JARINGAN SARAF TIRUAN

(ARTIFICIAL NEURAL NETWORK)

Bagian ini membahas jaringan saraf tiruan, pengenalan tulisan tangan, dan algoritma *backpropagation*.

2.1 Jaringan Saraf Tiruan

Jaringan saraf tiruan (JST) atau *neural network* adalah suatu metode komputasi yang meniru sistem jaringan saraf biologis. Metode ini menggunakan elemen perhitungan non-linier dasar yang disebut neuron yang diorganisasikan sebagai jaringan yang saling berhubungan, sehingga mirip dengan jaringan saraf manusia. Jaringan saraf tiruan dibentuk untuk memecahkan suatu masalah tertentu seperti pengenalan pola atau klasifikasi karena proses pembelajaran[YAN05].

Layaknya neuron biologi, JST juga merupakan sistem yang bersifat "*fault tolerant*" dalam 2 hal. Pertama, dapat mengenali sinyal input yang agak berbeda dari yang pernah diterima sebelumnya. Sebagai contoh, manusia sering dapat mengenali seseorang yang wajahnya pernah dilihat dari foto atau dapat mengenali sesorang yang wajahnya agak berbeda karena sudah lama tidak menjumpainya. Kedua, tetap mampu bekerja meskipun beberapa neuronnya tidak mampu bekerja dengan baik. Jika sebuah neuron rusak, neuron lain dapat dilatih untuk menggantikan fungsi neuron yang rusak tersebut.

Jaringan saraf tiruan, seperti manusia, belajar dari suatu contoh karena mempunyai karakteristik yang adaptif, yaitu dapat belajar dari data-data sebelumnya dan mengenal pola data yang selalu berubah. Selain itu, JST merupakan sistem yang tak terprogram, artinya semua keluaran atau kesimpulan yang ditarik oleh jaringan didasarkan pada pengalamannya selama mengikuti proses pembelajaran/pelatihan.

Hal yang ingin dicapai dengan melatih JST adalah untuk mencapai keseimbangan antara kemampuan *memorisasi* dan *generalisasi.* Yang dimaksud kemampuan *memorisasi* adalah kemampuan JST untuk mengambil kembali secara sempurna sebuah pola yang telah dipelajari. Kemampuan *generalisasi* adalah kemampuan JST untuk menghasilkan respons yang bisa diterima terhadap pola-pola input yang serupa (namun tidak identik) dengan pola-pola yang sebelumnya telah dipelajari. Hal ini sangat bermanfaat bila pada suatu saat ke dalam JST itu diinputkan informasi baru yang belum pernah dipelajari, maka JST itu masih akan tetap dapat memberikan tanggapan yang baik, memberikan keluaran yang paling mendekati [PUS06].

Jaringan saraf tiruan berkembang secara pesat pada beberapa tahun terakhir. Jaringan saraf tiruan telah dikembangkan sebelum adanya suatu komputer konvensional yang canggih dan terus berkembang walaupun pernah mengalami masa vakum selama beberapa tahun.

JST menyerupai otak manusia dalam dua hal, yaitu:

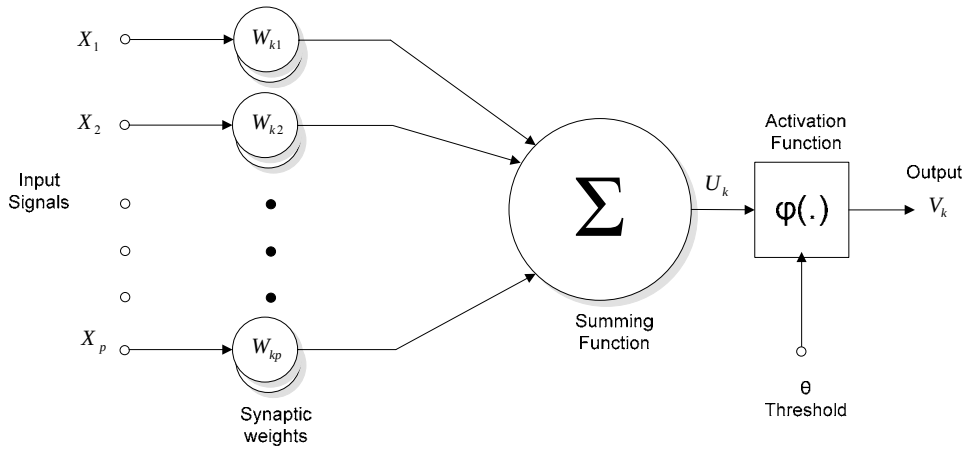
1. Pengetahuan diperoleh jaringan melalui proses belajar.
2. Kekuatan hubungan antar sel syaraf (neuron) yang dikenal sebagai bobot-bobot sinaptik digunakan untuk menyimpan pengetahuan.

JST ditentukan oleh 3 hal [JON04] :

1. Pola hubungan antar neuron (disebut arsitektur jaringan). Bagian ini selanjutnya akan dijelaskan pada Bab 2.1.3.
2. Metode untuk menentukan bobot penghubung (disebut metode *training/learning*). Bagian ini selanjutnya akan dijelaskan pada Bab 2.1.4.
3. Fungsi aktivasi, yaitu fungsi yang digunakan untuk menentukan keluaran suatu neuron. Bagian ini selanjutnya akan dijelaskan pada Bab 2.1.5.

2.1.1 Model Neuron

Satu sel syaraf terdiri dari tiga bagian, yaitu: fungsi penjumlah (*summing function*), fungsi aktivasi (*activation function*), dan keluaran (*output*).



Gambar 1. Model Neuron

Jika kita lihat, neuron buatan diatas mirip dengan sel neuron biologis. Informasi (input) akan dikirim ke neuron dengan bobot tertentu. Input ini akan diproses oleh suatu fungsi yang akan menjumlahkan nilai-nilai bobot yang ada. Hasil penjumlahan kemudian akan dibandingkan dengan suatu nilai ambang (*threshold)* tertentu melalui fungsi aktivasi setiap neuron. Apabila input tersebut melewati suatu nilai ambang tertentu, maka neuron tersebut akan diaktifkan, jika tidak, maka neuron tidak akan diaktifkan. Apabila neuron tersebut diaktifkan, maka neuron tersebut akan mengirimkan output melalui bobot-bobot outputnya ke semua neuron yang berhubungan dengannya.

Sehingga dapat disimpulkan bahwa neuron terdiri dari 3 elemen pembentuk, yaitu:

1. Himpunan unit-unit yang dihubungkan dengan jalur koneksi. Jalur-jalur tersebut memiliki bobot yang berbeda-beda. Bobot yang bernilai positif akan memperkuat sinyal dan yang bernilai negatif akan memperlemah sinyal yang dibawa. Jumlah, struktur, dan pola hubungan antar unit-unit tersebut akan menentukan arsitektur jaringan.

2. Suatu unit penjumlah yang akan menjumlahkan input-input sinyal yang sudah dikalikan dengan bobotnya.
3. Fungsi aktivasi yang akan menentukan apakah sinyal dari input neuron akan diteruskan ke neuron lain atau tidak.

2.1.2 Konsep Dasar Jaringan Saraf Tiruan

Setiap pola-pola informasi input dan output yang diberikan kedalam JST diproses dalam neuron. Neuron-neuron tersebut terkumpul di dalam lapisan-lapisan yang disebut *neuron layers*. Lapisan-lapisan penyusun JST tersebut dapat dibagi menjadi 3, yaitu :

1. Lapisan input

   Unit-unit di dalam lapisan input disebut unit-unit input. Unit-unit input tersebut menerima pola inputan data dari luar yang menggambarkan suatu permasalahan.
2. Lapisan tersembunyi

   Unit-unit di dalam lapisan tersembunyi disebut unit-unit tersembunyi. Dimana outputnya tidak dapat secara langsung diamati.
3. Lapisan output

   Unit-unit di dalam lapisan output disebut unit-unit output. Output dari lapisan ini merupakan solusi JST terhadap suatu permasalahan.

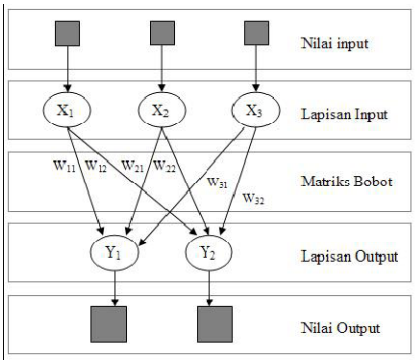
2.1.3 Arsitektur Jaringan

JST memiliki beberapa arsitektur jaringan yang sering digunakan dalam berbagai aplikasi. Arsitektur JST tersebut, antara lain [KUS03] :

1. Jaringan layar tunggal (*single layer network*)

   Jaringan dengan lapisan tunggal terdiri dari 1 *layer* input dan 1 *layer* output. Setiap neuron/unit yang terdapat di dalam lapisan/*layer* input selalu terhubung dengan setiap neuron yang terdapat pada *layer* output. Jaringan ini hanya menerima input kemudian secara langsung akan mengolahnya menjadi output tanpa harus melalui lapisan tersembunyi.

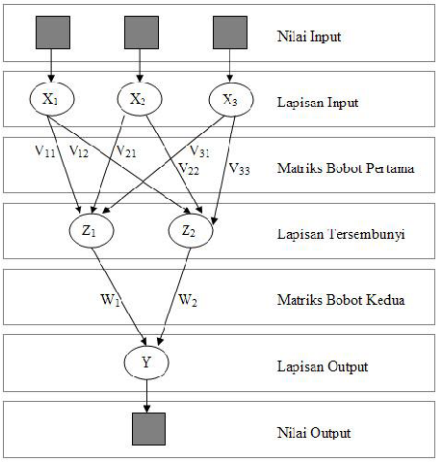
Contoh algoritma JST yang menggunakan metode ini yaitu : ADALINE, Hopfield, Perceptron.



Gambar 2. Arsitektur layer tunggal

2. Jaringan layar jamak (*multi layer network)*

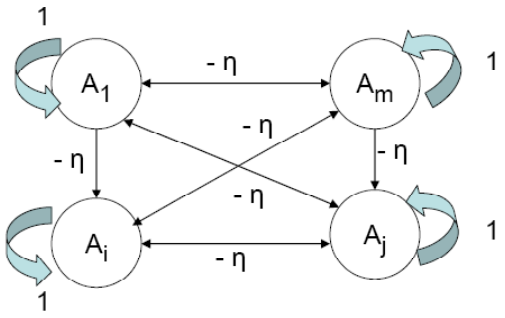
   Jaringan dengan lapisan jamak memiliki ciri khas tertentu yaitu memiliki 3 jenis *layer* yakni *layer* input, *layer* output, dan juga *layer* tersembunyi. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan jaringan dengan lapisan tunggal. Namun, proses pelatihan sering membutuhkan waktu yang cenderung lama. Contoh algoritma JST yang menggunakan metode ini yaitu : MADALINE, *backpropagation, Neocognitron*.



Gambar 3. Arsitektur layer jamak

3. Jaringan dengan lapisan kompetitif (*competitive layer network)*

   Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi aktif. Contoh algoritma yang menggunakan metode ini adalah LVQ.



Gambar 4. Arsitektur layer kompetitif

2.1.4 Metode Pelatihan/Pembelajaran

Cara berlangsungnya pembelajaran atau pelatihan JST dikelompokkan menjadi 3 yaitu [PUS06] :

a) *Supervised learning* (pembelajaran terawasi)

   Pada metode ini, setiap pola yang diberikan kedalam JST telah diketahui outputnya. Selisih antara pola output aktual (output yang dihasilkan) dengan pola output yang dikehendaki (output target) yang disebut error digunakan untuk mengoreksi bobot JST sehingga JST mampu menghasilkan output sedekat mungkin dengan pola target yang telah diketahui oleh JST. Contoh algoritma JST yang menggunakan metode ini adalah : Hebbian, Perceptron, ADALINE, Boltzman, Hopfield, *Backpropagation.*

b) *Unsupervised learning (*pembelajaran tak terawasi*)*

   Pada metode ini, tidak memerlukan target output. Pada metode ini tidak dapat ditentukan hasil seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai input yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk klasifikasi pola. Contoh

algoritma JST yang menggunakan metode ini adalah : Competitive, Hebbian, Kohonen, LVQ (*Learning Vector Quantization*), Neocognitron.

c) *Hybrid Learning* (pembelajaran hibrida)

Merupakan kombinasi dari metode pembelajaran *supervised learning* dan *unsupervised learning*. Sebagian dari bobot-bobotnya ditentukan melalui pembelajaran terawasi dan sebagian lainnya melalui pembelajaran tak terawasi. Contoh algoritma JST yang menggunakan metode ini yaitu : algoritma RBF.

Metode algoritma yang baik dan sesuai dalam melakukan pengenalan pola-pola gambar adalah algoritma *Backpropagation* dan *Perceptron*. Untuk mengenali teks bedasarkan tipe font akan digunakan algoritma *Backpropagation.*

2.1.5 Fungsi Aktivasi

Dalam JST, fungsi aktivasi digunakan untuk menentukan keluaran suatu neuron. Argumen fungsi aktivasi adalah net masukan (kombinasi linier masukan dan bobotnya) [JON04].

Jika net = $\Sigma x_i w_i$ ……..(1)

Maka fungsi aktivasinya adalah :

$f(net) = f(\Sigma x_i w_i)$ ……..(2)

Beberapa fungsi aktivasi yang digunakan adalah [JON04] :

a) Fungsi *threshold* (batas ambang)

$$f(x) = \begin{cases} 1 \ldots x \geq a \\ 0 \ldots x \leq a \end{cases} \quad \text{……..(3)}$$

Fungsi *threshold* (3) merupakan fungsi *threshold biner*. Untuk kasus bilangan bipolar, maka angka 0 diganti dengan angka -1. Sehingga persamaan (3) diubah menjadi :

$$f(x) = \begin{cases} 1 \ldots x \geq a \\ -1 \ldots x \leq a \end{cases} \quad \text{……..(4)}$$

Adakalanya dalam JST ditambahkan suatu unit masukan yang nilainya selalu 1. Unit tersebut dikenal dengan bias. Bias dapat dipandang sebagai sebuah input yang nilainya selalu 1. Bias berfungsi untuk mengubah *threshold* menjadi = 0.

b) Fungsi sigmoid

$f(x) = 1/1+e^{-x}$ …….(5)

Fungsi ini sering digunakan karena nilai fungsinya yang sangat mudah untuk didiferensiasikan.

$f(x) = f(x)(1-f(x))$ …….(6)

c) Fungsi identitas

$f(x) = x$ …….(7)

Digunakan jika keluaran yang dihasilkan oleh JST merupakan sembarang bilangan riil (bukan hanya pada range [0,1] atau [1,-1]).

2.1.6 Algoritma Umum JST

Algoritma pembelajaran/pelatihan JST [PUS06] :

Dimasukkan n contoh pelatihan kedalam JST, lakukan :

1. Inisialisasi bobot-bobot jaringan. Set i = 1.
2. Masukkan contoh ke-i (dari sekumpulan contoh pembelajaran yang terdapat dalam set pelatihan) kedalam jaringan pada lapisan input.
3. Cari tingkat aktivasi unit-unit input menggunakan algoritma aplikasi

   If → kinerja jaringan memenuhi standar yang ditentukan sebelumnya (memenuhi syarat untuk berhenti)

   Then → exit
4. Update bobot-bobot dengan menggunakan aturan pembelajaran jaringan.
5. If i = n then reset i = 1

   Else i = i-1

   Ke langkah 2.

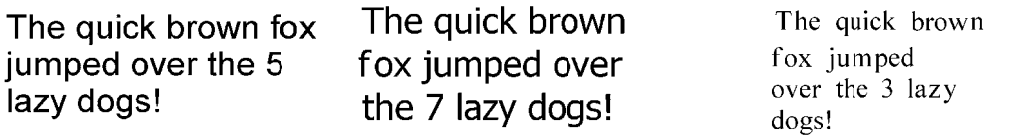
Algoritma aplikasi/inferensi jaringan saraf tiruan [PUS06] :

Dimasukkan sebuah contoh pelatihan kedalam jaringan saraf tiruan, lakukan :

1. Masukkan kasus kedalam jaringan pada lapisan input.
2. Hitung tingkat aktivasi node-node jaringan.
3. Untuk jaringan koneksi umpan maju, jika tingkat aktivasi dari semua unit outputnya telah dikalkulasi, maka exit. Untuk jaringan dengan kondisi balik, jika tingkat aktivasi dari semua unit output menjadi konstan atau mendekati konstan, maka exit. Jika tidak, kembali ke langkah 2. Jika jaringannya tidak stabil, maka exit dan fail.

2.2 Pengenalan Teks Cetak

Pada proses pengenalan teks cetak, *image* harus dikumpulkan dan di-*prepocessing*. Kemudian, teks juga diekstraksi berdasar karakter ataupun kata. Pengenalan teks cetak ini ditujukan untuk karakter alfanumerik a..z,A..Z. Teks cetak dibaca melalui scanner, disimpan dalam format file BMP monokrom. Kerangka karakter merupakan masukan bagi metode pendekatan yang digunakan untuk mengekstrasi (kerangka) karakter. Karakter tidak dapat dikenali selama polanya belum pernah dipelajari oleh sistem. Pertama kali proses belajar harus dilakukan untuk mendapatkan deskripsi pola karakter a..z,A..Z. Deskripsi disimpan dalam file teks secara akumulatif. Berdasarkan deskripsi tersebut sistem diharapkan dapat mengenali karakter alfanumerik [PUR08].



(a) The quick brown fox (Arial)

(b) The quick brown fox (Roman)

(c) The quick brown fox (Tahoma)

Gambar 5. Contoh sampel yang digunakan

Pengenalan selanjutnya dilakukan dengan mencocokkan komposisi struktur element dengan struktur yang sudah disimpan memakai aturan tertentu. Tahapan-tahapannya antara lain sebagai berikut [TRI08]:

1. *Data Capture*, merupakan proses konversi suatu dokumen (hardcopy) menjadi suatu file gambar (BMP).
2. *Preprocessing*, merupakan suatu proses untuk menghilangkan bagian-bagian yang tidak diperlukan pada gambar input untuk proses selanjutnya. Beberapa contoh preprocessing adalah *noise filtering*.
3. *Segmentation*, proses memisahkan area pengamatan (region) pada tiap karakter yang akan dideteksi.
4. *Normalization*, proses merubah dimensi region tiap karakter dan ketebalan karakter. Algoritma yang digunakan pada proses ini adalah algoritma *scaling* dan *thinning*.
5. *Feature Extraction*, proses untuk mengambil ciri-ciri tertentu dari karakter yang diamati.
6. *Recognition*, proses untuk mengenali karakter yang diamati dengan cara membandingkan ciri-ciri karakter yang diperoleh dengan ciri-ciri karakter yang ada pada database.
7. *Postprocessing*, pada umumnya proses yang dilakukan pada tahap ini adalah proses koreksi ejaan sesuai dengan bahasa yang digunakan.

2.3 Propagasi Balik

Propagasi balik atau b*ackpropagation* merupakan suatu teknik pembelajaran/pelatihan *supervised learning* yang paling banyak digunakan. Metode ini merupakan salah satu metode yang sangat baik dalam menangani masalah pengenalan pola-pola kompleks.

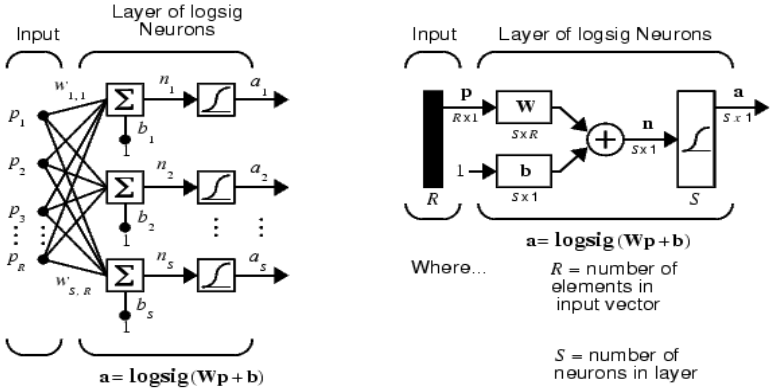
Di dalam jaringan propagasi balik, setiap unit yang berada di lapisan input terhubung dengan setiap unit yang ada di lapisan tersembunyi. Setiap unit yang ada di lapisan tersembunyi terhubung dengan setiap unit yang ada di lapisan output. Jaringan ini terdiri dari banyak lapisan (*multilayer network*). Ketika jaringan diberikan pola masukan sebagai pola pelatihan, maka pola tersebut menuju unit-unit lapisan tersembunyi untuk selanjutnya diteruskan pada unit-unit di lapisan keluaran. Kemudian unit-unit lapisan keluaran akan memberikan respon sebagai keluaran JST. Saat hasil keluaran tidak sesuai dengan yang diharapkan, maka keluaran akan disebarkan mundur (*backward*) pada lapisan tersembunyi kemudian dari lapisan tersembunyi menuju lapisan masukan [PUS06].

Tahap pelatihan ini merupakan langkah untuk melatih suatu JST, yaitu dengan cara melakukan perubahan bobot. Sedangkan penyelesaian masalah akan dilakukan jika proses pelatihan tersebut telah selesai, fase ini disebut fase pengujian [PUS06].

2.3.1 Arsitektur Propagasi Balik

Setiap unit di dalam *layer* input pada jaringan propagasi balik selalu terhubung dengan setiap unit yang berada pada *layer* tersembunyi, demikian juga setiap unit pada *layer* tersembunyi selalu terhubung dengan unit pada *layer* output. Jaringan propagasi balik terdiri dari banyak lapisan (*multilayer network*) [PUS06], yaitu :

1. Lapisan input (1 buah), yang terdiri dari 1 hingga n unit input.
2. Lapisan tersembunyi (minimal 1 buah), yang terdiri dari 1 hingga p unit tersembunyi.
3. Lapisan output (1 buah), yang terdiri dari 1 hingga m unit output.



Gambar 6. Arsitektur *Backpropagation*

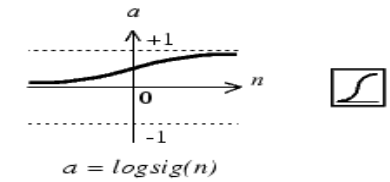
2.3.2 Fungsi Aktivasi Jaringan Propagasi Balik

Fungsi Sigmoid Biner

Fungsi ini merupakan fungsi yang umum digunakan dan akan diterapkan pada aplikasi, rangenya adalah (0,1) dan didefinisikan sebagai :

$$y = \varphi(u) = \frac{1}{1+e^{-u}} \qquad \text{…….(8)}$$

Dengan kurva sebagai berikut :



Log-Sigmoid Transfer Function

2.3.3 Pelatihan Jaringan Propagasi Balik

Aturan pelatihan jaringan propagasi balik terdiri dari 2 tahapan, *feedforward* dan *backward propagation*. Pada jaringan diberikan sekumpulan contoh pelatihan yang disebut set pelatihan. Set pelatihan ini digambarkan dengan sebuah *vector feature* yang disebut dengan vektor input yang diasosiasikan dengna sebuah output yang menjadi target pelatihannya. Dengan kata lain set pelatihan terdiri

dari vektor input dan juga vektor output target. Keluaran dari jaringan berupa sebuah vektor output aktual. Selanjutnya dilakukan perbandingan antara output aktual yang dihasilkan dengan output target dengan cara melakukan pengurangan diantara kedua output tersebut. Hasil dari pengurangan merupakan error. Error dijadikan sebagai dasar dalam melakukan perubahan dari setiap bobot yang ada dengan mempropagasikannya kembali [PUS06].

Setiap perubahan bobot yang terjadi dapat mengurangi error. Siklus setiap perubahan bobot (*epoch*) dilakukan pada setiap set pelatihan hingga kondisi berhenti dicapai, yaitu bila mencapai jumlah *epoch* yang diinginkan atau hingga sebuah nilai ambang yang ditetapkan terlampaui. [PUS06]. Algoritma pelatihan jaringan propagasi balik terdiri dari 3 tahapan yaitu [PUS06] :

1. Tahap umpan maju (*feedforward*)
2. Tahap umpan mundur (*backpropagation*)
3. Tahap pengupdatean bobot dan bias.

Secara rinci algoritma pelatihan jaringan propagasi balik dapat diuraikan sebagai berikut :

Langkah 0 : Inisialisasi bobot-bobot, konstanta laju pelatihan (α), toleransi error atau nilai bobot (bila menggunakan nilai bobot sebagai kondisi berhenti) atau set maksimal *epoch* (jika menggunakan banyaknya *epoch* sebagai kondisi berhenti).

Langkah 1 : Selama kondisi berhenti belum dicapai, maka lakukan langkah ke-2 hingga langkah ke-9.

Langkah 2 : Untuk setiap pasangan pola pelatihan, lakukan langkah ke-3 sampai langkah ke-8.

Tahap I : Umpan Maju (*feedforward)*

Langkah 3 : Setiap unit input x_i (dari unit ke-1 hingga unit ke-n pada lapisan input) mengirimkan sinyal input ke setiap input yang berada pada lapisan tersembunyi.

Langkah 4 : Masing-masing unit di lapisan tersembunyi (dari unit ke-1 hingga unit ke-p) dikalikan dengan bobotnya dan dijumlahkan serta ditambahkan dengan biasnya :

$$z\_net_j = v_{jo} + \sum_{i=1}^{n} x_i v_{ji} \qquad \text{.....(9)}$$

$$z_j = f(z\_net_j) = \frac{1}{1 + e^{-z\_net_i}} \qquad \text{.....(10)}$$

Langkah 5 : Masing-masing unit output (y_k, k=1,2,3,…m) dikalikan dengan bobot dan dijumlahkan serta ditambahkan dengan biasnya.

$$y\_net_k = w_{ko} + \sum_{j=1}^{p} z_j k_j \qquad \text{.....(11)}$$

$$y_k = f(y\_net_k) = \frac{1}{1 + e^{-y\_net_k}} \qquad \text{.....(12)}$$

Tahap II : Umpan Mundur (*backward propagation)*

Langkah 6 : Masing-masing unit output (y_k, k=1,2,3,…,m) menerima pola target t_k sesuai dengan pola masukan/input saat pelatihan dan kemudian informasi kesalahan/error lapisan output (δ_k) dihitung. δ_k dikirim ke lapisan dibawahnya dan digunakan untuk menghitung besarnya koreksi bobot dan bias (ΔW_{jk} dan ΔW_{ok}) antara lapisan tersembunyi dengan lapisan output :

$$\delta_k = (t_k - y_k)\, f'(y\_net_k) = (t_k - y_k)\, y_k(1-y_k) \qquad \text{.....(13)}$$

Hitung suku perubahan bobot W_{jk} (yang akan digunakan untuk merubah bobot W_{jk}) dengan laju pelatihan α

$$\Delta W_{kj} = \alpha\, \delta_k\, z_j \quad ;k=1,2,3,\ldots,m;j=0,1,\ldots,p \qquad \text{.....(14)}$$

Hitung perubahan bias

$$\Delta W_{ok} = \alpha\, \delta_k \qquad \text{.....(15)}$$

Langkah 7 : Pada setiap unit di lapisan tersembunyi (dari unit ke-1 hingga ke-p; i=1…n;k=1…m) dilakukan perhitungan informasi kesalahan lapisan tersembunyi (δ_j). δ_j kemudian digunakan untuk menghitung besar koreksi bobot dan bias (ΔV_{ji} dan ΔV_{jo}) antara lapisan input dan lapisan tersembunyi.

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k w_{kj} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(16)$$

$$\delta_j = \delta\_net_j \, f'(\delta\_net_j) = \delta\_net_j \, z_j(1-z_j) \qquad \ldots\ldots\ldots\ldots\ldots\ldots(17)$$

Hitung suku perubahan bobot V_{ji} (yang digunakan untuk perbaikan bobot V_{ji}).

$$\Delta V_{ji} = \alpha \, \delta_j \, x_i \qquad \ldots\ldots\ldots\ldots\ldots\ldots(18)$$

Hitung perubahan bias (untuk memperbaiki V_{jo}).

$$\Delta V_{jo} = \alpha \, \delta_j \qquad \ldots\ldots\ldots\ldots\ldots\ldots(19)$$

Tahap III : Pengupdatean Bobot dan Bias

Langkah 8 : Masing-masing unit output/keluaran (y_k, k=1,2,3,…,m) dilakukan pengupdatean bias dan bobotnya (j = 0,1,2,…,p) sehingga menghasilkan bobot dan bias baru :

$$W_{kj}\,(\text{baru}) = W_{kj}(\text{lama}) + \Delta W_{kj} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(20)$$

Demikian juga untuk setiap unit tersembunyi mulai dari unit ke-1 sampai dengan unit ke-p dilakukan pengupdatean bobot dan bias :

$$V_{ji}(\text{baru}) = V_{ji}(\text{lama}) + \Delta V_{ji} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(21)$$

Langkah 9 : Uji kondisi berhenti (akhir iterasi)

2.3.4 Inisialisasi Nguyen-Widrow

Inisialisasi Nguyen-Widrow merupakan sebuah metode penginisialisasian nilai bobot dan bias jaringan *backpropagation* yang lebih cepat bila dibandingkan dengan penginisialisasian secara random/acak. Tahap ini digunakan untuk mengoptimisasi jaringan *backpropagation* sehingga lebih mudah mencapai konvergensi.

Pada inisialisasi Nguyen-Widrow, inisialisasi acak tetap dipakai tetapi digunakan untuk menginisialisasi bias dan bobot dari unit tersembunyi ke unit output. Untuk bias dan bobot dari unit input ke unit tersembunyi digunakan bias dan bobot yang diskalakan agar jatuh pada range tertentu [PUS06].

Faktor skala Nguyen-Widrow (β) didefinisikan sebagai :

$$\beta = 0.7\ (p)^{1/n} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(22)$$

dimana :

n = banyak unit input

p = banyak unit tersembunyi

β = faktor skala

Prosedur Inisialisasi Nguyen-Widrow

Untuk setiap unit tersembunyi dari unit ke-1 sampai unit ke-p :

1. Inisialisasi vektor bobot dari unit input ke unit tersembunyi (j = 1,…,p) dengan cara :
   - Menentukan bobot-bobot antara unit input ke unit tersembunyi (v_{ij}) :

     V_{ij}(lama) = bilangan acak antara – β dan β dengan i = 1,…,n

   - Menghitung $\|v_j\| = \sqrt{v^2_{j1} + v^2_{j2} + \ldots + v^2_{jn}}$ ……(23)

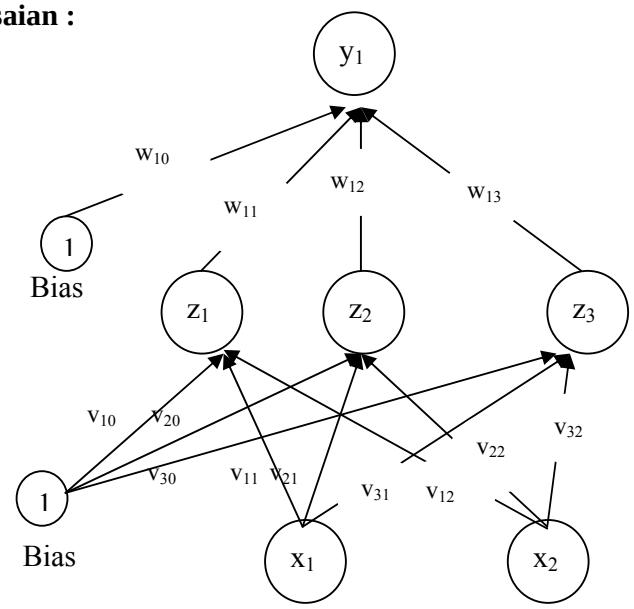
   - Menginisialisasikan kembali vij dengan : $v_{ij} = \dfrac{\beta . v_{ij}(\text{lama})}{\|v_j\|}$ ……(24)

2. Menentukan bias antara input ke unit tersembunyi (j = 1,…p) dengan v_{0j} di set bilangan acak yang terletak pada skala antara –β dan β.

Contoh 1 :

Menggunakan algoritma *backpropagation* dengan 1 buah layer tersembunyi untuk mengenali fungsi logika XOR dengan 2 masukan x_1 dan x_2. Buatlah iterasi untuk menghitung bobot jaringan untuk pola pertama ($x_1 = 1$, $x_2 = 1$ dan $t = 0$) dengan laju pelatihan $\alpha = 0.2$

Penyelesaian :



Langkah 0 : Mula-mula bobot diberi nilai acak yang kecil (range [-1,1]). Misalkan di dapat bobot seperti tabel II.1 (bobot dari layer input ke layer tersembunyi = v_{ij}) dan II.2 (bobot dari layer tersembunyi ke layer output = w_{kj}).

Tabel II.1 Tabel bobot dari layer input ke layer tersembunyi

	z_1	z_2	z_3
x_1	0.2	0.3	-0.1
x_2	0.3	0.1	-0.1
1	-0.3	0.3	0.3

Tabel II.2 Tabel bobot dari layer tersembunyi ke layer output

	y
z_1	0.5
z_2	-0.3
z_3	-0.4
1	-0.1

Langkah 1 : Jika kondisi penghentian belum terpenuhi, lakukan langkah 2-8

Langkah 2 : Untuk setiap pasang data pelatihan, lakukan langkah 3-8

Tahap I : Umpan Maju (*feedforward)*

Langkah 3 : Setiap unit input mengirim sinyal ke unit tersembunyi

Langkah 4 : hitung keluaran di unit tersembunyi (z_j)

$$z\_net_j = v_{jo} + \sum_{i=1}^{n} x_i v_{ji} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(9)$$

z_net1 = -0.3 + 1(0.2) + 1(0.3) = 0.2

z_net2 = 0.3 + 1(0.3) + 1(0.1) = 0.7

z_net3 = 0.3 + 1(-0.1) + 1(-0.1)= 0.1

$$z_j = f(z\_net_j) = \frac{1}{1 + e^{-z\_net_i}} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(10)$$

$$z_1 = \frac{1}{1 + e^{-0.2}} = 0.55 \quad ; z_2 = \frac{1}{1 + e^{-0.7}} = 0.67 \quad ; \quad z_3 = \frac{1}{1 + e^{-0.1}} = 0.52$$

Langkah 5 : Hitung keluaran unit output(y_k)

$$y\_net_k = w_{ko} + \sum_{j=1}^{p} z_j k_j \qquad \ldots\ldots\ldots\ldots\ldots\ldots(11)$$

Karena jaringan hanya memiliki satu unit output y, maka :

y_net_k = -0.1 + 0.55(0.5) + 0.67(-0.3) + 0.52(-0.4) = -0.24

$$y_k = f(y\_net_k) = \frac{1}{1 + e^{-y\_net_k}} \qquad \text{……(12)}$$

$$y_k = \frac{1}{1 + e^{0.24}} = 0.44$$

Tahap II : Umpan Mundur (*backward propagation*)

Langkah 6 : Hitung faktor δ di unit output y_k

$$\delta_k = (t_k - y_k)\, f'(y\_net_k) = (t_k - y_k)\, y_k(1-y_k) \qquad \text{……(13)}$$

Karena jaringan hanya memiliki satu buah unit output, maka :

$\delta_k = (t - y)\, y(1-y) = (0-0.44)(0.44)(1-0.44) = -0.11$

Suku perubahan bobot $w_{kj} = w_{jk}$ (dengan $\alpha = 0.2$) :

$$\Delta W_{kj} = \alpha\, \delta_k\, z_j \qquad \text{dengan } j = 0,1,\ldots,3 \qquad \text{……(14)}$$

$\Delta W_{10} = 0.2(-0.11)(1) = -0.02$

$\Delta W_{11} = 0.2(-0.11)(0.55) = -0.01$

$\Delta W_{12} = 0.2(-0.11)(0.67) = -0.01$

$\Delta W_{13} = 0.2(-0.11)(0.52) = -0.01$

Langkah 7 : Hitung penjumlahan kesalahan dari unit tersembunyi (δ)

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k w_{kj} \qquad \text{……(16)}$$

Karena jaringan hanya memiliki satu buah unit output, maka :

$\delta\_net_1 = (-0.11)(0.5) = -0.05$

$\delta\_net_1 = (-0.11)(-0.3) = 0.03$

$\delta\_net_1 = (-0.11)(-0.4) = 0.04$

Faktor kesalahan δ di unit tersembunyi :

$$\delta_j = \delta\_net_j \, f'(\delta\_net_j) = \delta\_net_j \, z_j(1-z_j) \qquad \text{.....(17)}$$

$\delta_1 = -0.05(0.55)(1-0.55) = -0.01$

$\delta_2 = 0.03(0.67)(1-0.67) = 0.01$

$\delta_3 = 0.04(0.52)(1-0.52) = 0.01$

Suku perubahan bobot ke unit tersembunyi :

$$\Delta V_{ji} = \alpha \, \delta_j \, x_i \qquad (j = 1,2,3,\ldots; i = 0,1,2) \qquad \text{.....(18)}$$

Ditunjukkan pada tabel berikut :

Tabel II.3 Tabel suku perubahan bobot ke unit tersembunyi

	z_1	z_2	z_3
x_1	$\Delta V_{11} = (0.2)(-0.01)(1) = 0$	$\Delta V_{21} = (0.2)(0.01)(1) = 0$	$\Delta V_{31} = (0.2)(0.01)(1) = 0$
x_2	$\Delta V_{12} = (0.2)(-0.01)(1) = 0$	$\Delta V_{22} = (0.2)(0.01)(1) = 0$	$\Delta V_{32} = (0.2)(0.01)(1) = 0$
1	$\Delta V_{10} = (0.2)(-0.01)(1) = 0$	$\Delta V_{20} = (0.2)(0.01)(1) = 0$	$\Delta V_{30} = (0.2)(0.01)(1) = 0$

Tahap III : Peng-update-an bobot dan bias

Langkah 8 : Hitung semua perubahan bobot

Perubahan bobot unit output dengan $w_{jk} = w_{kj}$:

$$W_{kj}\,(\text{baru}) = W_{kj}(\text{lama}) + \Delta W_{kj} \qquad \text{.....(20)}$$

$W_{11}\,(\text{baru}) = 0.5 - 0.01 = 0.49$

$W_{12}\,(\text{baru}) = -0.3 - 0.01 = -0.31$

$W_{13}\,(\text{baru}) = -0.4 - 0.01 = -0.41$

$W_{14}\,(\text{baru}) = -0.1 - 0.02 = -0.12$

Perubahan bobot unit tersembunyi :

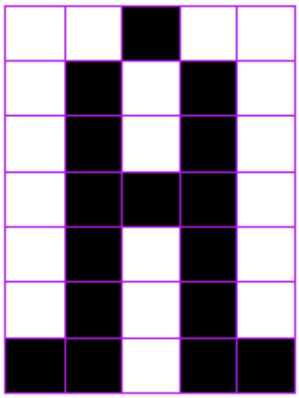
$$V_{ji}(\text{baru}) = V_{ji}(\text{lama}) + \Delta V_{ji}(j = 1,2,3 \; ; \; i = 0,1,2) \qquad \text{.....(21)}$$

Tabel II.4 Tabel perubahan bobot input ke unit hidden

	z_1	z_2	z_3
x_1	V_{11}(baru) = 0.2 + 0 = 0.2	V_{21}(baru) = 0.3 + 0 = 0.3	V_{31}(baru) = -0.1 + 0 = - 0.1
x_2	V_{12}(baru) = 0.3 + 0 = 0.3	V_{22}(baru) = 0.1 + 0 = 0.1	V_{32}(baru) = -0.1 + 0 = - 0.1
1	V_{10}(baru) = -0.3 + 0 = -0.3	V_{20}(baru) = 0.3 + 0 = 0.3	V_{30}(baru) = 0.3 + 0 = 0.3

Contoh 2 :

Diberikan pola huruf A yang direpresentasikan menggunakan kode 0 dan 1 pada matriks berukuran 5x7 seperti pada gambar berikut :



0 0 1 0 0
0 1 0 1 0
0 1 0 1 0
0 1 1 1 0
0 1 0 1 0
0 1 0 1 0
1 1 0 1 1

Penyelesaian :

Langkah 0 : Mula-mula bobot diberi nilai acak yang kecil (range [-1,1]). Misalkan di dapat bobot seperti tabel II.5 (bobot dari layer input ke layer tersembunyi = v_{ij}) dan II.6 (bobot dari layer tersembunyi ke layer output = w_{kj}).

Tabel II.5 Tabel bobot dari layer input ke layer tersembunyi

	z_1	z_2
x_1	0	0
x_2	0	0
x_3	0.1	-0.1
x_4	0	0
x_5	0	0
x_6	0	0
x_7	-0.2	0.2
x_8	0	0
x_9	0.3	0.2

x_{10}	0	0
x_{11}	0	0
x_{12}	0.3	-0.3
x_{13}	0	0
x_{14}	0.1	0.2
x_{15}	0	0
x_{16}	0	0
x_{17}	0.3	0.2
x_{18}	-0.2	-0.1
x_{19}	0.1	0.1
x_{20}	0	0
x_{21}	0	0
x_{22}	-0.1	-0.1
x_{23}	0	0
x_{24}	-0.2	-0.2
x_{25}	0	0
x_{26}	0	0
x_{27}	0.2	0.2
x_{28}	0	0
x_{29}	-0.2	-0.1
x_{30}	0	0
x_{31}	0.2	0.3
x_{32}	0.2	-0.1
x_{33}	0	0
x_{34}	-0.1	-0.2
x_{35}	0.2	0.1
B	0	0

Tabel II.6 Tabel bobot dari layer tersembunyi ke layer output

	y
z_1	0.3
z_2	-0.2
b	0.1

Langkah 1 : Jika kondisi penghentian belum terpenuhi, lakukan langkah 2-8

Langkah 2 : Untuk setiap pasang data pelatihan, lakukan langkah 3-8

Tahap I : Umpan Maju (*feedforward)*

Langkah 3 : Setiap unit input mengirim sinyal ke unit tersembunyi

Langkah 4 : hitung keluaran di unit tersembunyi (z_j)

$$z\_net_j = v_{jo} + \sum_{i=1}^{n} x_i v_{ji} \qquad \text{……(9)}$$

$z\_net1 = 0 + 0(0) + 0(0) + 1(0.1) + 0(0) + 0(0) + 0(0) + 1(-0.2) + 0(0) + 1(0.3) + 0(0) + 0(0) + 1(0.3) + 0(0) + 1(0.1) + 0(0) + 0(0) + 1(0.3) + 1(-0.2) + 1(0.1) + 0(0) + 0(0) + 1(-0.1) + 0(0) + 1(-0.2) + 0(0) + 0(0) + 1(0.2) + 0(0) + 1(-0.2) + 0(0) + 1(0.2) + 1(0.2) + 0(0) + 1(-0.1) + 1(0.2) = 1$

$z\_net2 = 0 + 0(0) + 0(0) + 1(-0.1) + 0(0) + 0(0) + 0(0) + 1(0.2) + 0(0) + 1(0.2) + 0(0) + 0(0) + 1(-0.3) + 0(0) + 1(0.2) + 0(0) + 0(0) + 1(0.2) + 1(-0.1) + 1(0.1) + 0(0) + 0(0) + 1(-0.1) + 0(0) + 1(-0.2) + 0(0) + 0(0) + 1(0.2) + 0(0) + 1(-0.1) + 0(0) + 1(0.3) + 1(-0.1) + 0(0) + 1(-0.2) + 1(0.1) = 0.3$

$$z_j = f(z\_net_j) = \frac{1}{1 + e^{-z\_net_i}} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(10)$$

$$z_1 = \frac{1}{1 + e^{-1}} = 0.73559 \; ; \qquad z_2 = \frac{1}{1 + e^{-0.3}} = 0.576141$$

Langkah 5 : Hitung keluaran unit output(y_k)

$$y\_net_k = w_{ko} + \sum_{j=1}^{p} z_j k_j \qquad \ldots\ldots\ldots\ldots\ldots\ldots(11)$$

Karena jaringan hanya memiliki satu unit output y, maka :

$y\_net_k = 0.1 + 0.73559\ (0.3) + 0.576141\ (-0.2) = 0.205449$

$$y_k = f(y\_net_k) = \frac{1}{1 + e^{-y\_net_k}} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(12)$$

$$y_k = \frac{1}{1 + e^{-0.205449}} = 0.55236$$

Tahap II : Umpan Mundur (*backward propagation*)

Langkah 6 : Hitung faktor δ di unit output y_k

$$\delta_k = (t_k - y_k)\, f'(y\_net_k) = (t_k - y_k)\, y_k(1-y_k) \qquad \ldots\ldots\ldots\ldots\ldots\ldots(13)$$

Karena jaringan hanya memiliki satu buah unit output, maka :

$\delta_k = (t- y)\ y(1-y) = (1-0.55236)(\ 0.55236)(1-0.55236) = 0.110683$

Suku perubahan bobot $w_{kj} = w_{jk}$ (dengan $\alpha = 0.2$) :

$$\Delta W_{kj} = \alpha\ \delta_k\ z_j \qquad \text{dengan } j = 0,1,\ldots,3 \qquad \ldots\ldots\ldots\ldots\ldots(14)$$

$\Delta W_{10} = 1(0.110683)(0.1) = 0.011068$

$\Delta W_{11} = 1(0.110683)(0.73559) = 0.081417$

$\Delta W_{12} = 1(0.110683)(0.576141) = 0.063769$

Langkah 7 : Hitung penjumlahan kesalahan dari unit tersembunyi (δ)

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k w_{kj} \qquad \ldots\ldots\ldots\ldots\ldots(16)$$

Karena jaringan hanya memiliki satu buah unit output, maka :

$\delta\_net_1 = (0.110683)(0.3) = 0.033205$

$\delta\_net_2 = (0.110683)(-0.2) = -0.02214$

Faktor kesalahan δ di unit tersembunyi :

$$\delta_j = \delta\_net_j\ f'(\delta\_net_j) = \delta\_net_j\ z_j(1-z_j) \qquad \ldots\ldots\ldots\ldots..(17)$$

$\delta_1 = 0.033205\ (0.73559)(1-0.73559) = 0.006973$

$\delta_2 = -0.02214\ (0.576141)(1-0.576141) = 0.005313$

Suku perubahan bobot ke unit tersembunyi :

$$\Delta V_{ji} = \alpha\ \delta_j\ x_i \qquad (j = 1,2,3,\ldots;\ i = 0,1,2\ ) \qquad \ldots\ldots\ldots\ldots..(18)$$

Ditunjukkan pada tabel berikut :

Tabel II.7 Tabel suku perubahan bobot ke unit tersembunyi

	z_1	z_2
x_1	0	0
x_2	0	0
x_3	0.006973	0.005313
x_4	0	0

x_5	0	0
x_6	0	0
x_7	0.006973	0.005313
x_8	0	0
x_9	0.006973	0.005313
x_{10}	0	0
x_{11}	0	0
x_{12}	0.006973	0.005313
x_{13}	0	0
x_{14}	0.006973	0.005313
x_{15}	0	0
x_{16}	0	0
x_{17}	0.006973	0.005313
x_{18}	0.006973	0.005313
x_{19}	0.006973	0.005313
x_{20}	0	0
x_{21}	0	0
x_{22}	0.006973	0.005313
x_{23}	0	0
x_{24}	0.006973	0.005313
x_{25}	0	0
x_{26}	0	0
x_{27}	0.006973	0.005313
x_{28}	0	0
x_{29}	0.006973	0.005313
x_{30}	0	0
x_{31}	0.006973	0.005313
x_{32}	0.006973	0.005313
x_{33}	0	0
x_{34}	0.006973	0.005313
x_{35}	0.006973	0.005313
b	0	0

Tahap III : Peng-update-an bobot dan bias

Langkah 8 : Hitung semua perubahan bobot

Perubahan bobot unit output dengan $w_{jk} = w_{kj}$:

$$W_{kj}\,(\text{baru}) = W_{kj}(\text{lama}) + \Delta W_{kj} \qquad \ldots\ldots\ldots\ldots\ldots(20)$$

$W_{11}\,(\text{baru}) = 0.3 + 0.081417 = 0.0381417$

$W_{12}\,(\text{baru}) = -0.2 + 0.063769 = -0.13623$

Perubahan bobot unit tersembunyi :

$$V_{ji}(\text{baru}) = V_{ji}(\text{lama}) + \Delta V_{ji}(j = 1,2,3 \; ; \; i = 0,1,2) \qquad \ldots\ldots\ldots\ldots(21)$$

Tabel II.8 Tabel perubahan bobot input ke unit hidden

	z_1	z_2
x_1	0	0
x_2	0	0
x_3	0.106973	-0.09469
x_4	0	0
x_5	0	0
x_6	0	0
x_7	-0.19303	0.205313
x_8	0	0
x_9	0.306973	0.205313
x_{10}	0	0
x_{11}	0	0
x_{12}	0.306973	-0.29469
x_{13}	0	0
x_{14}	0.106973	0.205313
x_{15}	0	0
x_{16}	0	0
x_{17}	0.306973	0.205313
x_{18}	-0.19303	-0.09469
x_{19}	0.106973	0.105313
x_{20}	0	0
x_{21}	0	0
x_{22}	-0.09303	-0.09469
x_{23}	0	0
x_{24}	-0.19303	-0.19469
x_{25}	0	0
x_{26}	0	0
x_{27}	0.206973	0.205313
x_{28}	0	0
x_{29}	-0.19303	-0.09469
x_{30}	0	0
x_{31}	0.206973	0.305313
x_{32}	0.206973	-0.09469
x_{33}	0	0
x_{34}	-0.09303	-0.19469
x_{35}	0.206973	0.105313
b	0	0

DAFTAR PUSTAKA

[JAE03] Jaeger, S., Liu, C.L., Nakagawa, M. (2003). *The State of The Art in Japanese Online Handwriting Recognition Compared to Techniques in Western Handwriting Recognition*. IJDAR (2003) vol. 6 pp.75-78.

[JON04] Jong Jek Siang, M.Sc, Drs. (2004). *Jaringan Saraf Tiruan & Pemrogramannya Menggunakan Matlab.* Penerbit Andi. Yogjakarta.

[KUS03] Kusumadewi, Sri. (2003). *Artificial Intelligence (Teknik dan Aplikasinya).* Penerbit Graha Ilmu. Yogyakarta.

[PUS06] Puspitaningrum, Diyah. (2006). *Pengantar Jaringan Saraf Tiruan*. Penerbit Andi. Jogjakarta.

[PUR08] Purwanto. (2008). *Sistem Pengenalan Huruf Tulisan Tangan dengan Pendekatan Heuristik.* http://rmsui.vlsm.org/fusilkom-ui/fusilkom-96-196abs.html.(Akses 25 Mei 2009)

[TRI08] Trilaksono, Mirza. (2008). *Implementasi Optical Character Recognition (OCR) dengan Pendekatan Metode Struktur Menggunakan Ekstraksi Ciri Vektor dan Region.* http:// www.ittelkom.ac.id//library/index.php (Akses 25 Mei 2009)

[YAN05] Yani, Eli. (2005). *Pengantar Jaringan Saraf Tiruan*. Artikel kuliah. http://trirezqiariantoro.files.wordpress.com/2007/05/jaringan_syaraf_tiruan.pdf (Akses 25 Mei 2009)