ABSTRAKSI

Back Propagation merupakan teknik yang sering digunakan untuk generalisasi pada Delta Rule. Delta Rule merupakan aturan yang berdasarkan pada ide sederhana yaitu secara berkesinambungan memodifikasi strength koneksi untuk mereduksi selisih / beda (disebut Delta) antara nilai keluaran yang diinginkan terhadap nilai keluaran saat itu dari hasil pemrosesan elemennya.

Algoritma Back Propagation pada saat diterapkan kemungkinan akan mempunyai pembelajaran yang rendah, dimana hal tersebut berhubungan dengan laju pembelajaran yang tetap. Karena hal itulah diperlukan suatu metoda pembelajaran dimana pengubahan nilai pada parameternya akan menyebabkan laju pembelajaran adaptif, sehingga iterasi yang berlebihan untuk mencapai konvergensi akan dapat dikurang. Adaptasi laju pembelajaran delta-bar-delta adalah salah satu metoda pengubahan laju pembelajaran yang diharapkan dapat mempercepat konvergensi dari backpropagation. Penelitian mengenai penerapan adaptasi laju pembelajaran dan penggunaan parameternya memerlukan suatu perangkat lunak, sekaligus juga untuk mengetahui pengaruhnya pada backpropagation.

Tugas Akhir ini menyajikan suatu perangkat lunak untuk mensaranai penelitian diterapkannya Adaptasi Laju Pembelajaran Delta-bar-delta pada Backpropagation dan melihat pengaruh parameter Delta-bar-delta (β,κ dan ζ) pada kinerja Backpropagation. Pengujian pada Delta-bar-delta memperlihatkan bahwa sistem konvergen pada iterasi 10 sampai iterasi ke 40, sedangkan pada backpropagation standar cenderung konvergen pada iterasi acak. Perangkat lunak menggunakan bahasa pemrograman aras tinggi Borland Delphi 3.0.

I. PENDAHULUAN

Latar Belakang

Jaringan Saraf Tiruan (*Artificial Neural Network)* melakukan pemetaan antara vektor masukan dan keluaran. Penamaan Jaringan Saraf Tiruan karana kemiripan strukturnya dengan apa yang ada pada otak manusia. Karakteristik jaringan ini ditentukan oleh sekumpulan elemen pemroses sederhana (atau disebut 'neuron') yang dikoneksikan oleh *weighted links.* [8]

Jaringan Saraf Tiruan terdiri dari sekumpulan *weighted links* dan sekumpulan *node*. *Node* dikelompokkan dalam suatu *layer*. *Layer* yang bukan merupakan masukan maupun keluaran dikenal sebagai *hidden layer*[8]. Algoritma yang efektif dan sering digunakan untuk algoritma *supervised learning* pada Jaringan Saraf Tiruan adalah Algoritma Backpropagation. Algoritma ini merupakan generalisasi dari *delta rule* (*perceptron learning rule)*.

Backpropagation (Rumelhart, Hinton, dan William, 1986) merupakan suatu metoda *network training* dari sekian banyak proyek Jaringan Saraf Tiruan dan untuk alasan yang bagus. Seperti halnya metoda 'lemah' lainnya, hal ini sederhana untuk diimplementasikan, lebih cepat daripada pendekatan "umumnya", *well-tested* di lapangan dan mudah untuk dibentuk (dengan domain pengetahuan terkode (*encoded*) dalam lingkungan pembelajarannya) ke dalam algoritma yang efisien dan sangat spesifik.[9]. Sayangnya metoda backpropagation bisa sangat lambat untuk praktek penerapannya[21].

Cost surface untuk jaringan multilayer dapat menjadi kompleks, kemudian untuk pemilihan laju pembelajaran bisa cukup sulit. Laju pembelajaran yang bagus ketika bekerja pada suatu lokasi *cost surface* bisa menjadi tidak bekerja sebagus itu pada lokasi yang lain. Delta-bar-delta marupakan algoritma heuristik untuk memodifikasi laju pembelajaran pada saat *training progress*. Ide dibalik metoda delta-bar-delta adalah membiarkan program itu sendiri untuk mencari laju pembelajaran pada tiap bobot (terjadi adaptasi laju pembelajaran pada saat *training progress).*

Perbandingan kinerja jaringan saraf tiruan backpropagation (tanpa adaptasi laju pembelajaran) dengan jaringan dengan metoda delta-bar-delta diterapkan, perlu diteliti lebih jauh terutama untuk mendapatkan perbedaan kecepatan konvergensinya (dalam hal iterasinya). Penelitian itu juga bisa mengungkap berapa sebaiknya penentuan awal / inisialisasi untuk parameter delta-bar-delta. Untuk dapat melakukan penelitian dan melakukan pengujian Jaringan Saraf Tiruan Backpropagation dengan adaptasi laju pembelajaran (delta-bar-delta), diperlukan suatu perangkat lunak yang mengimplementasikan jaringan saraf tiruan backpropagation dengan adaptasi laju pembelajaran, selain itu juga dapat menghasilkan data untuk penelitian selanjutnya .

Tujuan

Tujuan dari penulisan Tugas Akhir adalah : melakukan perancangan perangkat lunak Jaringan Saraf Tiruan Backpropagation dengan metoda delta-bar-delta dan menganalisis pengaruh adaptasi laju pembelajaran pada percepatan konvergensi.

Pembatasan Masalah

Dalam tulisan ini pembahasan dibatasi hanya pada ruang lingkup:

- arsitektur Jaringan Saraf Tiruan Backpropagation dengan menggunakan 2 *hidden layer*, 4 *node* pada *hidden layer* pertama, 3 *node* pada *hidden layer* kedua, 4 *node* masukan, 1 *node* keluaran, menggunakan 1 jenis *sample training* yang terdiri dari 40 *training set*.
- adaptasi laju pembelajaran dengan metoda delta-bar-delta
- pengaruh parameter κ,β,ζ pada laju perubahan rata-rata *squared error* (E_{av}), bobot, *learn rate* pada Jaringan Saraf Tiruan Backpropagation.
- jaringan mencapai konvergensi apabila laju perubahan rata-rata *squared error* (E_{av}) $\leq 2.10^{-3}$
- pembandingan kecepatan konvergensi diukur dari iterasinya.
- inisialisasi laju pembelajaran untuk jaringan Backpropagation dengan delta-bar-delta dan Backpropagation tanpa delta-bar-delta adalah 0,4.

II. ALGORITMA BACK PROPAGATION

Sebagai rangkuman algoritma bacpropagation adalah:

1. Koreksi $\Delta w_{ji}(n)$ yang ditrerapkan pada bobot yang menghubungkan neuron *i* sampai neuron j didefiniskan sebagai *delta rule* berikut :

$$\begin{pmatrix} Koreksi \\ bobot \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} Laju \\ pembelajaran \\ \eta \end{pmatrix} . \begin{pmatrix} Gradien \\ lokal \\ \partial_j(n) \end{pmatrix} . \begin{pmatrix} Sinyal masukan \\ neuron \ \ j \\ y_j(n) \end{pmatrix}$$

2. Gradien lokal $\delta_j(n)$ tergantung pada bagaimana neuron j sebagai keluaran atau hidden *node*:
   - Jika neuron j adalah *node* keluaran, $\delta_j(n)$ sama dengan produk turunan φ_j' $(v_j(n))$ dan sinyal *error* $e_j(n)$, keduanya berhubungan langsung dengan neuron *j*.
   - Jika neuron j adalah *node* keluaran, $\delta_j(n)$ sama dengan produk dari hubungan turunan φ_j' $(v_j(n))$ dan *weighted sum* dari semua δ dihitung untuk layer *hidden* atau keluaran selanjutnya dimana dikoneksikan ke neuron *j*

Laju Pembelajaran

Pada parameter laju pembelajaran η, semakin kecil nilainya akan semakin kecil pula perubahan bobot sinapsis dari satu iterasi ke iterasi selanjutnya, akan tetapi perubahan itu akan berpengaruh pada lambatnya laju belajar. Pada sisi yang lain parameter η dibuat terlalu besar, akan berbanding lurus dengan meningkatnya laju belajar, menghasilkan perubahan yang besar pada bobot sinapsis, diasumsikan jaringan akan menjadi tak stabil (misal berosilasi). Metoda sederhana untuk menaikkan laju belajar dan menghindari bahaya ketidak stabilan adalah memodifikasi *delta rule* yaitu dengan memasukkan *momentum*, seperti terlihat di bawah ini:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \partial_j(n) y_j(n) \quad (4)$$

dimana α biasanya adalah nilai positif yang disebut sebagai konstanta *momentum*, bila α diberi nilai nol maka algoritma dioperasikan tanpa momentum.

Konvergensi dan Inisialisasi Bobot

Jaringan dikatakan telah konvergen apabila laju perubahan rata-rata *squared error* (E_{av}) pada tiap *epoch* (siklus training) telah "cukup kecil", yaitu sekitar 0.01 persen.[4]. Inisialisasi *synaptic weight* dan level ambang pada jaringan seharusnya terdistribusi dalam *range* kecil, alasannya adalah untuk mengurangi kemungkinan neuron-neuron dalam jaringan saturasi dan menghasilkan gradien *error* kecil serta pembelajaran akan diinisialisasi sangat lambat[4]. Pilihan yang mungkin untuk inisialisasi bobot adalah dengan mengambil nilai acak pada range:

w = [- (2.4 / Jumlah input) , (2.4/Jumlah input)]

Algoritma Delta Bar Delta

- **Empat Aturan *Heuristic* Jacobs**

Jacobs memberi panduan sebagai pijakan untuk berpikir tentang bagaimana akselerasi konvergensi pada pembelajaran backpropagation dengan adaptasi laju pembelajaran sebagai berikut:

1. Tiap parameter jaringan dari *cost function* mempunyai parameter *individual* laju pembelajaran.
2. Tiap parameter laju pembelajaran bisa bervariasi dari satu iterasi ke iterasi selanjutnya.
3. Turunan dari *cost function* terhadap *synaptic weight* , apabila mempunyai tanda aljabar sama untuk iterasi berurutan dari algoritma tersebut maka parameter laju pembelajaran untuk khususnya weight tersebut dinaikkan.
4. Turunan dari *cost function* terhadap *synaptic weight* , apabila mempunyai tanda aljabar berkebalikan untuk iterasi berurutan dari algoritma tersebut maka parameter laju pembelajaran untuk khususnya weight tersebut diturunkan.

- **Aturan Pembelajaran Delta-bar-delta**[4]

Pada persamaan berikut, nilai *synaptic weight* w_{ji} menghubungkan neuron i ke neuron j, diukur pada iterasi n, parameter laju pembelajaran η_{ji} dihubungkan ke bobot pada iterasi ini. Perubahan laju pembelajaran kini dituliskan sebagai berikut:

$$\Delta\eta_{ji}(n+1) = \begin{cases} \kappa & \text{jika } S_{ji}(n-1)D_{ji}(n)>0 \\ -\beta\eta_{ji}(n) & \text{jika } S_{ji}(n-1)D_{ji}(n)<0 \\ 0 & \text{jika } S_{ji}(n-1)D_{ji}(n)=0 \end{cases} \quad \ldots\ldots\ldots\ldots (2.40)$$

dimana D_{ji} dan S_{ji} sendiri didefinisikan sebagai

$$D_{ji}(n) = \frac{\partial \xi(n)}{\partial w_{ji}(n)} \quad \ldots\ldots\ldots\ldots (2.41)$$

dan

$$S_{ji}(n) = (1-\zeta)D_{ji}(n-1) + \zeta S_{ji}(n-1)$$

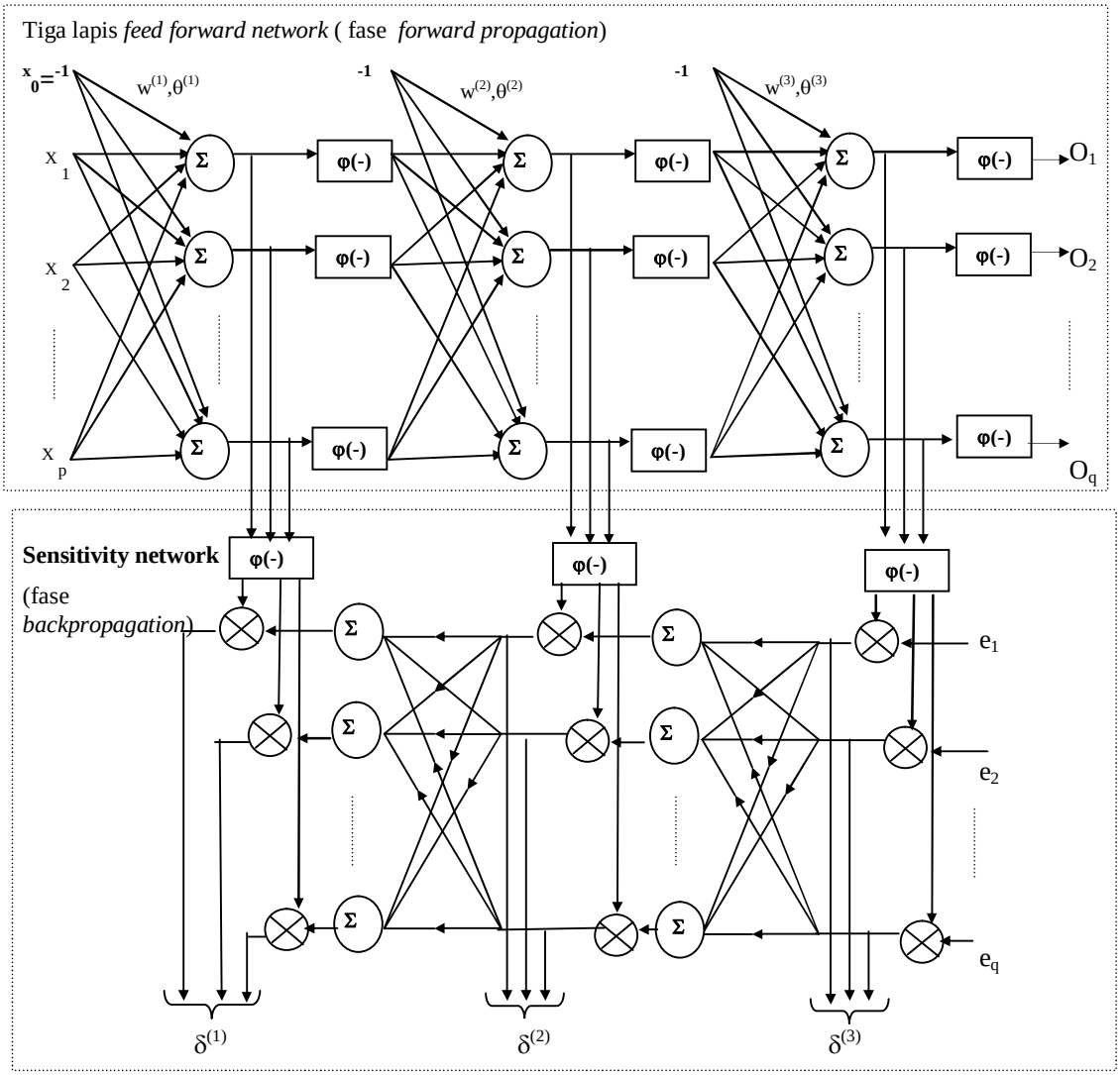
$$\ldots\ldots\ldots\ldots (2.42)$$

dimana ζ adalam konstanta positif. Kuantitas $D_{ji}(n)$ adalah *current value* dari turunan parsial permukaan *error* terhadap bobot $w_{ji}(n)$. Kuantitas kedua S_{ji} merupakan exponensial dari penjumlahan bobot pada turunan saat ini dan sebelumnya dari permukaan *error* terhadap bobot $w_{ji}(n)$, selanjutnya dengan ζ sebagai dasar dan iterasi ke n sebagai exponen. Sebagai catatan, jika parameter kontrol κ dan β di set sama dengan nol, parameter laju pembelajaran diasumsikan mempunyai nilai konstan sebagaimana ada pada *backpropagation* standar. Dari persamaan-persamaan di atas dapat ditulis pengamatan sebagai berikut:

1. Aturan pembelajaran delta-bar-delta menggunakan mekanisme seuai pada heuristik 3 dan 4. Secara spesifik, jika untuk *synaptic weight* w_{ji} turunan $D_{ji}(n)$ pada iterasi n dan q dan *exponentially weighted sum* $S_{ji}(n-1)$ pada iterasi n-1 sebelumnya mempunyai tanda sama, parameter laju pembelajaran pada bobot dinaikkan secara konstan κ. Jika pada sisi lain , kuantitas $D_{ji}(n)$ dan $S_{ji}(n-1)$ mempunyai tanda berlawanan, parameter laju pembelajaran pada bobot w_{ji} diturunkan dengan proporsi β pada *current value* $\eta_{ji}(n)$, S selain itu parameter laju pembelajaran tetap tak berubah
2. Parameter laju pembelajaran $\eta_{ji}(n)$ dinaikkan secara linier tetapi berbentuk penurunan exponensial (*decremented exponentially*). Menaikkan secara linier akan mencegah parameter laju pembelajaran menjadi tumbuh secara cepat, sebaliknya penurunan secara eksponensial mempunyai arti bahwa paramater laju

pembelajaran tetap positif dan berkurang / *reduced* secara cepat.

Pada penerapan secara *strict* dalam pembelajaran backpropagation menggunakan komputasi pengubahan bobot dan parameter untuk tiap pola masukan.



III. PERANCANGAN SISTEM

Perancangan perangkat lunak JST Backpropagation akan lebih mudah dipahami dengan melihat arsitektur Multilayer Perceptron seperti diperlihatkan pada Gambar 3.1.

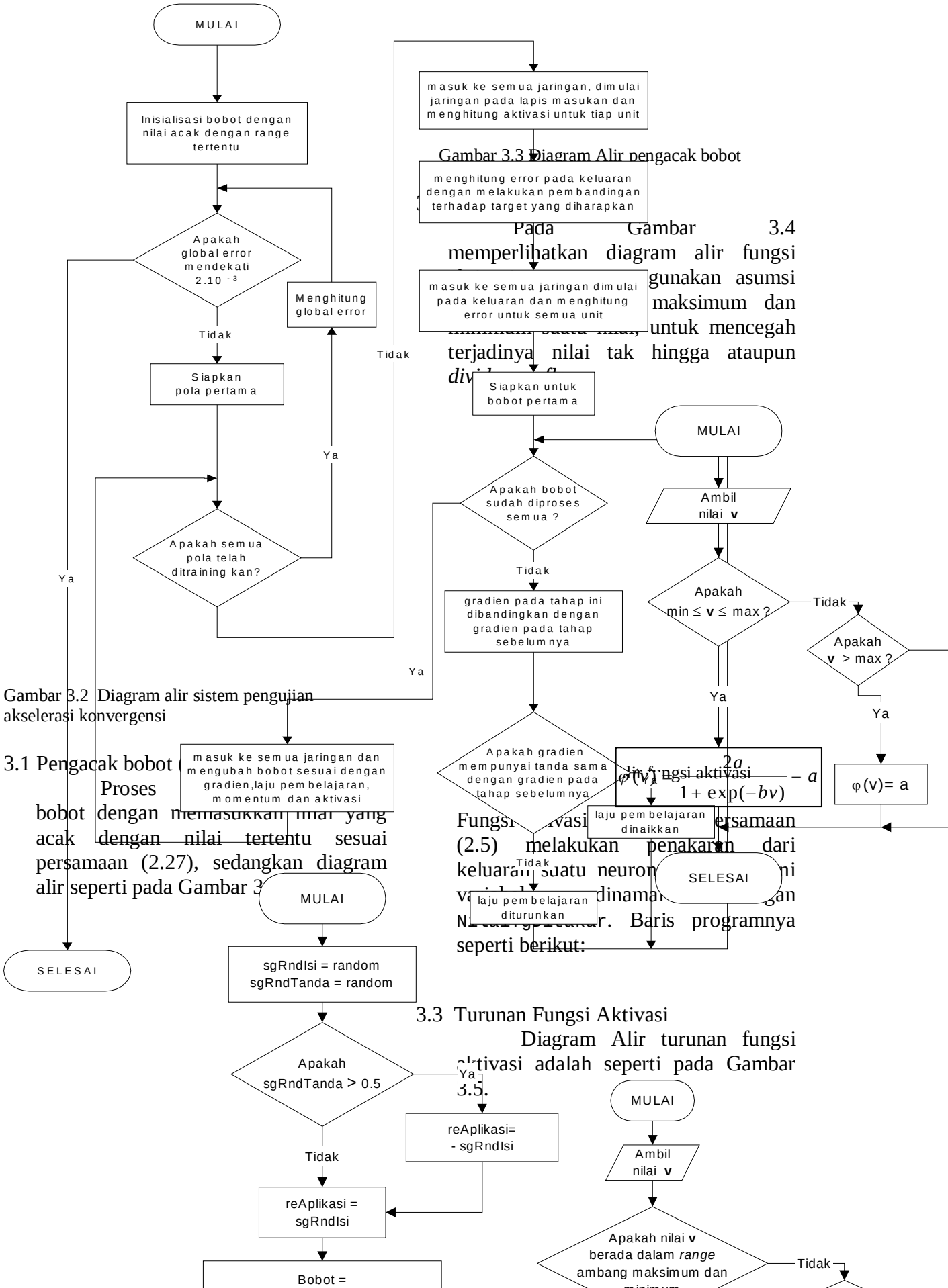
Gambar 3.1. Jaringan Multilayer Perceptron

Pada Gambar 3.1 terlihat koneksi / *link* antar node, mulai dari masukan x_n sampai dengan keluaran o_n, hanya saja untuk perancangan perangkat lunak pada tulisan ini menggunakan konfigurasi 4 node masukan , 4 node *hidden layer1*, 3 node *hidden layer2*, 1 node keluaran. Blok-blok penting yang perlu dibangun pada perancangan perangkat lunak ini adalah:

- Pengacak bobot/*weight*
- Fungsi aktivasi
- Turunan aktivasi
- Penghitungan *Forward* layer 1,2,3
- Penghitungan Gradien / δ di semua layer
- Penghitungan Laju Pembelajaran dengan Teknik Delta-bar-delta
- Penghitungan Perubahan bobot di layer 1,2,3
- Fungsi-fungsi pendukung

Blok-blok tersebut selanjutnya dilakukan integrasi satu sama lain dengan menggunakan aliran data tertentu, dimana hal ini terlihat pada Gambar 3.2.

Blok yang ada pada diagram alir tersebut tidak termasuk fungsi-fungsi antarmuka grafis dengan pengguna perangkat lunak, karena dengan pertimbangan bahwa fungsi tersebut bersifat umum, seperti pengelolaan file dan interprestasi grafis dan penentuan titik henti system. Interprestasi grafis itu sendiri terpisah dengan proses pembelajaran, dengan penjelasan bahwa proses *forward* dan *backward* system melakukan updating pada tiap iterasi pada suatu *variable global*. Blok interprestasi grafis secara periodik membaca *variable global* dengan *timer* dan menuliskannya dalam bentuk *chart*. Blok iterprestasi grafis tersebut berarti dipicu oleh suatu pewaktu (*timer*) yang artinya adalah tidak melakukan intervensi dalam proses – proses JST, namun hal ini tidak berlaku pada saat melakukan penulisan data parameter JST ke dalam suatu *history / log*. Penulisan ke *log file* harus dilaksanakan pada tiap iterasi, sehingga terlihat perubahan nilai bobot, laju pembelajaran dan perubahan error yang dicapai.



Gambar 3.2 Diagram alir sistem pengujian akselerasi konvergensi

3.1 Pengacak bobot (

Proses bobot dengan memasukkan nilai yang acak dengan nilai tertentu sesuai persamaan (2.27), sedangkan diagram alir seperti pada Gambar 3.

Gambar 3.3 Diagram Alir pengacak bobot

Pada Gambar 3.4 memperlihatkan diagram alir fungsi gunakan asumsi maksimum dan untuk mencegah terjadinya nilai tak hingga ataupun *divi*

3.2 Fungsi aktivasi

$$\varphi(v) = \frac{2a}{1+\exp(-bv)} - a$$

Fungsi persamaan (2.5) melakukan penakaran dari keluaran suatu neuron ni variabel dinamakan dengan Nilai tersebut.

Baris programnya seperti berikut:

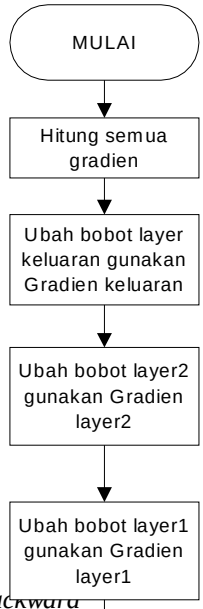
3.3 Turunan Fungsi Aktivasi

Diagram Alir turunan fungsi aktivasi adalah seperti pada Gambar 3.5.

Penghitungan *forward* mempunyai fungsi utama yaitu menjumlah input sebanding dengan bobotnya , menggunakan persamaan (2.1). Adapun v_i selanjutnya ditakar dengan fungsi aktivasi.

3.5 Penghitungan *Net Backward*

Pada blok ini mempunyai tujuan utama yaitu melakukan pemanggilan fungsi-fungsi yang akan mempersiapkan pengubahan bobot, selanjutnya barulah melakukan pemanggilan fungsi pengubahan bobot. Diagram Alir penghitungan *Net Backward* terlihat pada Gambar 3.7.



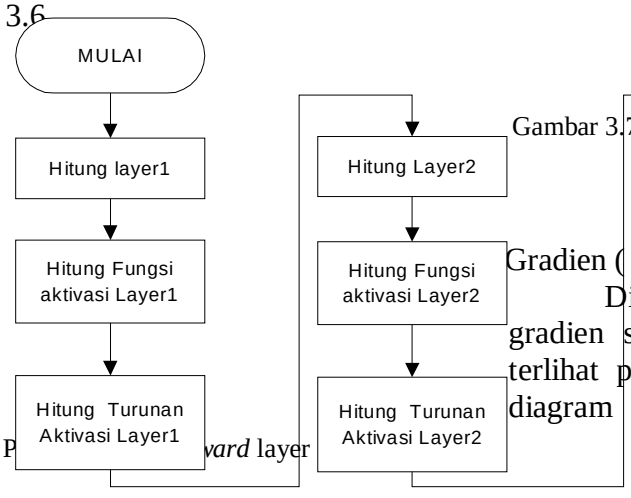
Gambar 3.7 Penghitungan *Net Backward*

Gambar 3.5 Diagram alir turunan fungsi aktivasi

Turunan fungsi aktivasi sesuai persamaan (2.6) merupakan pemroses dari keluaran fungsi aktivasi yang kemudian bisa digunakan untuk menghitung gradien.

3.4 Penghitungan *Forward* layer 1,2,3

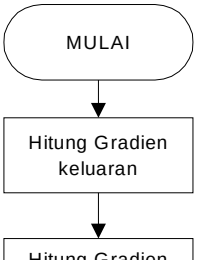
Diagram Alir penghitungan *forward* adalah seperti terlihat pada Gambar 3.6



Gambar 3.6 P[illegible]*ward* layer

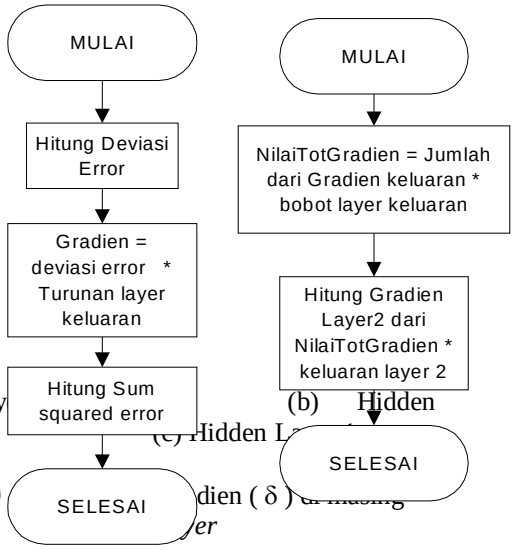
Gradien (δ) [illegible]r

Diag[illegible]enghitungan gradien secara umum (semua *layer*) terlihat pad[illegible], sedangkan diagram al[illegible]enghitungan



masing-masing *layer* seperti ditunjukkan pada Gambar 3.9.

Gambar 3.8 Diagram alir Gradien (δ) di semua layer



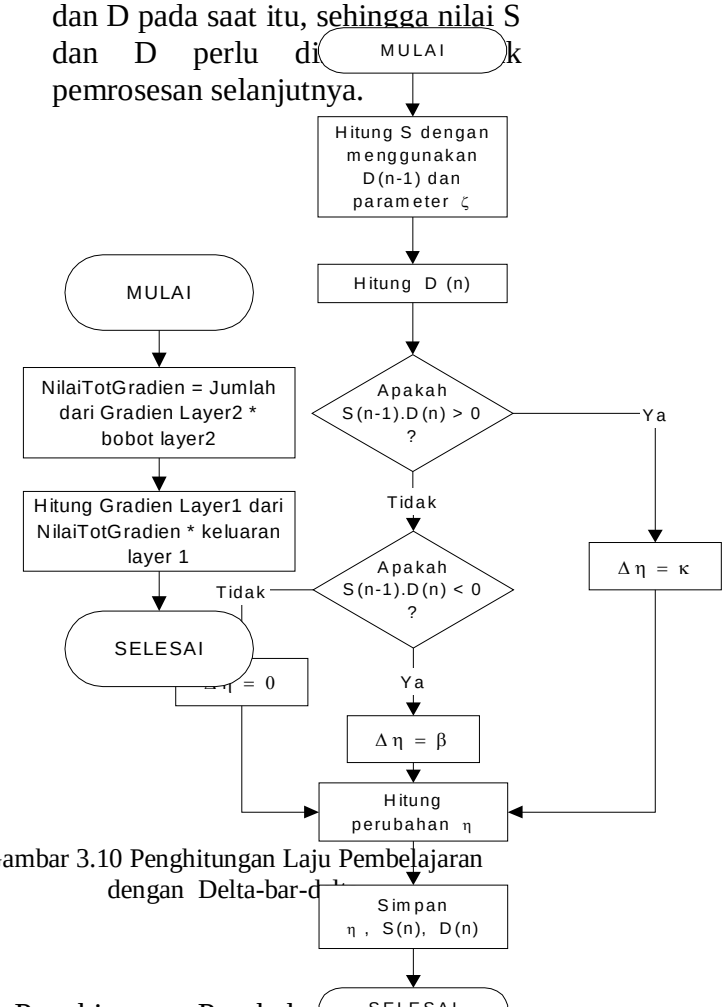
(a) Lay (b) Hidden Layer 2 (c) Hidden L

Gambar 3.9 dien (δ) di masing-masing *layer*

Prosedur ini berguna untuk nantinya menentukan nilai pengubahan bobot. Pada masing-masing lapisan prosedur ini mempunyai fungsi seperti pada persamaan (2.16).

3.7 Penghitungan Laju Pembelajaran dengan Teknik Delta-bar-delta

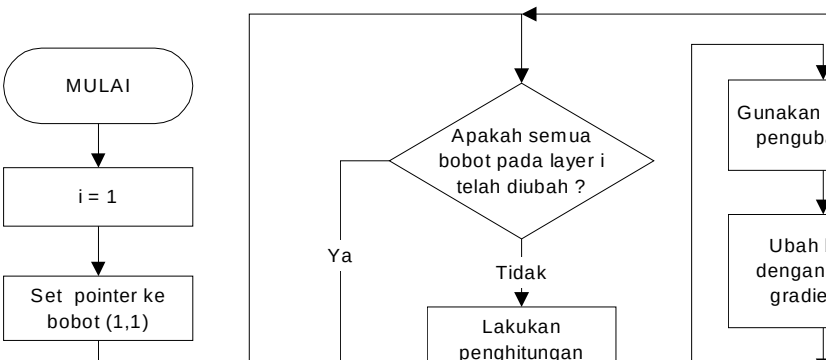
Diagram alir penerapan Adaptasi Laju Pembelajaran Delta-bar-delta seperti ditunjukkan pada Gambar 3.10. Blok ini memperlihatkan bahwa perubahan laju pembelajaran (Δη) dipengaruhi oleh beberapa parameter, yaitu S dan D, yang keduanya juga dipengaruhi oleh faktor δ baik yang lalu maupun yang saat itu dihitung. Untuk melakukan perubahan pada laju pembelajaran sendiri dipengaruhi secara langsung oleh nilai S yang lalu/iterasi sebelumnya dan D pada saat itu, sehingga nilai S dan D perlu di k pemrosesan selanjutnya.



Gambar 3.10 Penghitungan Laju Pembelajaran dengan Delta-bar-d

3.8 Penghitungan Perubaha er 1,2,3

Diagram alir penghitungan perubahan bobot di semua layer ditunjukkan pada Gambar 3.11.



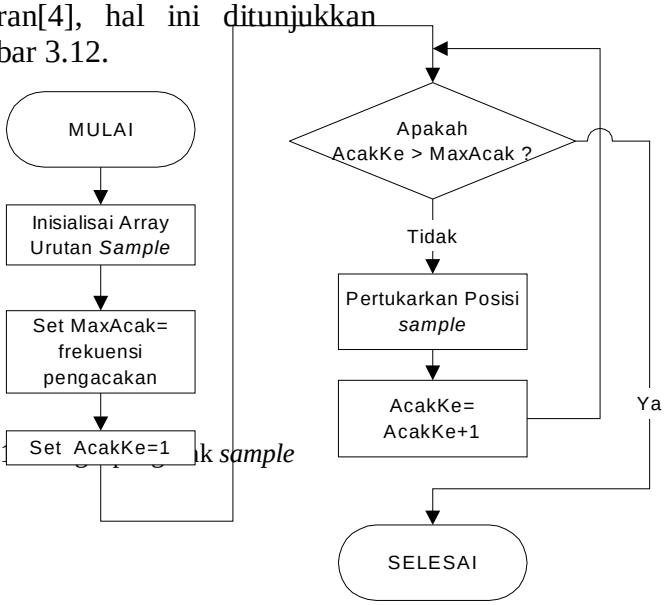
Keterangan : i adalah layer

Gambar 3.11 *Diagram alir* penghitungan Perubahan bobot di layer 1

Fungsi ini melakukan pengubahan bobot, dimana didalamnya terdapat Teknik Delta-Bar-Delta untuk mempersiapkan η. Fungsi ini selanjutnya menyimpan nilai sebelum pengubahan dan setelah pengubahan.

3.9 Fungsi pengacak *sample*

Urutan *sample* yang akan dijadikan masukan jaringan perlu diacak untuk meningkatkan kinerja pembelajaran[4], hal ini ditunjukkan pada Gambar 3.12.



Gambar 3.1 k *sample*