BAB II

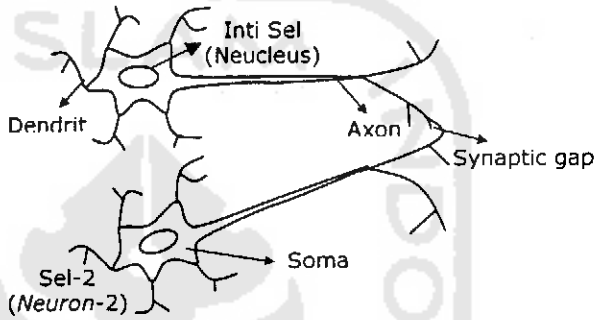
LANDASAN TEORI

Jaringan syaraf tiruan adalah merupakan salah satu representasi buatan dan otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplimentasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran.

Rober Hecht-Nielsend mendefinisikan jaringan syaraf tiruan sebagai berikut. Suatau neural network adalah struktur pemrosesan informasi yang terdistribusi dan bekerja secara paralel, yang terdiri dari elemen pemrosesan (yang memiliki memori lokal dan beroperasi dengan informasi lokal) yang diinterkoneksikan bersama dengan alur sinyal searah yang disebut koneksi. Setiap elemen pemroses memiliki koneksi keluarga tunggal yang bercabang (fanout) kesejumlah koneksi *collateral* yang diinginkan (setiap koneksi membawa sinyal yang sama dari keluaran elemen pemrosesan tersebut). Keluaran dari elemen pemrosesan tersebut dapat merupakan sebarang jenis persamaan matematis yang diinginkan. Seluruh proses yang berlangsung pada setiap elemen pemrosesan harus benar-benar dilakukan secara lokal yaitu keluaran hanya bergantung pada nilai masukan pada saat itu yang diperoleh melalui koneksi dan nilai yang tersimpan dalam memori lokal. [NIE90]

2.1 Sistem Syaraf Manusia

Otak manusia berisi berjuta-juta sel syaraf yang bertugas untuk memproses informasi. Tiap-tiap sel syaraf bekerja seperti suatu proses sederhana. Masing-masing sel tersebut saling berinteraksi untuk mendukung kemampuan kerja otak manusia.



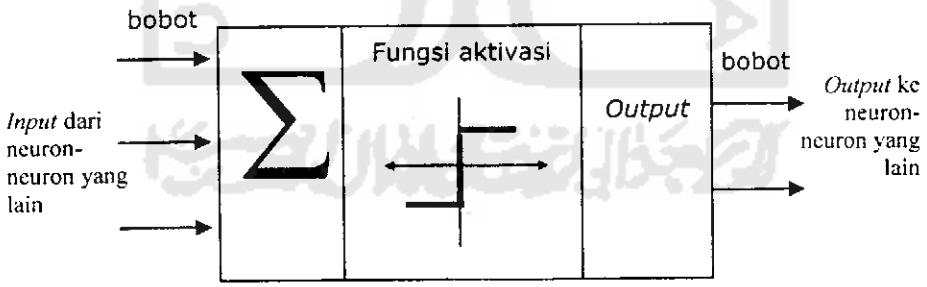
Gambar 2.1 Susunan *Neuron* Biologis

Gambar 2.1 menunjukan susunan syaraf pada manusia. Setiap sel syaraf (neuron) akan memiliki satu inti sel, inti sel ini nanti yang akan melakukan pemrosesan informasi. Informasi yang datang akan diterima oleh **dendrit**. Selain menerima informasi, dendrit yang menyertai **axon** sebagian keluaran dari suatu pemrosesan informasi. Informasi hasil olahan ini akan menjadi masukan bagi neuron lain yang mana antar dendrit kedua sel tersebut dipertemukan dengan **synapsis**. Informasi yang dikirim antar neuron ini sebagai mata rantai yang saling terhubung merupakan rangsangan yang dilewati melalui dendrit. Informasi yang datang dan diterima oleh dendrit akan dijumlahkan dan dikirim melalui axon ke dendrit akhir yang bersentuhan dengan dendrit dari neuron yang lain. Informasi ini akan diterima oleh neuron lain jika batas tertentu, yang sering dikenal dengan **nilai ambang** (*thershold*). Pada kasus ini, neuron tersebut dikatakan **teraktivasi**.

Hubungan antar neuron terjadi secara **adaptif**, artinya struktur hubungan tersebut terjadi secara dinamis. Otak manusia selalu memiliki kemampuan untuk belajar dengan melakukan adaptif.

2.2 Komponen Jaringan Syaraf Tiruan

Ada beberapa tipe jaringan syaraf, namun demikian, hampir semuanya memiliki komponen-komponen yang sama. Sepertinya halnya otak manusia, jaringan syaraf tiruan juga terdiri dari beberapa neuron, dan ada hubungan antar neuron tersebut. Neuron-neuron tersebut akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju ke neuron-neuron yang lain. Pada jaringan syaraf, hubungan ini dikenal dengan nama bobot. Informasi tersebut disimpan pada suatu nilai tertentu pada bobot tersebut. Gambar 2.2 menunjukan struktur neuron pada jaringan syaraf.



Gambar 2.2 Struktur neuron jaringan syaraf tiruan

Jika dilihat, neuron buatan ini sebenarnya mirip dengan sel neuron biologis. Neuron-neuron buatan tersebut bekerja dengan cara yang sama pula dengan neuron-neuron biologis. Informasi disebut dengan **input** akan dikirim ke neuron dengan bobot kedatangan tertentu. Input ini akan dioperasikan oleh suatu fungsi

perambatan yang akan menjumlahkan nilai-nilai semua bobot yang datang. Hasil penjumlahan ini kemudian akan dibandingkan dengan suatu nilai ambang (*thershold*) tertentu melalui fungsi aktifasi neuron. Apabila *input* tersebut melewati suatu nilai ambang tertentu, maka *neuron* tersebut akan diaktifkan. Apabila *neuron* tersebut diaktifkan, maka *neuron* tersebut akan mengirimkan keluaran (disebut dengan *output*) melalui bobot-bobot *output*nya ke semua *neuron* yang berhubungan dengannya, demikian seterusnya.

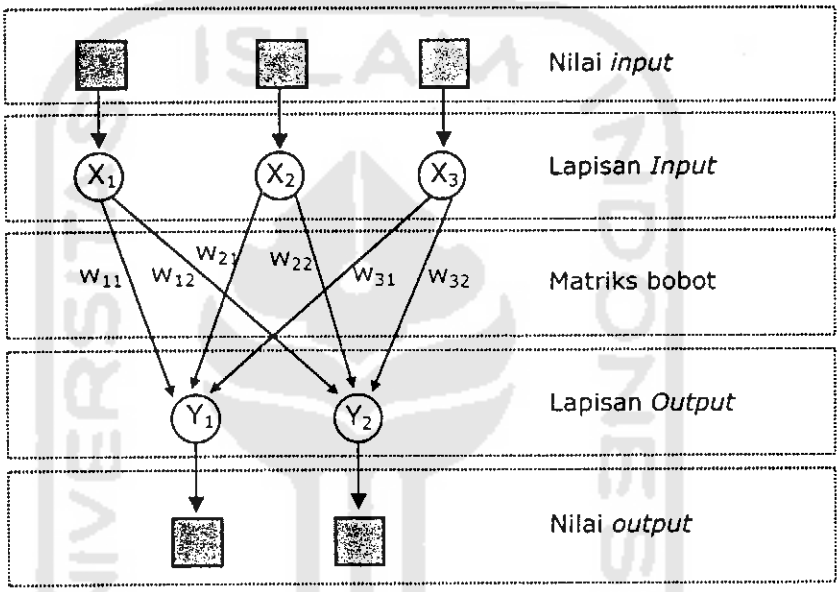
Pada jaringan syaraf, *neuron-neuron* akan dikumpulkan dalam lapisan-lapisan (*layers*) yang disebut dengan lapisan *neuron* (*neuron layers*). Biasanya *neuron-neuron* pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya (kecuali lapisan *input* dan *output*). Informasi yang diberikan pada jaringan syaraf akan dirambatkan lapisan ke lapisan, mulai dari lapisan *input* sampai ke lapisan *output* melalui lapisan yang lainnya yang sering dikenal dengan nama lapisan tersembunyi (*hidden layers*). Tergantung pada algoritma pembelajarannya, bisa jadi informasi tersebut dirambatkan secara mundur pada jaringan.

2.3 Arsitektur Jaringan Syaraf Tiruan

Seperti telah dijelaskan sebelumnya bahwa neuron-neuron dikelompokan dalam lapisan-lapisan. Umumnya, neuron-neuron yang terletak pada lapisan yang sama akan memiliki keadaan yang sama. Faktor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktifasi dan pola bobotnya. Ada beberapa arsitektur jaringan syaraf tiruan [KUS03], antara lain:

2.3.1 Jaringan dengan lapisan tunggal (single layer net)

Jaringan dengan lapisan tunggal hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi, seperti yang terlihat pada gambar 2.3.



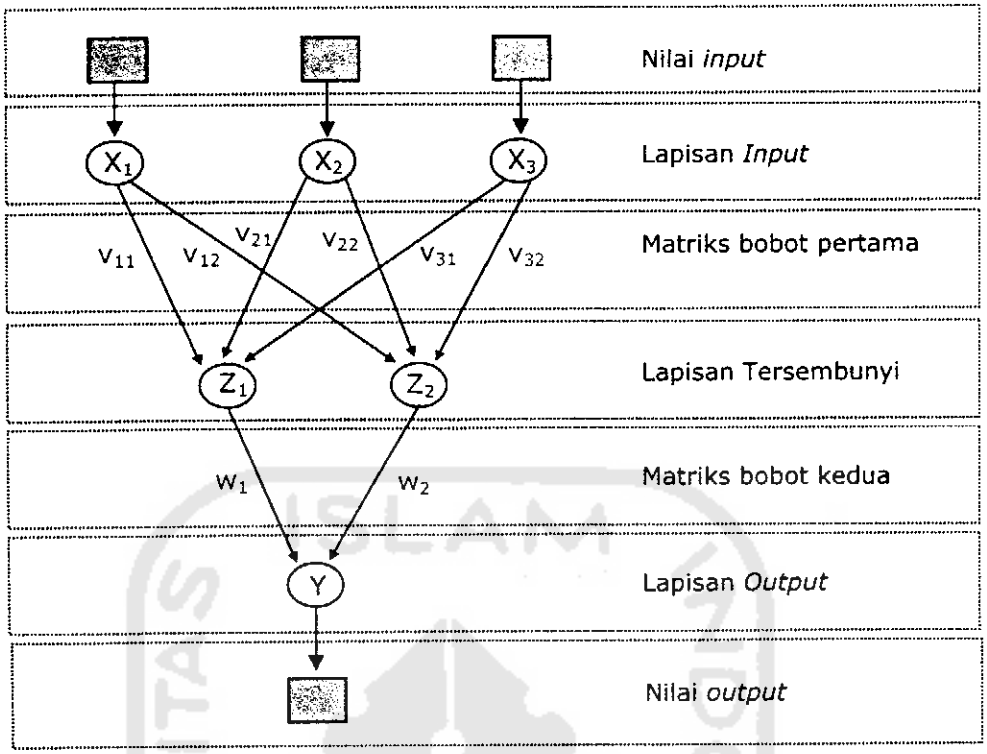
Gambar 2.3 Jaringan syaraf dengan lapisan tunggal

Pada Gambar 2.3 tersebut, lapisan *input* memiliki 3 neuron, yaitu X_1, X_2 dan X_3. Sedangkan pada lapisan *output* memiliki 2 neuron yaitu Y_1 dan Y_2. *Neuron-neuron* pada kedua lapisan saling berhubungan. Seberapa besar hubungan antara 2 *neuron* ditentukan oleh bobot yang bersesuaian. Semua unit *input* akan dihubungkan dengan setiap unit *output*.

2.3.2 Jaringan dengan banyak lapisan (multilayer net)

Jaringan dengan banyak lapisan memiliki 1 atau lebih lapisan yang terletak diantara lapisan *input* dan lapisan *output* (memiliki 1 atau lebih lapisan

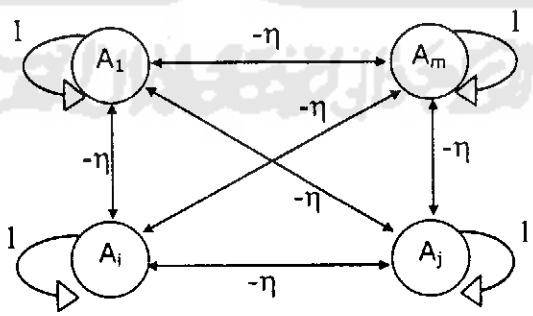
tersembunyi), seperti terlihat pada gambar 2.4 Umumnya, ada lapisan bobot-bobot yang terletak antara 2 lapisan yang bersebelahan. Setiap nilai yang di*input*kan akan dikalikan dengan bobot yang terhubung ke tiap *neuron* pada lapisan tersembunyi, lalu di jumlah. Hasil penjumlahannya di*input*kan pada fungsi aktivasi yang berlaku pada *neuron* lapisan tersembunyi tersebut untuk mendapatkan hasilnya. Kemudian, nilai hasil dari tiap *neuron* lapisan tersembunyi dikalikan dengan bobot yang terhubung ke masing-masing *neuron* pada sisi *output*. Hasil penjumlahannya dimasukan pada fungsi aktivasi yang berlaku untuk mendapatkan nilai keluarannya. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan dengan lapisan tunggal, tentu saja dengan pembelajaran yang lebih rumit. Namun demikian, pada banyak kasus, pembelajaran pada jaringan dengan banyak lapisan ini lebih sukses dalam menyelesaikan masalah.



Gambar 2.4 Jaringan syaraf dengan banyak lapisan

2.3.3 Jaringan dengan lapisan kompetitif (competitive layer net)

Umumnya hubungan antara neuron pada lapisan kompetitif ini tidak diperlihatkan pada diagram arsitektur. Gambar 2.5 menunjukan salah satu contoh arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot -η.



Gambar 2.5 Jaringan syaraf dengan lapisan kompetitif

2.4 Fungsi aktivasi

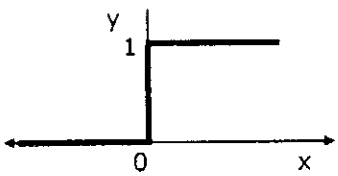
Fungsi aktivasi harus mempunyai beberapa karakterisrik penting, yaitu continue, terdefinisi, dan tidak monoton. Untuk fungsi aktivasi yang paling umum digunakan, nilai turunannya dapat dinyatakan dalam persamaan fungsi aktivasi tersebut. Tingkat aktivasi ditentukan dari perkalian setiap input dengan suatu bobot tertentu yang paling analog dengan synapsis atau sambungan dari suatu neuron target atau sesudahnya. Dalam penyelesaian setiap kasus yang menggunakan jaringan syaraf tiruan, ada beberapa fungsi aktivasi yang sering digunakan, antara lain:

2.4.1 Fungsi Undak Biner (Hard limit)

Jaringan dengan lapisan tunggal sering menggunakan fungsi undak (*step function*) untuk mengkonversikan input dari suatu variabel yang bernilai kontinue ke suatu output biner (0 atau 1). (Gambar 2.6)

Fungsi undak biner (*Hard limit*) dirumuskan sebagai:

$$y = \begin{cases} 0, & jika\ x \leq 0 \\ 1, & jika\ x > 0 \end{cases} \tag{2.1}$$



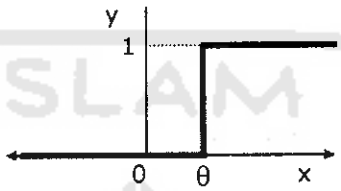
Gambar 2.6 Fungsi aktivasi: Undak Biner (*hard limit*)

2.4.2 Fungsi Undak Biner (*Threshold*)

Fungsi undak biner dengan menggunakan nilai amabang sering juga disebut dengan nama fungsi nilai ambang (*threshold*) atau fungsi Heavised. (Gambar 2.7)

Fungsi undak biner (dengan nilai amabang θ) dirumuskan sebagi:

$$y = \begin{cases} 0, & \text{jika } x < \theta \\ 1, & \text{jika } x \geq \theta \end{cases} \tag{2.2}$$



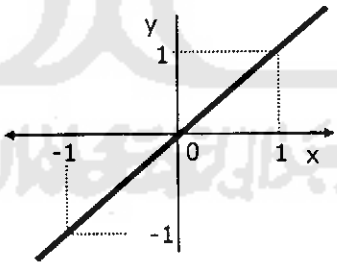
Gambar 2.7 Fungsi aktivasi: Under Biner (threshold)

2.4.3 Fungsi Linier (*indentitas*)

Fungsi linier memiliki nilai output yang sama dengan nilai inputnya. (Gambar 2.8)

Fungsi linier dirumuskan sebagai berikut:

$$y = x \tag{2.3}$$



Gambar 2.8 Fungsi aktivasi: Linier (identitas)

2.4.4 Fungsi Sigmoid Biner

Fungsi ini digunakan untuk jaringan syaraf yang dilatih dengan menggunakan metode backpropagation. Fungsi sigmoid biner memiliki nilai pada range 0 sampai 1. Oleh karena itu, fungsi ini sering digunakan untuk jaringan

syaraf yang membutuhkan nilai output yang terletak pada interval 0 sampai 1. Namun,fungsi ini bisa juga digunakan oleh jaringan syaraf yang nilai outputnya 0 atau 1,

Fungsi sigmoid biner dirumuskan sebagai:

$$y = f(x) = \frac{1}{1+\exp(-\sigma x)} \tag{2.4}$$

$$\text{dengan : } f'(x) = \sigma\, f(x)\,[1- f(x)] \tag{2.5}$$

2.4.5 Fungsi Sigmoid Bipolar

Fungsi sigmoid bipolar hampir sama dengan fungsi sigmoid biner, hanya saja output dari fungsi ini memiliki range antara 1 sampai -1.

Fungsi sigmoid bipolar dirumuskan sebagai:

$$g(x)= 2\, f(x) - 1 = \frac{2}{1+\exp(-\sigma x)} - 1 \tag{2.6}$$

$$= \frac{1-\exp(-\sigma x)}{1+\exp(-\sigma x)} \tag{2.7}$$

$$g'(x) = \frac{\sigma}{2}\,[1+g(x)][1-g(x)] \tag{2.8}$$

Fungsi ini sangat dekat dengan fungsi *hyperbolic tangent*. Keduanya memiliki range antara -1 sampai 1. Untuk fungsi *hyperbolic tangent*, dirumuskan sebagai berikut:

$$y = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.9}$$

atau $$y = f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.10}$$

dengan: $$f'(x) = [1 + f(x)][1 - f(x)] \tag{2.11}$$

2.5 Proses Pemberlajaran

Pada otak manusia, informasi yang dilewatkan dari satu neuron ke neuron yang lainnya berbentuk rangsangan listrik melalui dendrit. Jika rangsangan tersebut diterima oleh satu neuron, maka neuron tersebut akan membangkitkan output kesemua neuron yang berhubungan dengannya sampai informasi tersebut sampai ketujuannya yaitu terjadinya suatu reaksi. Selama proses pembelajaran, terjadi perubahan yang cukup berarti pada bobot-bobot yang menghubungkan antar neuron. Apabila ada rangsangan yang sama dengan rangsangan yang telah diterima oleh neuron, maka neuron akan memberikan reaksi dengan cepat.

Jaringan syaraf akan mencoba untuk mensimulasikan kemampuan otak manusia untuk belajar. Jaringan syaraf tiruan juga tersusun atas neuron-neuron dan dendrit. Tidak seperti model biologis, jaringan syaraf memiliki struktur yang tidak dapat diubah, dibangan oleh sejumlah neuron, dan memiliki nilai tertentu yang menunjukan seberapa besar koneksi antara neuron (yang dikenal dengan nama bobot). Perubahan yang terjadi selama proses pembelajaran adalah perubahan nilai bobot. Nilai bobot akan bertambah, jika informasi yang diberikan oleh neuron bersangkutan tersampaikan, sebaliknya jika informasi tidak disampaikan oleh satu neuron ke neuron yang lainya, maka nilai bobot yang menghubungkan ke duanya akan dikurangi. Pada saat pembelajaran dilakukan pada input yang berbeda, maka nilai bobot akan diubah secara dinamis hingga mencapai suatu nilai yang cukup seimbang.

Ada dua jenis pelatihan dalam sistem jaringan saraf tiruan, yaitu proses belajar terawasi (*supervised learning*) dan proses belajar tak terawasi *(unsupervised learning*).

2.5.1 Pembelajaran Terawasi (*Supervises Learning*)

Metode pembelajaran pada jaringan syaraf tiruan disebut terawasi jika output yang diharapkan telah diketahui sebelumnya. Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan *output*. Lapisan *output* ini akan membangkitkan pola *output* yang nantinya akan dicocokan dengan pola *output* targetnya. Apabila terjadi perbedaan antara pola *output* hasil pembelajaran dengan pola target, maka disini akan muncul error. Apabila nilai error ini masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi.

Dalam pembelajaran terawasi (*supervised learning*), proses pembelajaran dapat dikelompokan dalam beberapa metode, antara lain *Hebb Rule*, *Perceptron, Delta Rule*, *Backpropagation, Counterpropagation, Heteroassociative Memory, Bidirectional Associative Memory dan Learning Vector Quantization.*

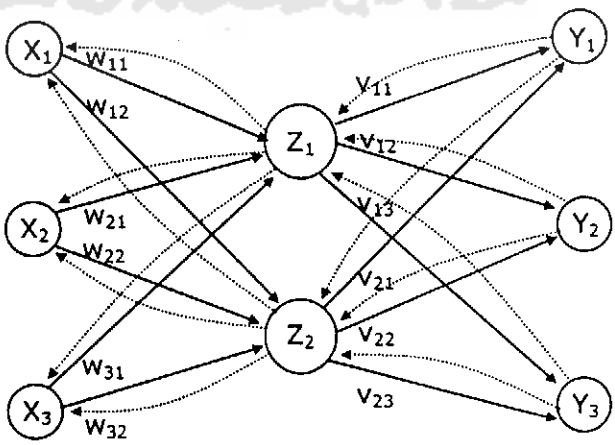
2.5.2 Pembelajaran Tak Terawasi (*Unsupervises Learning*)

Pada metode pembelajaran yang tak terawasi ini tidak memerlukan target *output*. Pada metode ini, tidak dapat ditentukan hasil yang 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 mengelompokan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk pengelompokan (klasifikasi) pola.

2.6 Metode Backpropagation

Backpropagation merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan *neuron-neuron* yang ada pada lapisan tersembunyinya. Algoritma *backpropagation* menggunakan error output untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan error ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, *neuron-neuron* diaktifkan dengan menggunakan fungsi aktivasi. Untuk metode backpropagatioan fungsi aktivasi yang sering digunakan diantaranya : *sigmoid biner, sigmoid bipolar, identitas*. Arsitektur jaringan *backpropagation* seperti terlihat pada Gambar (2.9) berikut ini :



Gambar 2.9 Arsitektur jaringan *backpropagation*.

2.6.1 Algoritma *Backpropagation*

Pada dasarnya, pelatihan dengan metode *Backpropagation* terdiri atas tiga langkah, yaitu sebagai berikut:

a. Data dimasukan ke *input* jaringan (*feedforward*)

b. Perhitungan dan propagasi balik dari *error* yang bersangkutan

c. Pembaharuan (*adjustment*) bobot dan bias

Saat umpan maju (*feedforward*), setiap unit input (X_i) akan menerima sinyal input dan akan menyebabkann sinyal tersebut pada tiap *hidden unit* (Z_j). Setiap *hiden unit* kemudian akan menghitung akativasinya dan menerima sinyal (Z_j) ke unit *output*. Kemudian, setiap unit *output* (Y_k) juga akan menghitung aktivasinya (Y_k) untuk menghasilkan respon terhadap *input* yang diberikan jaringan.

Saat proses pelatihan (*training*), setiap unit output membandingkan aktivasinya (y_k) dengan nilai target (*desired output*) untuk menentukan besarnya *error*. Berdasarkan *error* tersebut, dihitung faktor δ_k. Faktor δ_k digunakan untuk mendistribusikan *error* dari *output* kembali ke layer sebelumnya. Dengan cara yang sama, faktor δ_j juga dihitng pada *hidden* unit Z_j. Faktor δ_k digunakan untuk memperbaharui bobot antara *hidden layer* dan *input layer*.

Setelah semua faktor δ ditentukan, bobot untuk semua layer di adjust secara bersamaan. Pembaharuan bobot W_{jk} (dari *hidden unit* Z_j ke unit *output* Y_k) dilakukan berdasarkan faktor δ_k dan aktivasi z_j dari *hiden unit* Z_j. Sedangkan pembaharuan bobot v_{ij} (dari *input unit* X_i ke *hidden unit* Z_j) dilakukan berdasarkan faktor δ_j dan aktivasi x_i dari *input*.

Selengkapnya, notasi-notasi yang akan digunakan pada algoritma pelatihan yaitu sebagai berikut.

x Data training untuk input

$x = (x_1, \ldots, x_i, \ldots x_n)$

t Data training untuk output (target/desired output)

$t = (t_1, \ldots, t_k, \ldots t_m)$

α Learning rate yaitu parameter yang mengontrol perubahan bobot selama pelatihan. Jika *learning rate* besar, jaringan semakin cepat belajar, tetapi hasilnya kurang akurat. *Learning rate* biasanya dipilih atara 0 dan 1.

X_i Unit input ke-i. Untuk *unit input*, sinyal yang masuk dan keluar pada suatu unit dilambangkan dengan variabel yang sama, yaitu x_i.

Z_j *Hidden unit* ke-j. Sinyal input pada Z_j dilambangkan dengan z_{inj}. Sinyal *output* (aktivasi) untuk Z_j dilambangkan dengan z_j.

v_{0j} Bias untuk hidden unit ke-j.

v_{ij} Bobot antara unit *input* ke-i dan *hidden* unit ke-j.

Y_k Unit output ke-k. Sinyal input ke Y_k dilambangkan dengan y_{ink}. Sinyal *output* (aktivasi) untuk Y_k dilambangkan dengan y_k.

w_{0k} Bias untuk unit *output* ke-k.

w_{jk} Bobot antara hidden unit ke-j dan unit output ke-k.

δ_k Faktor koreksi *error* untuk bobot w_{jk}.

δ_j Faktor koreksi *error* untuk bobot v_{ij}.

Secara detail, langkah-langkah pelatihan *Backpropagation* sebagai berikut;

1. Langkah 1: Inisialisasi bobot dan bias. Baik bobot maupun bias dapat diset dengan sembarang angka (acak) dan biasanya, angka di sekitar 0 dan 1 atau -1 (bias positif atau negatif).
2. Langkah 2: Jika stop condition masih belum terpenuhi, jalankan langkah 2-10.
3. Langkah 3: Untuk setiap *data training*, lakukan langkah 4-9

Umpan maju (feedforward)

4. Langkah 4: Tiap-tiap unit input (X_i, i = 1,2,3,…,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada diatasnya (lapisan tersembunyi).
5. Langkah 5: Tiap-tiap hidden unit (Z_j, j=1,.....,p) akan menjumlahkan sinyal-sinyal input yang sudah berbobot, termasuk biasnya.

$$z_in_j = v_{0j} + \sum_{j=1}^{n} x_i v_{ij} \tag{2.12}$$

Dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal output dari hidden unit yang bersangkutan.

$$z_j = f(z_in_j) \tag{2.13}$$

6. Langkah 6: Setiap unit output (Yk, k=1,...,m) akan menjumlahkan sinyal-sinyal input yang sudah berbobot, termasuk biasnya.

$$y_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \tag{2.14}$$

Dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal output dari unit output yang bersangkutan.

$$y_k = f(y_in_k) \tag{2.15}$$

Lalu mengirim sinyal output ini ke seluruh unit pada unit output.

Propagasi error (*backpropagation of error*)

7. Langkah 7: Setiap output (Yk, k=1,...,m) menerima suatu target *pattern* (*desired output*) yang sesuai denganinput *training pattern* untuk meghitung kesalahan (*error*) antara target dengan output yang dihasilkan jaringan.

$$\delta_k = (t_k - y_k)\, f'(y_in_k) \tag{2.16}$$

kemudian hitung koreksi error (yang nantinya akan digunakan untuk memperoleh nilai w_{jk}) :

$$\Delta w_{jk} = \alpha\, \delta_k\, z_j \tag{2.17}$$

hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k}) :

$$\Delta w_{0k} = \alpha\, \delta_k \tag{2.18}$$

kirimkan ini ke unit-unit yang berada lapisan dibawahnya.

8. Langkah 8: Tiap-tiap unit tersembunyi (Z_j, j = 1,2,3,…,p) menjumlahkan delta inputnya (dari unit-unit yang berada pada lapisan diatasnya) :

$$\delta_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \tag{2.19}$$

Kalikan nilai ini dengan turunan dari fungsi aktivasinya untuk menghitung informasi error :

$$\delta_j = \delta_in_j\, f'(z_in_j) \tag{2.20}$$

Kemudian hitung koreksi error (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij} :

$$\Delta v_{ij} = \alpha\ \delta_j\ x_i \qquad (2.21)$$

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j} :

$$\Delta v_{0j} = \alpha\ \delta_j \qquad (2.22)$$

Perbaharui (*adjusment*) bobot dan bias :

9. Langkah 9: Tiap-tiap unit output (Y_k, k = 1,2,3,...m) memperbaiki bias dan bobotnya (j= 0,1,2,...,p) :

$$w_{jk}(\text{baru}) = w_{jk}(\text{lama}) + \Delta w_{jk} \qquad \text{(bobot)} \qquad (2.23)$$

$$w_{0k}(\text{baru}) = w_{0k}(\text{lama}) + \Delta w_{0k} \qquad \text{(bias)} \qquad (2.24)$$

Tiap-tiap unit tersembunyi (z_j, j = 1,2,3,...,p) memperbaiki bias dan bobotnya (i = 0,1,2,...,n) :

$$v_{ij}(\text{baru}) = v_{ij}(\text{lama}) + \Delta v_{ij} \qquad \text{(bobot)} \qquad (2.25)$$

$$v_{0j}(\text{baru}) = v_{0j}(\text{lama}) + \Delta v_{0j} \qquad \text{(bias)} \qquad (2.26)$$

10. Langkah 10: Tes kondisi berhenti (*stop condition*).

2.6.2 Pengujian Data

Setelah proses pembelajaran dilakukan berdasarkan langkah-langkah algoritma *backpropagation*, maka diperoleh nilai bobot akhir dan nilai bias akhir yaitu dari iterasi terakhir dengan nilai error atau galat 0,01. Nilai bobot dan bias inilah yang akan digunakan dalam melakukan pengujian terhadap data dengan

perambatan maju (*forward propagation*). Adapun langkah-langkah pengujian data adalah sebagai berikut :

Umpan maju (*Feedforward*) :

1. Langkah 1 : Tiap-tiap unit input (X_i, i = 1,2,3,...,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada diatasnya (lapisan tersembunyi).
2. Langkah 2 : Tiap-tiap unit tersembunyi (Z_j, j = 1,2,3,...,p) menjumlahkan sinyal-sinyal input berbobot :

$$z_in_j = v_{0j} + \sum_{j=1}^{n} x_i v_{ij} \tag{2.27}$$

gunakan fungsi aktivasi untuk menghitung sinyal outputnya :

$$z_j = f(z_in_j) \tag{2.28}$$

dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit output).

3. Langkah 3 : Tiap-tiap unit output (Y_k, k = 1,2,3,...,m) menjumlahkan sinyal-sinyal input berbobot.

$$y_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \tag{2.29}$$

gunakan fungsi aktivasi untuk menghitung sinyal outputnya :

$$y_k = f(y_in_k) \tag{2.30}$$

dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit output).

2.6.3 Backpropagation Sederhana

Jika tes kondisi telah terpenuhi, pelatihan jaringan dapat dihentikan. Untuk menentukan kondisi berhenti terdapat dua cara yang biasa di pakai yaitu sebagai berikut.

1. Dengan membatasi iterasi yang diingin dilakukan. Misalnya, jaringan akan dilatih sampai dengan iterasi ke-500, yang dimaksud satu iterasi adalah perulangan langkah ke-3 sampai langkah ke-8 untuk semua training data yang ada.
2. Dengan membatasi error. Pada metode *backpropagation*, dipakai metode *Mean Square Error* untuk menghitung rata-rata error antara output yang dikehendaki, pada training data dengan output yang dihasilkan oleh jaringan. Misalnya, error telah mencapai 0,01 (1%), pelatihan dihentikan. Besar persen error tersebut tergantung kepresisian yang dibutuhkan oleh sistem yang bersangkutan.

Selain kedua cara di atas, ada sebuah pertimbangan lagi yang dapat dipakai untuk menghentikan pelatihan. Ada kalanya, sebelum mencapai kondisi seperti yang diinginkan error justru semakin besar. Kejadian seperti itu disebut *over training*. Kondisi error yang diperhatikan disini tidak hanya error dari *training set*, tetapi juga dari *test set*. Jika salah satu dari *training set error* atau *test set error* bertambah besar, pelatihan harus dihentikan.

Cara memeriksa *stopping condition* dengan *Mean Square Error* adalah sebagai berikut:

a. Dengan bobot yang ada saat itu, lakukan set umpan maju (langkah ke-4 sampai dengan langkah ke-6) dimana input nya diambil dari input training set jika yang ingin dihitung adalah training set error dan atau input test set jika yang ingin dihitung adalah test set error. Langkah tersebut dilakukan untuk semua data training / test yang ada.

b. Kemudian, dicari selisih antara target output (t_k) dengan output jaringan (y_k) dan diimplementasikan pada persamaan *Mean Square Error*. Jika terdapat m training data, maka:

Mean Square Error $=E=0.5*\{(t_{k1}-y_{k1})^2+(t_{k2}-y_{k2})^2+ \dots +(t_{km}-y_{km})^2\}$ (2.31)

Setelah pelatihan selesai, jika kemudian jaringan diberi input, jaringan akan dapat menghasilkan output seperti yang diharapkan. Cara mendapatkan output adalah dengan mengimplementasikan metode *Backpropagation*, tetapi hanya pada bagian umpan majunya.

Sebagian besar algoritma pelatihan untuk jaringan feedforward menggunakan gradien dari fungsi kinerja untuk menentukan bagaimana mengatur bobot-bobot dalam rangka meminimumkan kinerja. Gradien ini ditentukan dengan menggunakan suatu teknik yang disebut dengan nama *backpropagation*. Pada dasarnya, algoritma pelatihan *backpropagation* akan menggerakan bobot dengan arah gradien negatif.

Prinsip dasar dari algoritma *backpropagation* sederhana adalah memperbaiki bobot-bobot jaringan dengan arah yang membuat fungsi kinerja menjadi turun dengan cepat. Ada 2 cara untuk mengimplementasikan algoritma gradient descent, yaitu:

2.6.3.1 *Data Updating*

Pada data updating, penghitungan gradien dan perbaikan nilai bobot-bobot dilakukan pada setiap pengoperasian input data. Artinya, setiap pasangan input/output selesai dilakukan pelatihan maka bobot-bobot dan bias akan di adjusment. Sehingga jika suatu jaringan dilakukan pelatihan sebanyak k iterasi, maka jaringan akan melakukan perbaikan bobot-bobot dan bias sebanyak k kali m (jumlah training pattern).

2.6.3.1.1 *Gradient Descent Method*

Model pelatihan ini menggunakan algoritma dasar gradient descent pada data updating, dimana bobot-bobot akan diperbaiki, dengan perubahan seperti yang ditunjukan pada persamaan (2.17), (2.18), (2.21) dan (2.22). Parameter yang berhubungan dengan gradient descent method, yaitu :

1. Learning rate, semakin besar nilai learning rate akan berimplikasi pada semakin besarnya langkah pembelajaran. Jika learning rate diset terlalu besar, maka algoritma akan menjadi tidak stabil. Sebaliknya, jika learning rate diset terlalu kecil, maka algoritma akan konvergen dalam jangka waktu yang sangat lama (default 0.5).
2. Maksimum iterasi, jumlah iterasi maksimum yang boleh dilakukan selama proses pelatihan. pelatihan akan dihentikan apabila nilai iterasi melebihi maksimum iterasi.
3. Kinerja tujuan (target error), pelatihan akan dihentikan apabila nilai fungsi kinerja kurang dari atau sama dengan kinerja tujuan (default 0.01).

2.6.3.1.2 *Gradient Descent Method with Momentum*

Pada model pelatihan ini tidak hanya merespon gradien lokal saja, namun juga mempertimbangkan kecenderungan yang baru saja terjadi pada permukaan error. Besarnya perubahan bobot ini dipengaruhi oleh suatu konstanta (yang dikenal dengan nama momentum; notasi : mc), yang bernilai antara 0 sampai 1.

Model pelatihan *garadient descent method with momentum*, perubahan bobot pada persamaan (2.17), berubah menjadi:

$$\Delta w_{jk} = \alpha\ \delta_k\ z_j \tag{2.32}$$

untuk iterasi = 1, dan

$$\Delta w_{jk} = mc * \Delta w_{jk}\ (\textit{iterasi sebelumnya}) + (1\text{-}mc) * \alpha\ \delta_k\ z_j \tag{2.33}$$

untuk iterasi > 1.

Demikian pula untuk bobot bias, persamaan (2.18) berubah menjadi:

$$\Delta w_{0k} = \alpha\ \delta_k \tag{2.34}$$

untuk iterasi = 1, dan

$$\Delta w_{0k} = mc * \Delta w_{0k}\ (\textit{iterasi sebelumnya}) + (1\text{-}mc) * \alpha\ \delta_k \tag{2.35}$$

untuk iterasi > 1.

Perubahan bobot pada persamaan (2.21), berubah menjadi:

$$\Delta v_{ij} = \alpha\ \delta_j\ x_i \tag{2.36}$$

untuk iterasi = 1, dan

$$\Delta v_{ij} = mc * \Delta v_{ij}\ (\textit{iterasi sebelumnya}) + (1\text{-}mc) * \alpha\ \delta_j\ x_i \tag{2.37}$$

untuk iterasi > 1, dan

Demikian pula untuk bobot bias, persamaan (2.11) berubah menjadi:

$$\Delta v_{0j} = \alpha\ \delta_j \tag{2.38}$$

untuk iterasi = 1, dan

$$\Delta v_{0j} = mc^* \ \Delta v_{0j}\ (\textit{iterasi sebelumnya}) + (1\text{-}mc)^*\alpha\ \delta_j \tag{2.39}$$

untuk iterasi > 1.

Dengan demikian, apabila nilai mc = 0, maka perubahan bobot hanya akan dipengaruhi oleh gradiennya. Namun apabila mc = 1, maka perubahan bobot akan sama dengan perubahan bobot sebelumnya.

Parameter yang digunakan pada model pelatihan ini sama dengan model pelatihan *gradient descent method* ditambah parameter momentum (default 0.5).

2.6.4 Normalisasi dan Denormalisasi

Sebelum dilakukan pelatihan, seringkali dilakukan penskalaan pada input dan target sedemikian hingga data-data input dan target tersebut masuk dalam suatu range tertentu. Ada beberapa cara untuk menormalisasikan data input dan target diantaranya yaitu:

2.6.4.1 Mean dan Deviasi standar

Selain dengan meletakan data-data input dan target pada range tertentu, proses normalisasi juga dapat dilakukan dengan bantuan mean dan deviasi standar yang akan membawa data ke bentuk normal dengan mean=0 dan deviasi standar=1.

Mean : $$m = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.40}$$

Deviasi standar : $$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - m)^2} \tag{2.41}$$

$$x_{ij} = \frac{x_{ij} - m}{\sigma} \text{ (normalisasi)} \tag{2.42}$$

$$x_{ij} = (\sigma * x_{ij}) + m \text{ (denormalisasi)} \tag{2.43}$$

2.6.5 Analisis Hasil Pelatihan

Kinerja dari suatu jaringan saraf setelah dilakukan pelatihan dapat diukur dengan melihat error hasil pelatihan, validasi, dan testing terhadap sekumpulan data input baru. Salah satu cara yang dapat digunakan untuk evaluasi ini adalah dengan menggunakan analisis regresi terhadapt output jaringan dan target.

$$A = f_1 T + f_0 \tag{2.44}$$

$$f_1 = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{m\sum x_i^2 - \left(\sum x_i\right)^2} \tag{2.45}$$

$$f_0 = \frac{m\sum x_i y_i - \sum x_i \sum y_i}{m\sum x_i^2 - \left(\sum x_i\right)^2} \tag{2.46}$$

$$r = \frac{\left(\sum x_i y_i - m\overline{x}\overline{y}\right)}{\sqrt{\left(\sum x^2 - m\overline{x}^2\right)\left(\sum y^2 - m\overline{y}^2\right)}} \tag{2.47}$$

A : actual output

T : target

f_0 : titik potong dengan sumbu y.

f_1 : gradien garis hasil regresi linear

r : koefisien korelasi

2.7 Prediksi Valuta Asing

Dalam bisnis internasional memprediksi suatu kurs mata uang merupakan hal yang sangat penting dilakukan karena jika hal tersebut diabaikan pengusaha akan mengalami kerugian. Banyaknya pendapat yang mengatakan bahwa tidak ada teknik dan metodelogi yang sempurna untuk meramal valas. Hal ini dimaklumi karena tidak ada suatu penjelasanyang tepat mengenai bagaimana kurs valas bergerak. Pergerakan kurs valas tergantung dari interaksi berbagai faktor secara stimultan. Bagaimana faktor tersebut mempengaruhi pergerakan valas relatif sulit dikuantifikasikan maupun diramal. Kurs valas dapat saja bereaksi sangat tajam akibat suatu peristiwa yang tak terduga sebelumnya sehingga mengacaukan berbagai teori dan ramalan pada periode tersebut. Oleh karena itu diharapkan mampu membuat suatu aplikasi yang dapat membatu hal tersebut.