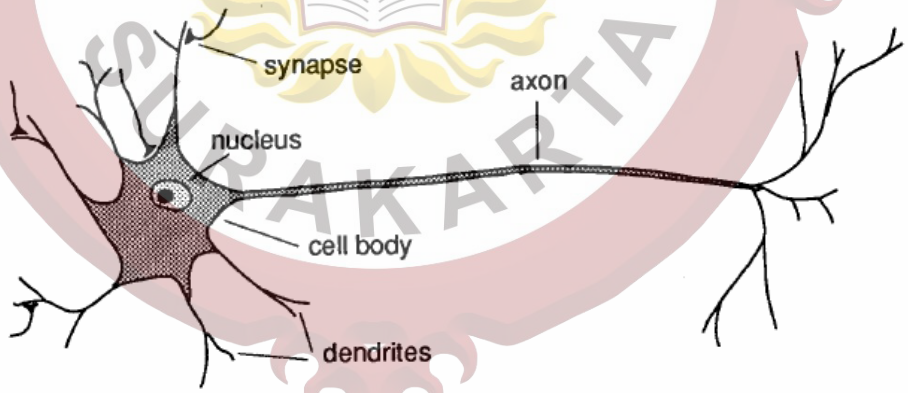
BAB II
LANDASAN TEORI

A. Pengertian Dasar

1. Definisi Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan keluar dari penelitian kecerdasan buatan, terutama percobaan untuk menirukan *fault-tolerence* dan kemampuan untuk belajar dari sistem syaraf biologi dengan model struktur *low-level* dari otak (Yani, 2005).

Elemen yang paling mendasar dari jaringan syaraf adalah sel syaraf. Sel-sel syaraf inilah membentuk bagian kesadaran manusia yang meliputi beberapa kemampuan umum. Pada dasarnya sel syaraf biologi menerima masukan dari sumber yang lain dan mengkombinasikannya dengan beberapa cara, melaksanakan suatu operasi yang non-linear untuk mendapatkan hasil dan kemudian mengeluarkan hasil akhir tersebut.



Gambar 2.1. Sel Syaraf Biologis

Dalam tubuh manusia terdapat banyak variasi tipe dasar sel syaraf, sehingga proses berpikir manusia menjadi sulit untuk direplikasi secara elektrik. Sekalipun demikian, semua sel syaraf alami mempunyai empat komponen dasar yang sama. Keempat komponen dasar ini diketahui berdasarkan nama biologinya yaitu, dendrit, soma, akson, dan sinapsis. Dendrit merupakan suatu perluasan dari

soma yang menyerupai rambut dan bertindak sebagai saluran masukan. Saluran masukan ini menerima masukan dari sel syaraf lainnya melalui sinapsis. Soma dalam hal ini kemudian memproses nilai masukan menjadi sebuah output yang kemudian dikirim ke sel saraf lainnya melalui akson dan sinapsis.

Penelitian terbaru memberikan bukti lebih lanjut bahwa sel syaraf biologi mempunyai struktur yang lebih kompleks dan lebih canggih daripada sel syaraf buatan yang kemudian dibentuk menjadi jaringan syaraf buatan yang ada sekarang ini. Ilmu biologi menyediakan suatu pemahaman yang lebih baik tentang sel syaraf sehingga memberikan keuntungan kepada para perancang jaringan untuk dapat terus meningkatkan sistem jaringan syaraf buatan yang ada berdasarkan pada pemahaman terhadap otak biologi.

Sel-sel syaraf ini terhubung satu dengan yang lainnya melalui sinapsis. Sel syaraf dapat menerima rangsangan berupa sinyal elektrokimiawi dari sel-sel syaraf yang lain. Berdasarkan rangsangan tersebut, sel syaraf akan mengirimkan sinyal atau tidak berdasarkan kondisi tertentu. Konsep dasar semacam inilah yang ingin dicoba para ahli dalam menciptakan sel syaraf tiruan (Wikipedia, 2007).

Jaringan Syaraf Tiruan adalah paradigma pemrosesan suatu informasi yang terinspirasi oleh sistem sel syaraf biologi, sama seperti otak yang memproses suatu informasi. Elemen mendasar dari paradigma tersebut adalah struktur yang baru dari sistem pemrosesan informasi. Jaringan Syaraf Tiruan, seperti manusia, belajar dari suatu contoh. Jaringan Syaraf Tiruan dibentuk untuk memecahkan suatu masalah tertentu seperti pengenalan pola atau klasifikasi karena proses pembelajaran.

Sistem jaringan syaraf tiruan merupakan suatu sistem yang memiliki pengetahuan dalam menganalisa suatu masalah dan melakukan pekerjaan-pekerjaan klasifikasi pola, pemodelan sistem dan memori asosiasi. Klasifikasi pola digunakan untuk menganalisis pola-pola masukkan dengan cara mencari kemiripan pola-pola masukan. Pemodelan sistem digunakan untuk pembuat simulasi sistem yang mampu menghasilkan keluaran dari suatu pola masukan

yang akan disimulasikan. Sedangkan memori asosiasi digunakan untuk menganalisis pola masukan yang tidak lengkap, misalnya pola masukan memiliki derau, terpotong-potong, rusak dan hanya bisa tampil sebagian (Yani, 2005).

2. Model Dasar Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan syaraf biologi.

JST dibentuk sebagai generalisasi model matematika dari jaringan syaraf biologi, dengan asumsi bahwa :

1. Pemrosesan informasi terjadi pada banyak elemen sederhana (*neuron*)
2. Sinyal dikirimkan di antara *neuron-neuron* melalui penghubung-penghubung
3. Penghubung antar *neuron* memiliki bobot yang akan memperkuat atau memperlemah sinyal

Untuk menentukan output, setiap *neuron* menggunakan fungsi aktivasi (biasanya bukan fungsi linier) yang dikenakan pada jumlahan input yang diterima. Besarnya output ini selanjutnya dibandingkan dengan suatu ambang batas.

JST ditentukan oleh tiga hal :

1. Pola hubungan antar *neuron* (disebut arsitektur jaringan)
2. Metode untuk menentukan bobot penghubung (disebut metode *training*/*learning*/algoritma)
3. Fungsi aktivasi

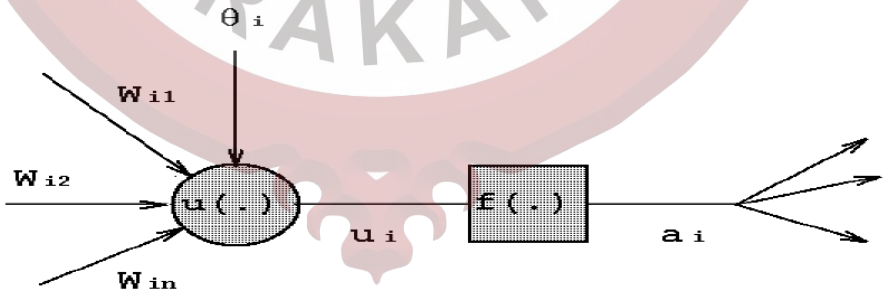
Neuron adalah unit pemroses informasi yang menjadi dasar dalam pengoperasian jaringan syaraf tiruan. *Neuron* terdiri atas tiga elemen pembentuk :

1. Himpunan unit –unit yang dihubungkan dengan jalur koneksi. Jalur-jalur tersebut memiliki bobot/kekuatan yang berbeda-beda. Bobot yang bernilai positif akan memperkuat sinyal dan yang negatif akan memperlemah sinyal yang dibawanya. Jumlah, struktur, dan pola hubungan antar unit-unit tersebut akan menentukan arsitektur jaringan (dan juga model jaringan yang akan terbentuk).

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 ataukah tidak (Siang, 2005).

Syaraf tiruan digambarkan sebagai berikut : Menerima input atau masukan (baik dari data yang dimasukkan atau dari output sel syaraf pada jaringan syaraf). Setiap input datang melalui suatu koneksi atau hubungan yang mempunyai sebuah bobot *(weight*). Setiap sel syaraf mempunyai sebuah nilai ambang. Jumlah bobot dari input dan dikurangi dengan nilai ambang kemudian akan mendapatkan suatu aktivasi dari sel syaraf (*post synaptic potential*, PSP, dari sel syaraf). Sinyal aktivasi kemudian menjadi fungsi aktivasi / fungsi transfer untuk menghasilkan output dari sel syaraf.

Jika tahapan fungsi aktivasi digunakan (output sel syaraf = 0 jika input <0 dan 1 jika input >= 0) maka tindakan sel syaraf sama dengan sel syaraf biologi yang dijelaskan diatas (pengurangan nilai ambang dari jumlah bobot dan membandingkan dengan 0 adalah sama dengan membandingkan jumlah bobot dengan nilai ambang). Biasanya tahapan fungsi jarang digunakan dalam Jaringan Syaraf Tiruan. Fungsi aktivasi (f(.)) dapat dilihat pada gambar 2.2.



Gambar 2.2. Fungsi Aktivasi

Bagaimana sel syaraf saling berhubungan? Jika suatu jaringan ingin digunakan untuk berbagai keperluan maka harus memiliki input (akan membawa nilai dari suatu variabel dari luar) dan output (dari prediksi atau

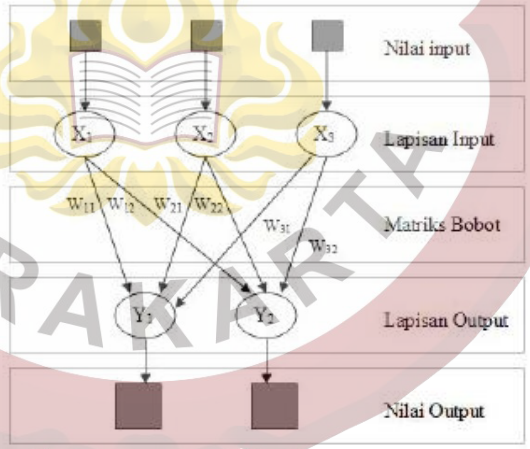
signal kontrol). Input dan output sesuai dengan sensor dan syaraf motorik seperti signal datang dari mata kemudian diteruskan ke tangan, Dalam hal ini terdapat sel syaraf atau neuron pada lapisan tersembunyi berperan pada jaringan ini. Input, lapisan tersembunyi dan output sel syaraf diperlukan untuk saling terhubung satu sama lain (Yani, 2005).

3. Arsitektur Jaringan

Beberapa arsitektur jaringan yang sering dipakai dalam jaringan syaraf tiruan antara lain :

a. Jaringan lapisan tunggal (*single layer network*)

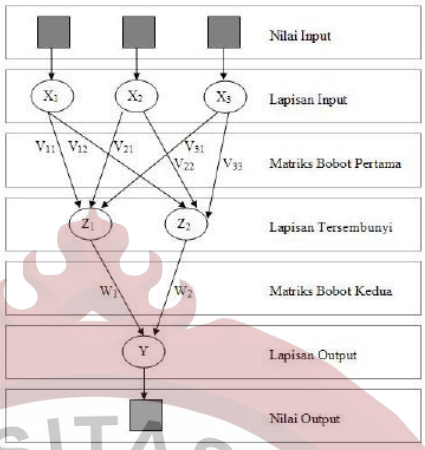
Dalam jaringan ini, sekumpulan *input neuron* dihubungkan langsung dengan sekumpulan outputnya. Dalam beberapa model (misal *perceptron*), hanya ada sebuah unit *neuron* output.



Gambar 2.3. Jaringan lapisan tunggal

b. Jaringan lapisan jamak (*multilayer network*)

Jaringan lapisan jamak merupakan perluasan dari lapisan tunggal. Dalam jaringan ini, selain unit input dan output, ada unit-unit lain (sering disebut layar tersembunyi). Dimungkinkan pula ada beberapa lapisan tersembunyi. Sama seperti pada unit input dan output, unit-unit dalam satu lapisan tidak saling berhubungan.



Gambar 2.4. Jaringan lapisan jamak

Jaringan lapisan jamak dapat menyelesaikan masalah yang lebih kompleks dibandingkan dengan lapisan tunggal, meskipun kadangkala proses pelatihan lebih kompleks dan lama.

c. Jaringan *reccurent*

Model jaringan *reccurent* mirip dengan jaringan lapisan tunggal ataupun ganda. Hanya saja, ada *neuron* output yang memberikan sinyal pada unit input (sering disebut *feedback loop*).

4.Fungsi Aktivasi

Dalam jaringan syaraf tiruan, fungsi aktivasi dipakai untuk menentukan keluaran suatu *neuron*. Argumen fungsi aktivasi adalah net masukan (kombinasi linier masukan dan bobotnya).

Beberapa fungsi aktivasi yang sering dipakai adalah sebagai berikut :

a. Fungsi *threshold* (ambang batas)

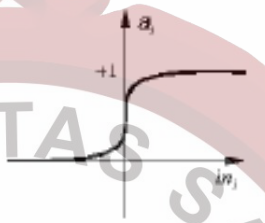
$$Y = \begin{cases} 0, & \text{jika } x < \theta \\ 1, & \text{jika } x \geq \theta \end{cases}$$

Untuk beberapa kasus, fungsi *threshold* yang dibuat tidak berharga 0 atau 1, tapi berharga -1 atau 1 (sering disebut *threshold* bipolar)

$$Y = \begin{cases} 1, & \text{jika } x \geq \theta \\ -1, & \text{jika } x < \theta \end{cases}$$

b. Fungsi sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$



Gambar 2.5. Fungsi sigmoid

Fungsi sigmoid sering dipakai karena nilai fungsinya yang terletak antara 0 dan 1 dan dapat diturunkan dengan mudah.

$f'(x) = f(x)\,(1 - f(x))$ (1)

c. Fungsi identitas

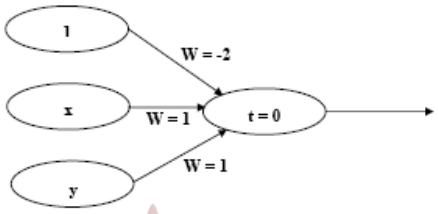
$f(x) = x$

Fungsi identitas sering dipakai apabila kita menginginkan keluaran jaringan berupa sembarang bilangan riil (bukan hanya pada range [0, 1] atau [-1, 1].

5.Bias

Kadang-kadang dalam jaringan ditambahkan sebuah unit masukan yang nilainya selalu = 1. Unit yang sedemikian itu disebut bias. Bias dapat dipandang sebagai sebuah input yang nilainya = 1.

$y_i = wx_i + b$ (2)



Gambar 2.6. Jaringan dengan satu bias pada *input layer*

6.Pelatihan dengan dan tanpa Supervisi

Berdasarkan cara memodifikasi bobotnya, ada dua macam pelatihan yang dikenal yaitu dengan supervisi (*supervised*) dan tanpa supervisi (*unsupervised*).

Dalam pelatihan dengan supervisi, terdapat sejumlah pasangan data (masukan – target keluaran) yang dipakai untuk melatih jaringan hingga diperoleh bobot yang diinginkan. Pasangan data tersebut berfungsi sebagai ”guru” untuk melatih jaringan hingga diperoleh bentuk yang terbaik. ”Guru” akan memberikan informasi yang jelas tentang bagaimana sistem harus mengubah dirinya untuk meningkatkan unjuk kerjanya.

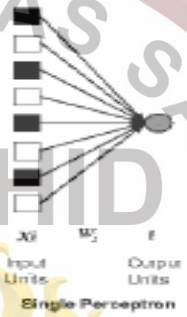
Pada setiap kali pelatihan, suatu input diberikan ke jaringan. Jaringan akan memproses dan mengeluarkan keluaran. Selisih antara keluaran jarinagn dengan target (keluaran yang diinginkan) merupakan kesalahan yang terjadi. Jaringan akan memodifikasi bobot sesuai dengan kesalahan tersebut. Jaringan *perceptron*, ADALINE dan *backpropagation* merupakan model-model yang menggunakan pelatihan dengan supervisi.

Sebaliknya, dalam pelatihan tanpa supervisi (*unsupervised learning*) tidak ada ”guru” yang akan mengarahkan proses pelatihan. Dalam pelatihannya, perubahan bobot jaringan dilakukan berdasarkan parameter tertentu dan jaringan dimodifikasi menurut ukuran parameter tersebut (Siang, 2005).

B. *Perceptron*

Model jaringan *perceptron* ditemukan oleh Rosenblatt (1962) dan Minsky – Papert (1969). Model tersebut merupakan model yang memiliki aplikasi dan pelatihan paling baik pada era tersebut.

Perceptron terdiri atas beberapa unit masukan (ditambah sebuah bias), dan memiliki sebuah unit keluaran. Hanya saja fungsi aktivasi bukan merupakan fungsi biner (atau bipolar), tetapi memiliki kemungkinan nilai -1, 0, atau 1.



Gambar 2.7. *Single perceptron*

Untuk sebuah harga *threshold* θ yang ditentukan :

$$f(net) = \begin{cases} 1 & \text{jika } net > \theta \\ 0 & \text{jika } \theta \leq net \leq \theta \\ -1 & \text{jika } net < -\theta \end{cases}$$

secara geometris, fungsi aktivasi membentuk dua garis sekaligus, masing-masing dengan persamaan :

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n + b = \theta \text{ dan} \quad (3)$$

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n + b = -\theta \quad (4)$$

C. ADALINE dan MADALINE

Model ADALINE (*Adaptive Linear Neuron*) ditemukan oleh Widrow & Hoff (1960). Arsitekturnya mirip dengan *perceptron*. Beberapa masukan (dan

sebuah bias) dihubungkan langsung dengan sebuah *neuron* keluaran. Perbedaan dengan *perceptron* adalah dalam cara modifikasi bobotnya. Bobot dimodifikasi dengan aturan delta (sering disebut juga *least mean square*). Selama pelatihan, fungsi aktivasi yang dipakai adalah fungsi identitas.

Kuadrat selisih antara target (t) dan keluaran jaringan (*f(net)*) merupakan *error* yang terjadi. Dalan aturan delta, bobot dimodifikasi sedemikian hingga *error*-nya minimum (Siang, 2005).

Beberapa ADALINE dapat digabungkan untuk membentuk suatu jaringan baru yang disebut MADALINE (*many* ADALINE). Dalam MADALINE terdapat sebuah layar tersembunyi. Adanya unit tersembunyi dalam MADALINE akan meningkatkan kapabilitas komputasi dibandingkan ADALINE, meskipun pelatihannya juga lebih kompleks (Siang, 2005).

D. *Backpropagation*

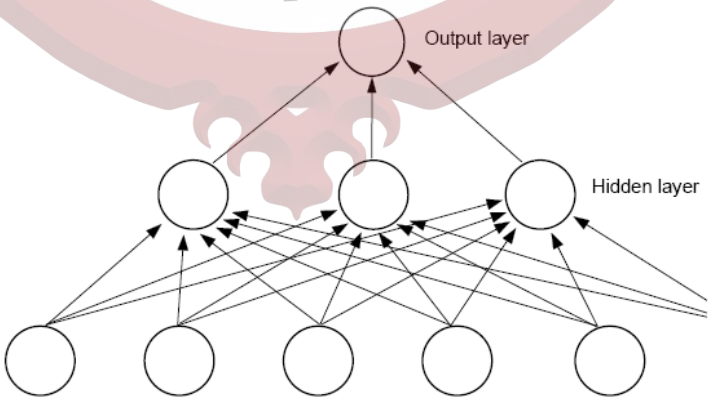
1. Algoritma *Backpropagation*

Di antara struktur jaringan syaraf tiruan yang berbeda, *Backpropagation* adalah yang paling terkenal karena kemampuannya dalam banyak bidang. Jaringan Syaraf Tiruan *Backpropagation* (BP) pertama kali diperkenalkan oleh Rumelhart, Hinton dan William pada tahun 1986, kemudian Rumelhart dan McClelland mengembangkannya pada tahun 1988. Untuk keperluan prakiraan, sebuah model BP yang khas terdiri atas satu lapisan input, satu atau dua *hidden layer*, dan satu *output layer* yang hanya memiliki satu *neuron*. Seperti yang ditunjukkan Gambar 2.8. adalah sebuah struktur khas yang paling sering digunakan dalam prakiraan. Dalam hal ini, *input layer* memiliki beberapa *neuron*, setiap *neuron* mewakili satu variabel input. *Hidden layer* juga memiliki beberapa *neuron* dan merepresentasikan non-linearitas sistem jaringan. *Output layer* hanya memiliki satu *neuron* yang merepresentasikan nilai prakiraan yang sesuai dengan set nilai input. Prinsipnya, sebuah jaringan syaraf tiruan BP memiliki beberapa *hidden layer*, tapi dalam prakteknya, hanya satu atau dua

hidden layer digunakan. Jumlah *neuron* pada *hidden layer* ditentukan terutama dengan *trial and error*.

BP adalah sebuah metode pelatihan (kalibrasi) jaringan syaraf multilayer yang sistematis. BP menggunakan satu set pasangan nilai input dan output (disebut pola). Sebuah pola input diumpankan ke jaringan untuk menghasilkan sebuah output, yang kemudian dibandingkan dengan pola output aktual. Jika tidak ada perbedaan antara output jaringan dan output aktual, maka pembelajaran tidak diperlukan. Jika sebaliknya, bobot – yang mengekspresikan kontribusi *input neuron* kepada *hidden neuron*, dan *hidden neuron* kepada output – diubah (dengan arah terbalik dari *output layer* ke *input layer*). Karena pelatihan tersebut menggunakan output aktual, metode BP disebut juga metode pelatihan dengan supervisi (Danh, 1999).

Jadi algoritma BP terdiri atas tiga fase. Fase pertama adalah fase maju, yaitu pola input dihitung maju mulai dari lapisan input hingga lapisan output menggunakan fungsi aktivasi yang ditentukan. Fase kedua adalah fase mundur, yaitu berdasarkan selisih antara output jaringan dengan target yang diinginkan dipropagasikan mundur mulai dari garis yang berhubungan langsung dengan lapisan output. Fase ketiga adalah modifikasi bobot untuk menurunkan *error* yang terjadi.



Gambar 2.8. JST *backpropagation* dengan satu *hidden layer*

Proses matematis pada fase maju :

a. Masing-masing unit masukan (X_i , i = 1,..,n) menerima sinyal masukan X_i dan sinyal tersebut disebarkan ke unit-unit berikutnya (unit-unit lapisan tersembunyi)

b. Masing-masing unit di lapisan tersembunyi dikalikan dengan bobot dan dijumlahkan serta ditambah dengan biasnya:

$$z\_in = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \tag{5}$$

Kemudian menghitung sesuai dengan fungsi aktivasi yang digunakan:

$$z_j = f(z\_in_j) \tag{6}$$

Bila yang digunakan adalah fungsi sigmoid, maka bentuk fungsi tersebut adalah:

$$Z_j = \frac{1}{1+\exp^{(-z\_in_j)}} \tag{7}$$

Kemudian mengirim sinyal tersebut ke semua unit keluaran.

c. Masing-masing unit keluaran (y_k, k = 1,2,3,..,m) dikalikan dengan bobot dan dijumlahkan:

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \tag{8}$$

Menghitung kembali sesuai dengan fungsi aktivasi

$$y_k = f(y\_in_k) \tag{9}$$

Fase mundur :

a. Masing-masing unit keluaran (Y_k, k = 1,..,m) menerima pola target sesuai dengan pola masukan saat pelatihan dan dihitung galatnya:

$$\delta_k = (t_k - y_k)\, f'(y\_in_k) \tag{10}$$

Karena f' (y_in_k) = y_k menggunakan fungsi sigmoid, maka:

$$f'(y\_in_k) = f(y\_in_k)(1 - f(y\_in_k))$$
$$= y_k(1 - y_k) \quad (11)$$

Menghitung perbaikan bobot (kemudian untuk memperbaiki w_{jk}).

$$\Delta W_{kj} = \alpha . \partial_k . Z_j \quad (12)$$

Menghitung perbaikan bobot bias:

$$\Delta W_{0k} = \alpha . \partial_k \quad (13)$$

Dan menggunakan nilai δ_k pada semua unit lapisan sebelumnya (*hidden layer*).

b. Masing-masing bobot yang menghubungkan unit-unit lapisan keluaran dengan unit-unit pada lapisan tersembunyi (Z_j, j = 1,..,p) dikalikan delta dan dijumlahkan sebagai masukan ke unit-unit lapisan berikutnya.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk} \quad (14)$$

Selanjutnya dikalikan dengan turunan dari fungsi aktivasinya untuk menghitung galat.

$$\delta_j = \delta\_in_j f'(z\_in_j) \quad (15)$$

kemudian menghitung perbaikan bobot (digunakan untuk memperbaiki v_{ij})

$$\Delta v_{ij} = \alpha \delta_j x_i \quad (16)$$

kemudian menghitung perbaikan bias (untuk memperbaiki v_{0j})

$$\Delta v_{0j} = \alpha \delta_j \quad (17)$$

Perbaikan bobot dan bias :

Masing-masing keluaran unit (yk, k=1,..,m) diperbaiki bias dan bobotnya

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \quad (18)$$

masing-masing unit tersembunyi (Zj, j=1,..,p) diperbaiki bias dan bobotnya

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij} \quad (19)$$

Selanjutnya proses yang terjadi adalah proses iterasi dalam dengan syarat penghentian yang ditentukan (Fausett, 1994).

Keterangan :

X_i : unit ke-i pada lapisan masukan

x_i : nilai aktivasi dari unti X_i

Z_j : unit ke-j pada lapisan tersembunyi

Z_in_j : keluaran untuk unit Z_j

z_j : nilai aktivasi dari unit Z_j

Y_k : unit ke-k pada lapisan keluaran

Y_in_k : net masukan untuk unit Y_k

y_k : nilai aktivasi dari unit Y_k

w_{k0} : nilai bobot pada bias untuk unit Y_k

w_{kj} : nilai bobot dari Z_{ij} ke unit Y_k

Δw_{kj} : selisih antara $w_{kj}(t)$ dengan $w_{kj}(t+1)$

v_{i0} : nilai bobot pada bias untuk unit Z_i

v_{ij} : nilai bobot dari unit X_i ke unit Z_i

Δv_{ij} : selisih antara $v_{ij}(t)$ dengan $v_{ij}(t+1)$

δ_k : faktor pengaturan nilai bobot pada lapisan keluaran

δ_j : faktor pengaturan nilai bobot pada lapisan tersembunyi

α : konstanta laju pelatihan (*learning rate*) $0<\alpha<1$

Contoh perhitungan :

Backpropagation dengan sebuah layar tersembunyi (dengan 3 unit) untuk mengenali fungsi logika XOR dengan 2 masukan x_1 dan x_2. Iterasi pertama

untuk menghitung bobot jaringan untuk pola pertama ($x_1 = 1$, $x_2 = 1$ dan $t = 0$) dengan laju pemahaman $\alpha = 0.2$.

Mula-mula bobot diberi nilai acak yang kecil (range [-1, 1]). Misal didapat bobot seperti tabel 2.1 (bobot dari layar masukan ke layar tersembunyi = V_{ji}) dan 2.2 (bobot dari layar tersembunyi ke layar keluaran = W_{kj})

Tabel 2.1. Bobot dari layar masukan ke layar 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 2.2. bobot dari layar tersembunyi ke layar keluaran

	Z_1
z_1	0.5
z_2	-0.3
z_3	-0.4
1	-0.1

Hitung keluaran unit tersembunyi (z_j)

$$z\_net_j = v_{jo} + \sum_{i=1}^{2} x_i v_{ji}$$

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

z_net 2 = 0.3 +1 (0.3) + 1 (0.1) = 0.7

z_net 3 = 0.3 +1 (-0.1) + 1 (-0.1) = 0.1

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

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

Hitung keluaran unit y_k

$y\_net_k = w_{ko} + \sum_{i=1}^{3} z_i w_{kj} =$

Karena jaringan hanya memiliki sebuah unit keluaran y maka y_net_k

$y\_net = w_{1o} + \sum_{j=1}^{3} z_i w_{1j} = -0.1 + 0.55 (0.5) + 0.67 (-0.3) + 0.52 (-0.4) =$

-0.24

$y = f(y\_net) = \frac{1}{1 + e^{-y\_net}} = \frac{1}{1 + e^{.24}} = 0.44$

Hitung faktor δ di unit keluaran y_k

$\delta_k = (t_k - y_k) f^1(y\_net_k) = (t_k - y_k) y_k (1 - y_k)$. Karena Jaringan hanya memiliki sebuah keluaran maka $\delta_k = \delta = (t - y) y (1 - y) = (0-0.44) (0.44) (1-0.44) = -0.11$

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

$\Delta w_{kj} = \alpha \delta_k Z_j = \alpha \delta Z_j$; j = 0,1,..., 3

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

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

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

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

Hitung penjumlahan kesalahan dari unit tersembunyi (= δ)

$\delta\_\text{net}_j = \sum_{k=1}^{m} \delta_k w_{kj}$. Karena jaringan hanya memiliki sebuah unit keluaran maka $\delta\_net_j = \delta w_{1j}$.

δ_net_1 = (-0.11) (0.5) =-0.05

δ_net_2 = (-0.11) (-0.3) = 0.03

δ_net_3 = (-0.11) (-0.4) = 0.04

Faktor kesalahan δ di unit tersembunyi :

$\delta_j = \delta\_net_j \; f^1 (z\_net_j) = \delta\_net_j \; z_j (1 - z_j)$

δ_1 = -0.05 (0.55) (1 - 0.55) = -0.01

δ_2 = 0.03 (0.67) (1 - 0.67) = 0.01

δ_3 = 0.04 (0.52) (1 - 0.52) = 0.01

Suku perubahan bobot ke unit tersembunyi $\Delta v_{ji} = \alpha \; \delta_j \; x_i$ (j = 1,2,3 ; i = 0,l,2)

Tabel 2.3. Suku perubahan bobot ke unit tersembunyi

	Z_1	Z_2	Z_3
x_1	Δv_{11} = (0.2) (-0.01) (1) = 0	Δv_{21} = (0.2) (0.01) (1) = 0	Δv_{31} = (0.2) (0.01) (1) = 0
x_2	Δv_{12} = (0.2) (-0.01) (1) = 0	Δv_{22} = (0.2) (0.01) (1) = 0	Δv_{32} = (0.2) (0.01) (1) = 0
1	Δv_{10} = (0.2) (-0.01) (1) = 0	Δv_{20} = (0.2) (0.01) (1) = 0	Δv_{30} = (0.2) (0.01) (1) = 0

Hitung semua perubahan bobot

perubahan bobot unit keluaran :

$w_{kh}\,(baru) = w_{kh}\,(lama) + \Delta w_{kj}$ (k=1 ; j = 0,1,...,3)

w_{11} (baru) = 0.5-0.01 = 0.49

w_{12} (baru) = -0.3-0.01 = -0.31

w_{13} (baru) = -0.4-0.01 = -0.41

w_{12} (baru) = -0.1-0.02 = -0.12

Perubahan bobot unit tersembunyi :

v_{ji} (*baru*) = v_{ji} (*lama*) + Δv_{ji} (j = l,2,3 ; i = 0,l,2)

Tabel 2.4. Perubahan bobot unit tersembunyi

	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) = 0.2 + 0 = 0.3	v_{22} (baru) = 0.1 + 0 = 0.1	v_{32} (baru) = -0.1 + 0 = -0.1
1	v_{10} (baru) = 0.2 + 0 = -0.3	v_{20} (baru) = 0.3 + 0 = 0.3	v_{30} (baru) = 0.3 + 0 = 0.3

2. Pemrograman *Backpropagation* dengan MATLAB

Backpropagation dibentuk dengan membuat generalisasi aturan pelatihan dalam model Widrow-Hoff dengan cara menambahkan layar tersembunyi. Kata *backpropagation* merujuk pada cara bagaimana gradien perubahan bobot dihitung.

Standar *backpropagation* menggunakan algoritma penurunan gradien (*gradien descent*). Variasi terhadap model standar *backpropagation* dilakukan dengan mengganti algoritma penurunan gradien dengan metode optimasi yang lain.

Hasil percobaan menunjukkan bahwa *backropagation* yang sudah dilatih dengan baik akan memberikan keluaran yang masuk akal jika diberi masukan yang serupa (tidak harus sama) dengan pola yang dipakai dalam pelatihan. Sifat generalisasi ini membuat pelatihan lebih efisien karena tidak perlu dilakukan pada semua data (Siang, 2005).

Berikut tahap-tahap pemrograman *backpropagation* :

a. Inisialisasi Jaringan

Langkah pertama yang harus dilakukan untuk memprogram *backpropagation* dengan MATLAB adalah membuat inisialisasi jaringan. Perintah yang dipakai untuk membentuk jaringan adalah `newff` yang formatnya adalah sebagai berikut:

```
net = newff (PR, [S1 S2..SN], (TF1 TF2..TFN), BTF, BLF, PF)
```

dengan

`net` = jaringan *backpropagation* yang terdiri atas N layar

`PR` = matriks ordo Rx2 yang berisi nilai minimum dan maksimum R buah elemen masukannnya

`Si (i = 1, 2, …, n)` = jumlah unit pada layar ke-i

`TFi (i = 1, 2, …, n)` = fungsi aktivasi yang dipakai pada layar ke-i. *Default* = `tansig` (sigmoid bipolar)

`BTF` = fungsi pelatihan jaringan. *Default* = `traingdx`

`BLF` = fungsi perubahan bobot/bias. *Default* = `learngdm`

`PF` = fungsi perhitungan *error*. *Default* = `mse`

Beberapa fungsi aktivasi yang dipakai MATLAB dalam pelatihan *backpropagation* adalah:

a. `tansig` (sigmoid bipolar). $f(net) = \frac{2}{1 + e^{-net}} - 1$. Fungsi ini adalah *default* yang dipakai
b. Fungsi sigmoid bipolar memiliki *range* [-1,1].
c. `logsig` (sigmoid biner) $f(net) = \frac{1}{1 + e^{-net}}$. Fungsi sigmoid biner memiliki bentuk serupa dengan sigmoid bipolar, hanya *range*-nya adalah [0,1].
d. `purelin` (fungsi identitas) $f(net) = net$.

b. Inisialisasi Bobot

Setiap kali membentuk jaringan *backpropagation*, MATLAB akan memberi nilai bobot dan bias awal dengan bilangan awal kecil. Bobot dan bias ini akan berubah setiap kali kita membentuk jaringan. Akan tetapi jika diinginkan memberi bobot tertentu, kita bisa melakukannya dengan memberi nilai pada `net.IW`, `net.LW`, dan `net.b`.