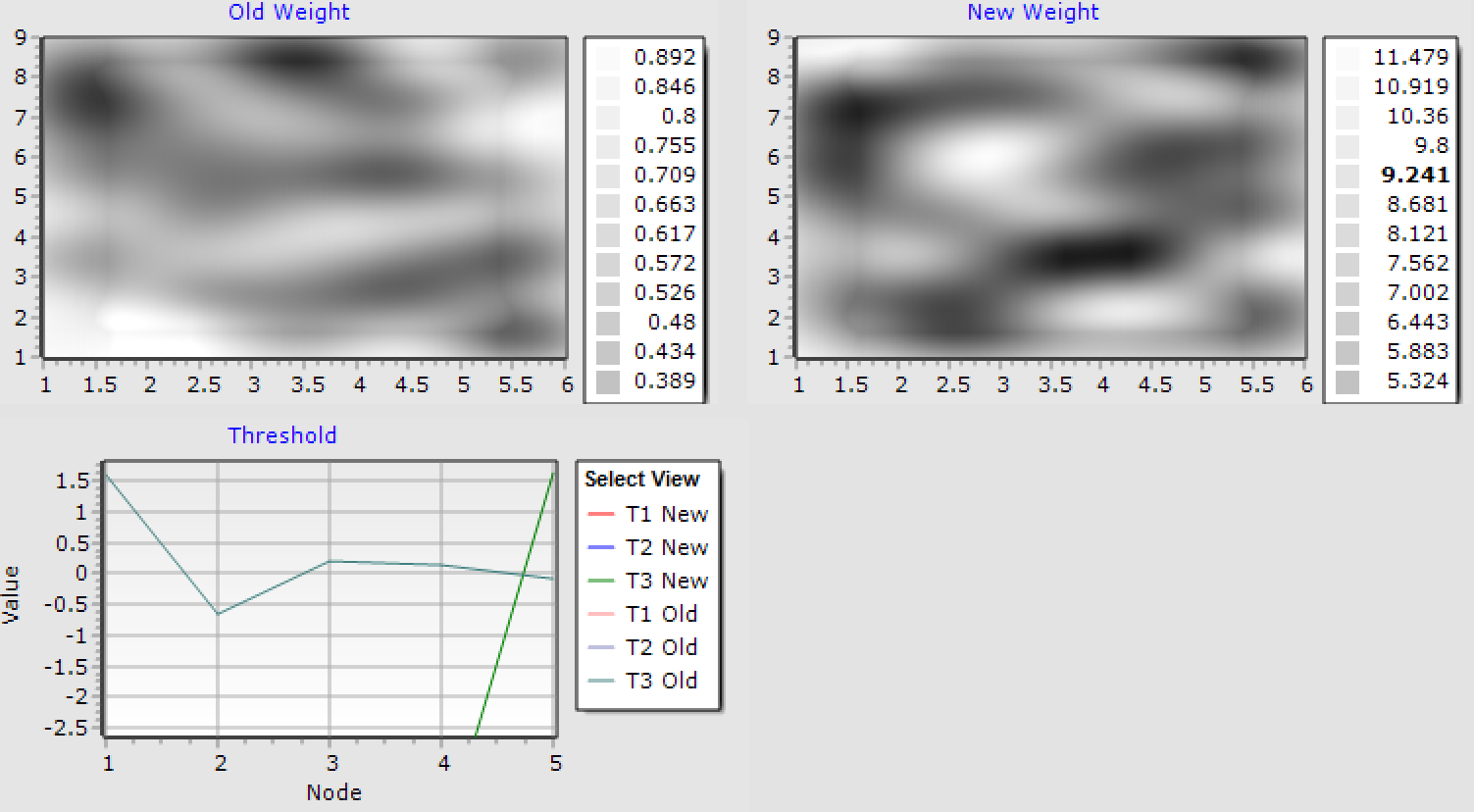
## LAMPIRAN



**Gambar A.1** Nilai W1 dan T1 sebelum dan sesudah pembelajaran untuk 8 node



**Gambar A.2** Nilai W2 dan T2 sebelum dan sesudah pembelajaran untuk 8 node



**Gambar A.3** Nilai W3 dan T3 sebelum dan sesudah pembelajaran untuk 8 node

**Tabel B.1** Galat untuk penggunaan 32 node pada *hidden layer 1* dan *hidden layer 2*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Kelas** | **X1** | **X2** | **Keluaran** | | | | | **Galat rata-rata** |
| **Y1** | **Y2** | **Y3** | **Y4** | **Y5** |
| 1 | 0 | 0 | 0,9993 | 0 | 0 | 0,0003 | 0,0009 | -0,0013 |
| 0,7169 | -1,5817 | 0,9412 | 0 | 0 | 0,0084 | 0,0296 | -0,0038 |
| 1,3109 | -1,8863 | 0,9889 | 0,0019 | 0,0007 | 0,0045 | 0,0123 | -0,0061 |
| 2 | 5 | 5 | 0,0005 | 0,9970 | 0,0022 | 0 | 0,0012 | -0,0031 |
| 5,5091 | 4,1244 | 0,0008 | 0,9987 | 0,0002 | 0 | 0 | -0,0023 |
| 3,3389 | 2,2193 | 0,0048 | 0,9943 | 0 | 0 | 0,0009 | -0,0028 |
| 3 | -5 | 5 | 0,0011 | 0,0018 | 0,9977 | 0,0004 | 0 | -0,0002 |
| -7,6886 | 8,6976 | 0,0012 | 0,0003 | 0,9990 | 0 | 0 | -0,0025 |
| -4,7628 | 3,8314 | 0,0008 | 0 | 0,9957 | 0,0094 | 0 | -0,0029 |
| 4 | -5 | -5 | 0,0009 | 0 | 0,0018 | 0,9989 | 0 | -0,0007 |
| -7,7663 | -5,7238 | 0,0002 | 0 | 0,0002 | 0,9996 | 0 | -0,0003 |
| -4,2126 | -3,9882 | 0,0126 | 0 | 0,0003 | 0,9858 | 0 | -0,0001 |
| 5 | 5 | -5 | 0,0019 | 0,0029 | 0 | 0 | 0,9982 | -0,0013 |
| 3,9337 | -4,6211 | 0,0036 | 0 | 0 | 0,0003 | 0,9986 | -0,0005 |
| 6,4906 | -7,5807 | 0,0016 | 0 | 0 | 0,0003 | 0,9993 | -0,0002 |
| **Galat total** | | | | | | | | -0,0019 |

**Tabel B.2** Galat untuk penggunaan 50 node pada *hidden layer 1* dan *hidden layer 2*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Kelas** | **X1** | **X2** | **Keluaran** | | | | | **Galat rata-rata** |
| **Y1** | **Y2** | **Y3** | **Y4** | **Y5** |
| 1 | 0 | 0 | 0,9993 | 0,0031 | 0,0019 | 0,0023 | 0,0030 | -0,0022 |
| -1,6862 | -0,2834 | 0,9812 | 0 | 0,0066 | 0,0136 | 0 | -0,0003 |
| -0,8478 | -0,8193 | 0,9144 | 0 | 0,0021 | 0,1181 | 0,0001 | -0,0069 |
| 2 | 5 | 5 | 0 | 0,9989 | 0,0009 | 0 | 0,0006 | -0,0005 |
| 5,3815 | 7,3448 | 0,0084 | 0,9922 | 0,0057 | 0 | 0,0001 | -0,0013 |
| 9,3636 | 6,6832 | 0 | 0,9991 | 0,0005 | 0 | 0,0026 | -0,0004 |
| 3 | -5 | 5 | 0,0014 | 0,0003 | 0,9973 | 0,0002 | 0 | -0,0009 |
| -8,0938 | 7,6623 | 0,0014 | 0,0002 | 0,9979 | 0,0002 | 0 | -0,0007 |
| -2,4146 | 3,5708 | 0,0209 | 0,0084 | 0,9609 | 0 | 0 | -0,0019 |
| 4 | -5 | -5 | 0,0009 | 0 | 0,0018 | 0,9989 | 0 | -0,0007 |
| -5,1228 | -6,9705 | 0,0104 | 0 | 0,0012 | 0,9867 | 0,0004 | -0,0003 |
| -4,5879 | -5,8058 | 0,0060 | 0 | 0,0012 | 0,9936 | 0,0002 | -0,0002 |
| 5 | 5 | -5 | 0,0019 | 0,0029 | 0 | 0 | 0,9982 | -0,0013 |
| 5,4015 | -6,1197 | 0,0032 | 0,0006 | 0 | 0,0004 | 0,9988 | -0,0009 |
| 5,0330 | -4,4977 | 0,0029 | 0,0025 | 0 | 0 | 0,9971 | -0,0005 |
| **Galat total** | | | | | | | | -0,0013 |

## Tabel B.3 Perbandingan galat rata-rata dengan topologi jaringan

|  |  |
| --- | --- |
| Topologi Jaringan (Jumlah node *hidden layer-*1dan 2) | Galat rata-rata |
| 8 | -0.0016 |
| 32 | -0.0019 |
| 50 | -0.0013 |

## Lampiran C.1: Prosedur algoritma pembelajaran *EBPA*:

procedure TForm1.learning();

var

h,i,j,k,l,q,n,m,c,waktu:integer;

begin

Label6.Caption:='Status: Learning...';

ndata:=115;

hlayer1node:=strtoint(edit2.Text);

hlayer2node:=strtoint(edit3.text);

kelas:=5;

//mulai training

for i:=1 to ndata do

begin

//forward layer 1

for j:=1 to hlayer1node do begin

tempx:=0; tempy:=0;

tempx:=tempx+x1[i]\*w1[j,1]; //q input pattern

tempy:=tempy+x2[i]\*w1[j,2]; //disini ganti

v1[j]:= tempx+tempy+th1[j];

y1[j]:= 1/(1+exp(-alpha\*v1[j]));

g1[j]:= alpha\*y1[j]\*(1-y1[j]);

end;

//layer 2

for k:=1 to hlayer2node do begin

temp:=0;

for j:=1 to hlayer1node do begin

temp:=temp+y1[j]\*w2[k,j];

end;

v2[k]:= temp+th2[k];

y2[k]:= 1/(1+exp(-alpha\*v2[k]));

g2[k]:= alpha\*y2[k]\*(1-y2[k]);

end;

//layer 3

for l:=1 to kelas do begin

temp:=0;

for k:=1 to hlayer2node do begin

temp:=temp+y2[k]\*w3[l,k];

end;

v3[l]:= temp+th3[l];

y3[l]:= 1/(1+exp(-alpha\*v3[l]));

g3[l]:= alpha\*y3[l]\*(1-y3[l]);

end;

//backward

if (i<=15) then q:=1;

if ((i>15) and (l<=40)) then q:=2;

if ((i>40) and (l<=65)) then q:=3;

if ((i>65) and (l<=90)) then q:=4;

if ((i>90) and (l<=115)) then q:=5;

//menghitung error

ekuadrat:=0;

for l:=1 to kelas do begin

er[l]:=d[l,q]-y3[l];

ekuadrat:=ekuadrat+sqr(er[l]);

end;

error[q]:=0.5\*ekuadrat;

series2.AddXY(iterasi,error[q]);

//menghitung delta error

//delta error layer3

for l:=1 to kelas do begin

delta3[l]:= (d[l,q]-y3[l])\*g3[l];

end;

//delta error layer2

for k:=1 to hlayer2node do begin

temp:=0;

for l:=1 to kelas do begin

temp:=temp+delta3[l]\*w3[l,k];

end;

delta2[k]:= temp\*g2[k];

end;

//delta error layer1

for j:=1 to hlayer1node do begin

temp:=0;

for k:=1 to hlayer2node do begin

temp:=temp+delta2[k]\*w2[k,j];

end;

delta1[j]:= temp\*g1[j];

end;

//update w dan th layer 3

for l:= 1 to kelas do begin

for k:= 1 to hlayer2node do begin

wt3[l,k]:=w3[l,k];

w3[l,k]:=w3[l,k]+miu\*delta3[l]\*y2[k];

end;

tht3[l]:=th3[l];

th3[l]:=th3[l]+miu\*delta3[l];

end;

//update w dan th layer 2

for k:= 1 to hlayer2node do begin

for j:= 1 to hlayer1node do begin

wt2[k,j]:=w2[k,j];

w2[k,j]:=w2[k,j]+miu\*delta2[k]\*y1[j];

end;

tht2[k]:=th2[k];

th2[k]:=th2[k]+miu\*delta2[k];

end;

//update w dan th layer 1

for j:= 1 to hlayer1node do begin

wt1[j,1]:=w1[j,1];

wt1[j,2]:=w1[j,2];

w1[j,1]:=w1[j,1]+miu\*delta1[j]\*x1[j];

w1[j,2]:=w1[j,2]+miu\*delta1[j]\*x2[j]; //disini ganti

tht1[j]:=th1[j];

th1[j]:=th1[j]+miu\*delta1[j];

end;

end;

//akhir training

//mengecek output y3[l,q] menggunakan w dan th terakhir

for i:=1 to ndata do begin

//forward

//menghitung output layer 1

for j:=1 to hlayer1node do begin

tempx:=0; tempy:=0;

tempx:=tempx+x1[i]\*wt1[j,1];

tempy:=tempy+x2[i]\*wt1[j,2]; //q input pattern

v1[j]:= tempx+tempy+tht1[j];

y1[j]:= 1/(1+exp(-alpha\*v1[j]));

end;

//menghitung output layer 2

for k:=1 to hlayer2node do begin

temp:=0;

for j:=1 to hlayer1node do begin

temp:=temp+y1[j]\*wt2[k,j];

end;

v2[k]:= temp+tht2[k];

y2[k]:= 1/(1+exp(-alpha\*v2[k]));

g2[k]:= alpha\*y2[k]\*(1-y2[k]);

end;

if (i<=15) then q:=1;

if ((i>15) and (l<=40)) then q:=2;

if ((i>40) and (l<=65)) then q:=3;

if ((i>65) and (l<=90)) then q:=4;

if ((i>90) and (l<=115)) then q:=5;

listbox6.Clear;

listbox7.Clear;

listbox8.Clear;

listbox9.Clear;

listbox10.Clear;

//menghitung output layer 3

for l:=1 to kelas do begin

temp:=0;

for k:=1 to hlayer2node do begin

temp:=temp+y2[k]\*wt3[l,k];

end;

v3[l]:= temp+tht3[l];

yy3[l,q]:= 1/(1+exp(-alpha\*v3[l]));

listbox6.Items.Add('y3['+inttostr(l)+']='+format('%.6f',[yy3[l,1]]));

listbox7.Items.Add('y3['+inttostr(l)+']='+format('%.6f',[yy3[l,2]]));

listbox8.Items.Add('y3['+inttostr(l)+']='+format('%.6f',[yy3[l,3]]));

listbox9.Items.Add('y3['+inttostr(l)+']='+format('%.6f',[yy3[l,4]]));

listbox10.Items.Add('y3['+inttostr(l)+']='+format('%.6f',[yy3[l,5]]));

end;

iterasi:=iterasi+1;

edit7.Text:=inttostr(iterasi);

end;

//menghitung mse total

mse\_total:=0;

for q:=1 to 5 do begin

mse\_total:=mse\_total+error[q];

end;

stop:=GetTickCount;

elapsed:=stop-start;

edit11.Text:=floattostr(elapsed/1000);

//series1.AddXY(iterasi,mse\_total);

edit8.Text:=format('%.5f',[mse\_total]);

Label6.Caption:='Status: Learning...';

//konvergen??

if mse\_total<epsilon then begin

//timer1.Enabled:=false;

Windows.Beep(1000,1000);

Speedbutton1.Caption:='Start';

Label6.Caption:='Status: Learning done! Displaying results!';

displayweightthres();

delay(2000);

Label6.Caption:='Status: Updated W and Th can be used! Test it.';

for j:=1 to hlayer1node do begin

for i:=1 to 2 do begin

w1[j,i]:=wt1[j,i];

//series4.AddXYZ(j,i,w1[j,i]);

end;

end;

for j:=1 to hlayer1node do begin

th1[j]:=tht1[j];

end;

for k:=1 to hlayer2node do begin

for j:=1 to hlayer1node do begin

w2[k,j]:=wt2[k,j];

//series7.AddXYZ(k,j,w2[j,i]);

end;

end;

for k:=1 to hlayer2node do begin

th2[k]:=tht2[k];

end;

for l:=1 to kelas do begin

for k:= 1 to hlayer2node do begin

w3[l,k]:=wt3[l,k];

//series8.AddXYZ(k,j,w3[j,i]);

end;

end;

for l:=1 to kelas do begin

th3[l]:=tht3[l];

end;

end;

end;

**Lampiran C.2: Fungsi untuk melakukan pengujian *clustering* dengan ANN MLP:**

function TForm1.mlp(input1,input2:double):arr;

var

pointer,j,k,l:integer;

begin

//forward

//menghitung output layer 1

for j:=1 to hlayer1node do begin

tempx:=0; tempy:=0;

tempx:=tempx+input1\*w1[j,1];

tempy:=tempy+input2\*w1[j,2]; //q input pattern

v1[j]:= tempx+tempy+th1[j];

y1[j]:= 1/(1+exp(-alpha\*v1[j]));

end;

//menghitung output layer 2

for k:=1 to hlayer2node do begin

temp:=0;

for j:=1 to hlayer1node do begin

temp:=temp+y1[j]\*w2[k,j];

end;

v2[k]:= temp+th2[k];

y2[k]:= 1/(1+exp(-alpha\*v2[k]));

g2[k]:= alpha\*y2[k]\*(1-y2[k]);

end;

listbox11.Clear;

//menghitung output layer 3

for l:=1 to kelas do begin

temp:=0;

for k:=1 to hlayer2node do begin

temp:=temp+y2[k]\*w3[l,k];

end;

v3[l]:= temp+th3[l];

y3[l]:= 1/(1+exp(-alpha\*v3[l]));

listbox11.Items.Add('y3['+inttostr(l)+']='+format('%.6f',[y3[l]]));

end;

listbox11.Items.Add('--------------------------------');

y3[-4]:=0;

y3[-3]:=0;

y3[-2]:=0;

y3[-1]:=0;

y3[0]:=0;

for l:=1 to kelas do begin //determine input class algorithm (c) steven seaver

if (y3[l]>y3[l-4])and (y3[l]>y3[l-3]) and (y3[l]>y3[l-2])

and (y3[l]>y3[l-1]) and (y3[l]>y3[l+1])

and (y3[l]>y3[l+2]) and(y3[l]>y3[l+3]) and(y3[l]>y3[l+4]) then begin

listbox11.Items.Add('Class '+inttostr(l));

pointer:=l;

break;

end;

end;

listbox11.Items.Add('--------------------------------');

for l:=1 to kelas do begin

if pointer=l then begin

err\_mlp[l]:=1-y3[l];

end else

err\_mlp[l]:=0-y3[l];

listbox11.Items.Add('Error['+inttostr(l)+']='+format('%.6f',[err\_mlp[l]]));

end;

totalerror:=(err\_mlp[1]+err\_mlp[2]+err\_mlp[3]+err\_mlp[4]+err\_mlp[5])/5;

listbox11.Items.Add('Mean error = '+format('%.6f',[totalerror]));

mlp:=y3;

end;

## REFERENSI

 Rosenblatt, Frank. 1961. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books: Washington DC.