BAB 2

LANDASAN TEORI

2.1 Format Citra

Citra adalah representasi dari suatu obyek nyata baik dalam bentuk dua dimensi atau tiga dimensi menjadi bentuk gambar digital yang dapat dimengerti oleh komputer [Jain, 1989, p1-2]. Secara umum format citra dikelompokkan menjadi dua, yaitu *raster format* dan *vector format*, dimana terdapat perbedaan yang mendasar di antara kedua format tersebut yang tergantung pada penyimpanan citra.

Raster format menyimpan data citra digital secara keseluruhan. *Raster format* atau *bitmap* berisi kumpulan *pixel* dari seluruh data citra. *Pixel* adalah nilai dari intensitas cahaya pada satu titik nyata di atas *scan line* [Turban, 1992, p345]. Format file *bitmap* biasa digunakan untuk merepresentasikan benda – benda nyata yang ditangkap dengan peralatan kamera video atau *scanner*. *Raster format* merupakan tipe citra yang paling sering digunakan karena memiliki keuntungan seperti menghasilkan warna fotografik, sifat dan textur yang paling akurat. Beberapa tipe file untuk menyimpan suatu citra *bitmap* adalah *Bitmapped File* (*.BMP), *Graphics Image Format* (*.GIF), *Image* (*.IMG), *PC Paintbrush Format* (*.PCX), *Tagged Image File Format* (*.TIFF), *Joint Photographic Expert Group* (*.JPEG), dan sebagainya.

Vector format menyimpan elem3.en – elemen yang ada pada suatu citra. *Vector format* berisi bentuk – bentuk atau segmen – segmen garis. Biasanya *vector format* digunakan pada aplikasi – aplikasi yang mementingkan ketelitian dan hubungan antar elemen dalam citra. Citra vektor dapat dikonversi menjadi *raster format* untuk ditampilkan atau dicetak, sedangkan konversi dari *raster format* ke *vector format* lebih sulit untuk dilakukan. [Lindley, 1991, p185 – 186]. *Vector format* menggambarkan suatu citra dengan garis, kurva, titik, panjang, dan lain-lain. Dimana keuntungan dari *vector format* adalah ukuran file yang relatif lebih kecil daripada *bitmap*, tetapi sayangnya *vector format* kurang baik untuk menghasilkan gambar fotografik. Biasanya *vector format* dikeluarkan ke sebuah *plotter* (alat yang bertipe vektor). Beberapa tipe file untuk menyimpan suatu *vector format* adalah *Hewlett-Packard Graphics Language* (*.HPGL), *Initial Graphics Exchange Standard* (*.IGES), *Computer Graphics Metafile* (*.CGM), dan sebagainya.

Setiap tipe format di atas memiliki keuntungan dan kerugian, tidak bisa dikatakan format mana yang paling baik karena semuanya tergantung dari kebutuhan, misalnya apabila dilihat dari besar ukuran, *Graphics Image Format* (*.GIF) memiliki ukuran yang terkecil, tetapi apabila dilihat dari variasi warna, *.GIF memiliki kekurangan yaitu banyak hilangnya variasi warna.

2.1.1. Gray Scale Modification

Gray scale modification merupakan teknik untuk menjadi terang atau gelap pada layar. Perkiraan tingkat keterangan dari sebuah *pixel* diwakili oleh 8-bit

nomor biner dimana 0 (00000000) sebagai hitam dan 255 (11111111) sebagai putih. Jika pada layar sangat gelap, semua *pixel* akan mempunyai nilai yang rendah. Salah satu cara mengolah citra menjadi terang yaitu dengan menambahkan nilai tertentu pada setiap *pixel*. Hubungan antara *pixel* yang berdekatan tidak berubah, tetapi hanya mempengaruhi seluruh layar menjadi terang. Kemajuan kontras merupakan salah satu tipe dari *gray scale modification*. Kontras adalah jarak antara titik paling gelap dan paling terang di sebuah citra. [Turban, 1992, p350 – 351].

2.1.2. Format file PCX

Format file PCX merupakan *raster format* (*bitmap*) yang paling lama, pertama kali dikembangkan pada awal tahun 1980-an oleh *Zsoft Corporation* untuk mendukung program *PC Paintbrush*, dimana setelah itu penggunaanya menyebar ke program – program berbasis *bitmap* (*raster format*) lainnya.

Format file PCX secara luas mendukung aplikasi grafik, dimana telah direvisi tahun – tahun terakhir untuk menyelesaikan langkah dari timbulnya teknologi tampilan grafik. Ini semula hanya mendukung *monochrome* dan *16-color-palette-based imagery*. Sesudah IBM mengeluarkan *PS / 2 line*, penyesuaian VGA dengan cepat menjadi tampilan standar yang baru, dan format file PCX merevisi untuk mendukung *256-color imagery*. Kemudian format file PCX meningkatkan lagi dengan menambahkan *24-bit color imagery*. Khusus nama – nama file PCX digunakan satu dari dua spesifik ekstensi antara lain .PCX atau .PCC, dimana keduanya menunjukkan format file PCX. [Luse, 1993, p301-302].

Format file PCX mendukung 1, 4, 8, atau 24 bit yang dapat disimpan dari RGB color, Indexed Color, Grayscale, dan Bitmap display modes. Karena sudah berkembang sangat lama, saat ini banyak terdapat versi PCX, yang biasanya disupport oleh banyak software adalah PCX versi 5. Sedangkan PCX versi 3 hanya mendukung warna 256 bit, apabila kita membuka suatu file PCX versi 3, VGA pallete yang standart harus digunakan.

Format PCX memiliki beberapa keuntungan yaitu :

1. Menyimpan seluruh informasi suatu citra, hampir sama dengan format TIFF .
2. Memiliki kualitas citra yang sangat baik.

Format file PCX terdiri dari tiga bagian, yaitu : *header* yang menempati 128 byte pertama, data *bitmap* dan palet warna. *Header* file PCX dapat dilihat pada Tabel 2.1. *Header* file PCX tersebut berisi informasi yang diperlukan untuk dapat menerjemahkan dengan benar data file yang mengikutinya.

Byte#	Ukuran (Byte)	Nama	Keterangan
1	1	Zsoft flag	Berisi nilai 10 (0Ah) = Zsoft PCX file
2	1	Versi	0 = PC Paintbrush 2.5 2 = PC Paintbrush 2.8 dengan palet 3 = PC Paintbrush 2.8 tanpa palet 4 = PC Paintbrush for Windows 5 = PC Paintbrush 3.0 ke atas
3	1	Encoding	1 = PCX run-length encoding
4	1	Bitpx	Jumlah bit per pixel dalam sebuah citra
5	2	Xmin	Lokasi sisi kiri citra

Byte#	Ukuran (Byte)	Nama	Keterangan
7	2	Ymin	Lokasi sisi atas citra
9	2	Xmax	Lokasi sisi kanan citra
11	2	Ymax	Lokasi sisi bawah citra
13	2	Hres	Resolusi horisontal citra (dots per inch)
15	2	Vres	Resolusi vertikal citra (dots per inch)
17	4	ColorMap	Berisi palet warna
65	1	Reserve	Tidak dipakai
66	1	NPlane	Jumlah bit plane dalam sebuah citra
67	2	Bplin	Jumlah byte per baris per plane
69	2	PalType	Nilai byte dalam menentukan jenis palet. 1 = warna 2 = gray scale
71	2	HscreenSize	Resolusi horisontal layar
73	2	VscreenSize	Resolusi vertikal layar
75	5	Filter	Kosong

Tabel 2.1 *Header* File PCX

2.2 Principal Component Analysis

Principal Component Analysis (PCA) merupakan sebuah metode transformasi koordinat yang diasosiasikan dengan *multiband imagery* [Corner, 1997]. PCA adalah metode statistik yang terbaik [Hollmen, 1996]. Transform garis ini telah digunakan secara luas di dalam analisis data dan compression. Komponen analisis terpenting didasarkan atas penjelasan statistik dari variabel acak. PCA meliputi prosedur matematika yang mentransformasi sejumlah variabel yang berhubungan

menjadi sejumlah variabel yang tidak berhubungan disebut sebagai *principal components* [Boersma dan Weenink, 1999]. PCA dapat mengurangi sebuah redudansi yang dimuat oleh suatu data dengan cara membentuk suatu seri baru dari sebuah citra (komponen) yaitu dengan menentukan sumbu dari system koordinat yang baru dengan mengurangi perbedaan yang ada. Komponen baru yang dihasilkan lebih *interpretable* dari pada citra yang sebelumnya.

Saat merepresentasikan citra, PCA hanya menggunakan intensitas *pixel* citra dan tidak menggunakan ciri - ciri atau bentuk - bentuk tertentu dari sebuah citra. Jadi PCA ini merupakan teknik yang bersifat *shape-free*. Dalam pengenalan pola, teknik PCA ini digunakan baik secara langsung maupun tidak secara langsung. *Principal component* secara langsung digunakan sebagai basis proyeksi, sedangkan PCA secara tidak langsung digunakan untuk mereduksi dimensi [Liu dan Wechsler, 1998, p3-4].

Perhitungan teknik PCA dapat didefinisikan sebagai berikut :

Misalkan terdapat sebuah himpunan citra – citra $\Gamma_1, \Gamma_2, ..., \Gamma_P$, maka rata – rata dari citra – citra tersebut dapat dihitung dengan :

$$\Psi = \frac{1}{P} \sum_{i=1}^{P} \Gamma_i \qquad (2.1)$$

Kemudian didapat himpunan baru $\Phi_1, \Phi_2, ..., \Phi_P$ yang didapatkan dari $\Phi_i = \Gamma_i - \Psi$. *Principal component* dari himpunan citra – citra tersebut merupakan vektor *Eigen* dari *covariance matrix* :

$$C = \frac{1}{P-1}\sum_{i=1}^{P}\Phi_i\Phi_i^T = AA^T \qquad (2.2)$$

Dimana $A = [\Phi_1\Phi_2...\Phi_P]$ dan C adalah matriks N x N (N adalah dimensi dari citra). Karena dimensi dari citra tinggi, maka untuk menghitung N vektor *Eigen* dari C adalah sulit. Misalkan V_i adalah vektor *Eigen* dari $A^T A$ (berdimensi P x P), maka dilakukan langkah berikut :

$$(A^T A)V_i = \lambda_i V_i$$

dimana λ_i adalah nilai – nilai *Eigen*. Kemudian masing – masing ruas dikalikan dengan A :

$$A(A^T A)V_i = A(\lambda_i V_i)$$

maka didapatkan :

$$(AA^T)(AV_i) = \lambda_i(AV_i)$$

Ini berarti AV_i adalah vektor *Eigen* dari $C = AA^T$, maka vektor *Eigen* dari C :

$$U_i = AV_i = [A_1, A_2, ..., A_P]\begin{bmatrix} v_1^i \\ v_2^i \\ \text{M} \\ v_P^i \end{bmatrix} = \sum_{k=1}^{P} v_k^i A_k \qquad (2.3)$$

Dimana i = 1, 2, …, P-1 dan v_k^i adalah elemen ke-k dari V_i.

2.3 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) atau *Artificial Neural Network* (ANN) adalah sebuah sistem pengolah informasi yang memiliki beberapa karakteristik yang pada

umumnya hampir sama dengan jaringan saraf manusia [Fausett, 1994, p3]. Pengertian lain dari Jaringan Saraf Tiruan (JST) adalah suatu kelompok elemen pengolah dimana biasanya satu sub kelompok memiliki komputasi tersendiri yang hasilnya menjadi masukkan bagi sub kelompok lainnya [Rao, 1993, p2].

Jaringan Saraf Tiruan sudah dikembangkan sebagai suatu model matematika dari *human cognition* atau *neural biology*, berdasarkan asumsi sebagai berikut :

1. Pengolahan informasi terjadi pada beberapa elemen sederhana, yang disebut dengan *neurons*.
2. Neuron - neuron tersebut dilewati oleh sinyal – sinyal melalui *connection link*.
3. Setiap *connection link* memiliki *associated weight*, dimana biasanya memperbanyak sinyal yang ditransmisi.
4. Setiap neuron memakai fungsi aktivasi ke jaringan input untuk menentukan sinyal *output*nya.

Jaringan Saraf Tiruan digambarkan dengan pola dari koneksinya antara *neurons* (disebut dengan *architecture*), metode atas penentuan *weights* pada koneksi (disebut dengan *training* atau *learning*, *algorithm*) serta fungsi aktivasinya.

Jaringan Saraf Tiruan terdiri dari elemen - elemen pengolah sederhana dalam jumlah besar, yang disebut dengan *neurons*, *units*, *cells*, atau *nodes*. Tiap *nodes* dihubungkan dengan *nodes* lain dengan menggunakan sebuah *associates weight*, dimana *weight* tersebut menggambarkan informasi yang digunakan jaringan untuk memecahkan suatu masalah.

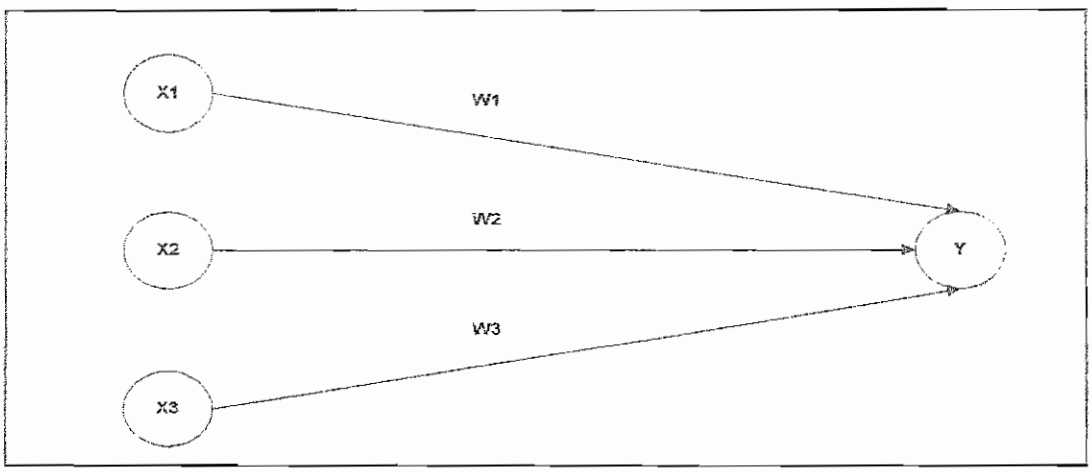
Setiap *neuron* memiliki *internal state* yang disebut *activation* atau *activity level*. Sebuah *neuron* mengirimkan aktivasinya berupa suatu sinyal ke beberapa *neurons* lain. Adalah sangat penting untuk diperhatikan bahwa suatu neuron hanya dapat mengirimkan satu sinyal pada suatu saat tertentu, walaupun sinyal tersebut disebar ke beberapa *neuron* lain.

Misalkan terdapat suatu *neuron* adalah Y, yang menerima input dari *neurons* X_1, X_2 dan X_3 (Gambar 2.1). Aktivasi (sinyal output) dari *neurons* tersebut adalah x_1, x_2, x_3. *Weights* pada koneksi dari X_1, X_2, X_3 ke *neuron* Y adalah w_1, w_2, dan w_3. Jaringan input, *y_in* ke *neuron* Y adalah penjumlahan dari sinyal-sinyal *weight* X_1, X_2 dan X_3, yang dijabarkan dalam persamaan berikut :

$$y\_in = w_1x_1 + w_2x_2 + w_3x_3$$

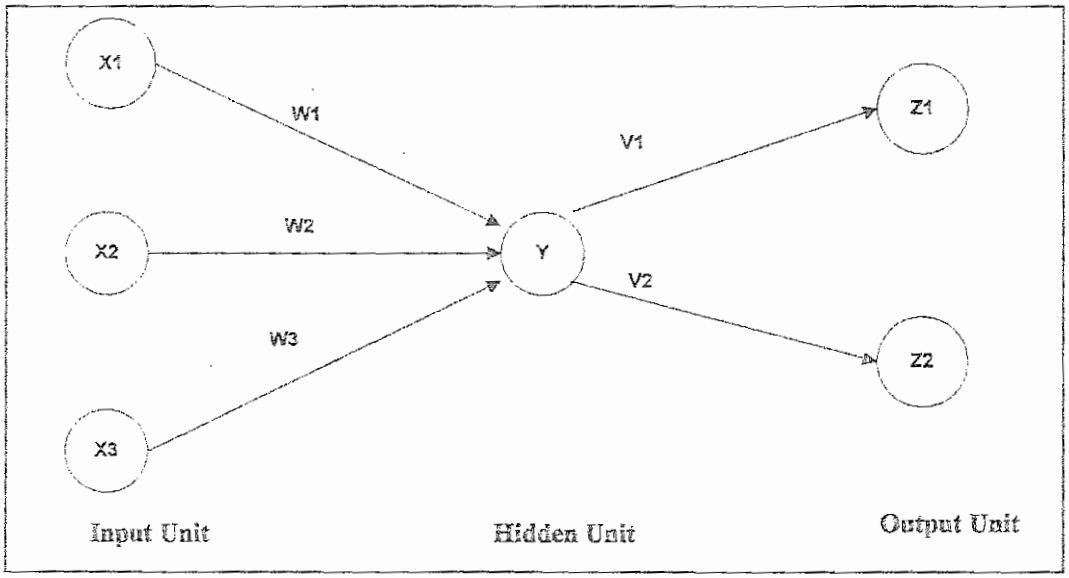
Aktivasi y dari *neuron* Y diberikan oleh fungsi dari jaringan input, $y = f(y\_in)$, sebagai contoh fungsi dari *logistic sigmoid*, sebagai berikut :

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



Gambar 2.1 Jaringan Saraf Tiruan Sederhana

Apabila *neuron* Y dihubungkan dengan Z_1 dan Z_2 yang memiliki *weights* v_1 dan v_2, *neuron* Y mengirimkan sinyal y ke unit-unit tersebut. Akan tetapi pada umumnya nilai yang diterima oleh *neuron* Z_1 dan Z_2 akan berbeda, karena setiap sinyal diukur melalui *weight* yang sesuai, v_1 atau v_2. Biasanya aktivasi z_1 dan z_2 dari *neuron* Z_1 dan Z_2 tergantung dari beberapa input atau banyak *neuron*, tidak hanya satu (Gambar 2.2)



Gambar 2.2 Jaringan Saraf Tiruan Sangat Sederhana

Walaupun jaringan saraf pada gambar 2.2 sangat sederhana, keberadaan dari *hidden unit* bersamaan dengan fungsi aktivasi bukan bersifat linier memberikan kemampuan untuk memecahkan suatu masalah yang dapat diselesaikan dengan suatu jaringan yang hanya memiliki unit input dan output, tetapi sebaliknya jaringan yang memiliki *hidden unit* lebih sulit untuk dilatih. [Fausett, 1994, p3-5].

2.3.1. Sejarah Singkat Perkembangan Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) pertama kali dikembangkan oleh Warren McCulloch dan Walter Pitts pada tahun 1943. Dalam penelitian ini ditemukan bahwa penggabungan *neurons* sederhana menjadi suatu sistem saraf merupakan suatu sumber peningkatan kekuatan komputasi, dimana *neuron* McCulloch-Pitts ini dapat diatur menjadi suatu jaringan untuk menghasilkan suatu keluaran yang dapat direpresentasikan menjadi suatu kombinasi dari fungsi – fungsi logik. Kemudian penelitian dilanjutkan oleh Donald Hebb pada tahun 1949, dasar pemikirannya adalah jika dua *neuron* diaktifkan secara serentak maka kekuatan dari koneksi antara dua *neuron* tersebut dapat ditingkatkan.

Pada tahun 1950-an dan 1960-an, ditemukan beberapa bentuk JST, antara lain : *Perceptrons* (ditemukan oleh Block, Minsky & Papert, dan Frank Rosenblatt), *Adaline* (ditemukan oleh Bernard Widrow dan beberapa muridnya).

Pada sekitar tahun 1980, ditemukannya *Backpropagation*, yang merupakan suatu metode propagasi suatu informasi error pada output, balik kembali ke *hidden unit*. Kemudian John Hopfield mengembangkan sejumlah jaringan saraf berdasarkan *weights* yang tetap dan aktivasi adaptif. Setelah itu Kunihiko Fukushima dengan beberapa koleganya mengembangkan jaringan saraf untuk mengenal karakter yang dinamakan dengan *neorecognition*.

2.3.2 Aplikasi Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) banyak sekali digunakan dalam berbagai bidang khususnya teknologi, baik pada tahap pengembangan maupun tahap penerapannya. JST dapat digunakan untuk menyelesaikan permasalahan - permasalahan yang memiliki fungsi pemetaan yang tidak jelas atau permasalahan yang membutuhkan kemampuan untuk mengenali data masukan disertai dengan *noise* (gangguan). Beberapa contoh aplikasi kegunaan JST adalah sebagai berikut :

2.3.2.1. Pengenalan Pola (*Pattern Recognition*)

Salah satu area yang dikembangkan dengan menggunakan JST adalah pengenalan dari tulisan tangan (angka maupun huruf).

Jaringan saraf multilayer seperti jaringan *Backpropagation* (jaringan multilayer yang dilatih secara propagasi balik), telah digunakan untuk mengenali kode pos yang ditulis dengan tangan [Le Cun et al.. 1990]. Walaupun aplikasi tersebut berdasarkan algoritma pelatihan yang biasa, tetapi dirasa cukup baik untuk mengembangkan suatu aplikasi, dimana jaringan *backpropagation* tersebut memiliki beberapa *hidden layer*, tetapi pola koneksi dari satu layer ke yang lain sudah dapat dibatasi.

2.3.2.2. Signal Processing

Banyak aplikasi JST yang ada pada teknologi *signal processing*. Salah satu aplikasi komersil pertama yang dihasilkan adalah untuk mengurangi *noise* pada suatu jaringan telepon. JST yang digunakan adalah jaringan *Adaline*. Jaringan

Adaline tersebut dilatih untuk menghilangkan *noise* (gangguan) dari sinyal keluaran hibrid (komponen sistem telepon).

2.3.2.3. Bidang Medis

Salah satu contoh aplikasi dari JST yang digunakan pada bidang medis, pertama kali dikembangkan oleh Anderson [Anderson, 1986; Anderson, Golden dan Murphy, 1986] pada pertengahan tahun 1980. Aplikasi tersebut dinamakan dengan "*Instant Physician*", dimana ide dari aplikasi tersebut adalah melatih sebuah JST memori autoasosiatif untuk menyimpan record – record medis dalam jumlah yang besar, dimana tiap record tersimpan informasi tentang diagnosa dan perawatan untuk suatu kasus medis. Sehingga setelah proses pelatihan, JST tersebut dapat mengenali suatu input kasus medis sehingga muncul ouput diagnosa dan cara perawatan yang terbaik.

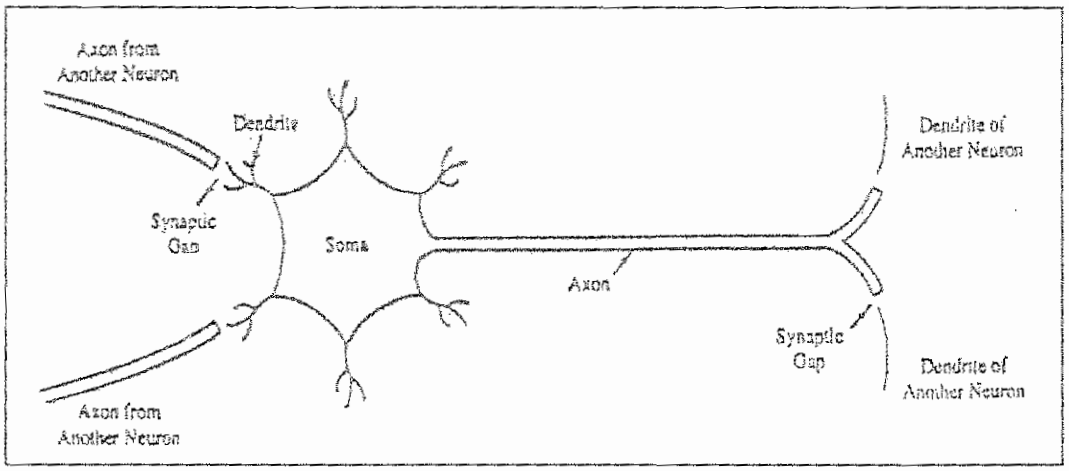
2.3.3. Jaringan Saraf Manusia

Jaringan saraf manusia memiliki 3 komponen yaitu *dendrites*, *soma* dan *axon*. Beberapa *dendrite* menerima sinyal dari *neuron* lain, dimana sinyal tersebut berupa impuls elektrik yang dikirim melewati celah *synaptic* melalui proses kimia. *Soma* atau *cell body*, menjumlahkan sinyal - sinyal yang masuk. Suatu *cell* dapat mengetahui sejumlah input diterima, dengan cara mengirimkan sinyal melalui *axon* ke *cell* lain. Sedangkan transmisi dari suatu signal yang berasal dari neuron tertentu disempurnakan oleh sebuah hasil aksi potensial dari konsentrasi ion - ion yang berbeda pada sisi lain *neuron's axon.*[Fausett, 1994, p5].

Jaringan saraf manusia diilustrasikan seperti Gambar 2.3 yang memiliki *axon* dari dua *neuron* yang berbeda dan *dendrite* untuk dua *neuron* yang berbeda. Beberapa karakteristik elemen – elemen pengolah dari JST yang memiliki kesamaan dengan jaringan saraf manusia [Fausett, 1994, p5-7], adalah :

1. Elemen – elemen pengolah tersebut menerima banyak sinyal
2. Sinyal – sinyal tersebut dapat dimodifikasi melalui sebuah *weight* pada *synapse* penerimaan.
3. Elemen – elemen pengolah mampu menjumlahkan *weight* dari input.
4. Dalam keadaaan tertentu (jumlah input yang mencukupi), *neuron* dapat mengirim sebuah input.
5. Output dari *neuron* tertentu dapat berpindah ke *neurons* lain (cabang-cabang *axon*).
6. Pengolahan informasi terjadi secara lokal.
7. Memori didistribusikan sebagai berikut :
   a. Memori *long-term*, yang terletak pada *synapse* atau *weights* dari *neurons*.
   b. Memori *short-term*, yang dapat disamakan dengan sinyal yang terkirim melalui *neuron*.
8. Kekuatan dari suatu *synapse* dapat berubah sesuai dengan pengalaman (*experience*).
9. *Neurotransmitters* untuk suatu *synapse* kemungkinan *excitatory* atau *inhibitory*.

Namun masih terdapat karakteristik lain dari JST yang memiliki kesamaan dengan jaringan saraf manusia, yaitu toleransi kesalahan. Dimana jaringan saraf manusia dapat menemukan kesalahan dalam dua hal. Pertama, jaringan saraf manusia dapat mengenali banyak sinyal input yang berbeda dari sinyal yang pernah dilihat sebelumnya. Kedua, jaringan saraf manusia menemukan kerusakan dari sistem saraf itu sendiri.



Gambar 2.3 Jaringan Saraf Manusia

2.3.4. Komponen - Komponen Jaringan Saraf Tiruan

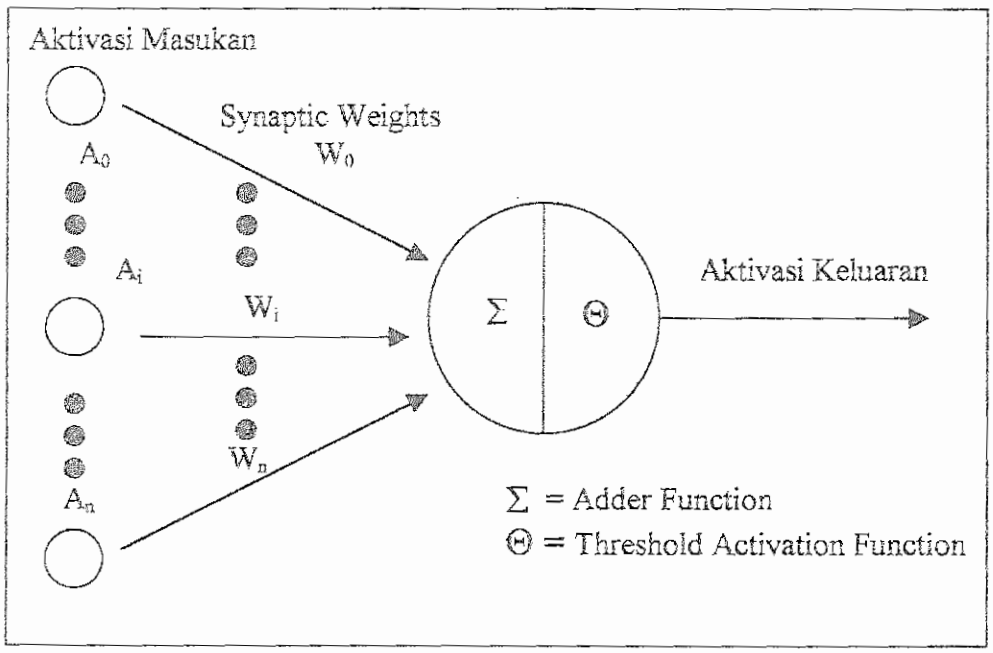
Aspek -aspek utama dari Jaringan Saraf Tiruan (JST) yang perlu diperhatikan [Rumelhart dan McClelland, 1986, p45-54], sebagai berikut :

1. Himpunan unit - unit pemroses

   Unit - unit pemroses ini berupa *neuron* tiruan yang pertama kali dikemukakan pada tahun 1940 oleh McCulloch dan Pitts. *Neuron* tersebut dideskripsikan dengan model sebagai berikut :

$$y = g\left(\sum_{i=0}^{N} x_i w_i\right) \qquad (2.4)$$

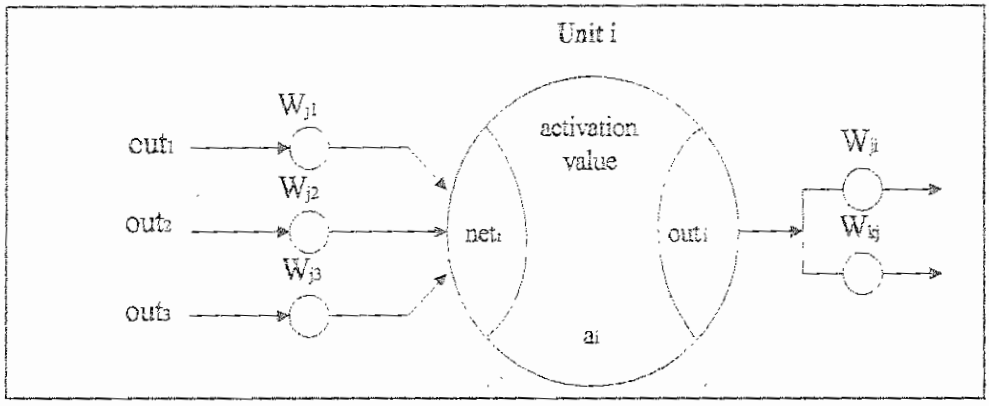
dimana y adalah keluaran dari sebuah *neuron*, N adalah jumlah masukan dan x_i adalah variabel - variabel masukan. W_i disebut *synaptic weights* (Gambar 2.4). Fungsi g adalah sebuah fungsi yang *nonlinear* [Ojala, 1994, p13].



Gambar 2.4 Contoh Sebuah *Neuron*

Misalkan N adalah banyaknya unit dan u_i adalah unit ke-i (Gambar 2.5). Tugas sederhana yang dilakukan masing - masing unit pemroses yaitu menerima masukan dari unit lain dan menghitung sebuah sinyal keluaran kemudian disebarkan ke unit - unit lain, tugas lain dari unit pemroses ini adalah menyesuaikan bobot. Unit - unit pemroses dibagi menjadi tiga tipe di dalam jaringan saraf, yaitu :

- Unit - unit masukan (*input units*) yang menerima data dari luar jaringan.
- Unit - unit keluaran (*output units*) yang mengirimkan data keluar dari jaringan.
- Unit - unit tersembunyi (*hidden units*) tempat sinyal masukan dan sinyal keluaran berada dalam jaringan.



Gambar 2.5 Model Umum Unit Pemroses

2. Keadaan Aktivasi (*State of Activation*)

Keadaan dari jaringan pada saat *t* yang direpresentasikan oleh sebuah vektor yang terdiri dari *N* bilangan nyata *a(t)*. Elemen - elemen vektor tersebut merupakan nilai aktivasi dari sebuah unit pada saat *t* dan nilai aktivasi dari unit u_i dimana *t* adalah $a_i(t)$, $a(t) = [a_1(t)\ a_2(t)\ \ldots\ a_N(t)]$. Nilai yang didapat bisa *bipolar* yang mempunyai nilai 1 dan -1 atau *biner* yang mempunyai nilai 0 dan 1. Vektor tersebut disebut sebagai *short-term memory* (STM) dari jaringan.

3. Fungsi Keluaran (*output function*)

Fungsi keluaran untuk unit u_i, f_i (a(t)), memetakan keadaan aktivasi pada saat t, $a_i(t)$, ke sinyal keluaran $out_i(t)$. Kemudian nilai keluaran tersebut dikirimkan ke unit lain dari jaringan.Tiga kemungkinan dari fungsi keluaran ini, antara lain:

- Fungsi *linear* atau *semi-linear* :

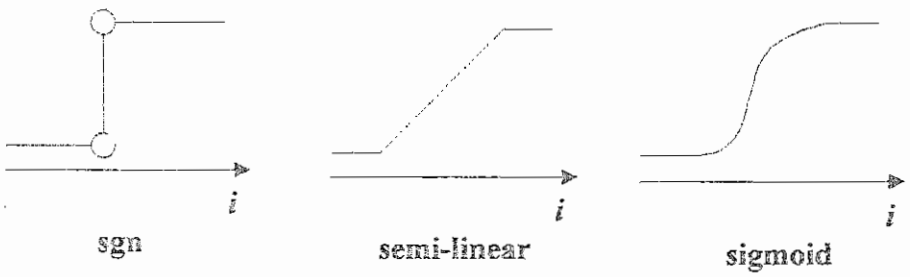
  $out_i = Gain * a_i$, *Gain* adalah sebuah nilai.

- Fungsi *threshold* (sgn) :

  If $a_i >$ Threshold then $out_i = 1$ else if $out_i = 0$

- Fungsi *sigmoid* :

  $out_i = 1/[1+exp(-a_i)]$



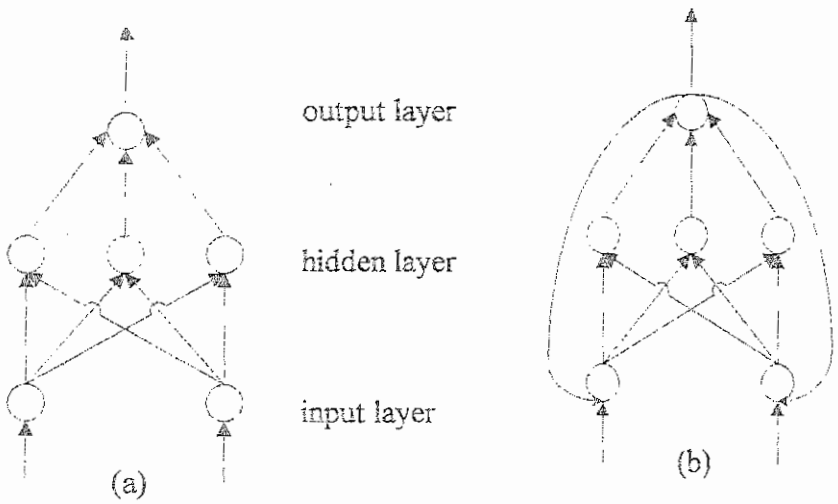
Gambar 2.6 Fungsi Keluaran/Aktivasi

4. Pola Hubungan (*Pattern of Connectivity*)

Kekuatan dari setiap hubungan direpresentasikan oleh sebuah bilangan nyata w. w_{ij} adalah bobot dari unit j ke unit i. Pola hubungan dari seluruh jaringan dengan N unit dapat direpresentasikan oleh matriks bobot W dengan dimensi NxN. Baris i dari W berisi bobot yang diterima oleh unit i dari unit - unit lain dalam jaringan sedangkan kolom j berisi semua bobot yang dikirim oleh unit j ke unit - unit yang lain. Pengetahuan disimpan dalam jaringan pada

matriks bobot ini, oleh sebab itu matriks *W* berisi *long-term memory* (LTM) dari jaringan [Cairo, 1994, p19-20].

Berdasarkan topologi jaringan, JST diklasifikasikan sebagai *feedforward* dan *feedback* atau *recurrent nerwork* (Gambar 2.7). Pada *feedforward* sebuah unit hanya mengirimkan keluarannya ke unit - unit yang tidak menerima masukan secara langsung atau tidak langsung yang artinya tidak ada putaran balik (*feedback loop*). *Feedforward* diatur dalam lapisan - lapisan yang unit - unitnya hanya dihubungkan ke unit - unit di lapisan selanjutnya secara berurutan. Sedangkan *feedback* terdapat putaran balik dan secara umum pada pemetaannya menggunakan *nonlinear dynamical system*, sehingga stabilitas dari jaringan adalah masalah utama.



Gambar 2.7 Feedforward (a) dan Feedback (b) network

5. Peraturan Penyebaran (*Propagation Rule*)

Peraturan penyebaran menentukan bagaimana nilai - nilai keluaran dari unit - unit lain dikombinasikan menjadi himpunan nilai yang lebih kecil, biasanya

berupa nilai tunggal yang disebut *net input* dari sebuah unit, oleh sebab itu kadang - kadang peraturan penyebaran itu disebut *combining function*. *Combining function* mendefinisikan sebuah nilai *net input* untuk setiap unit sebagai sebuah *weighted summation* atau $net(t) = W(t)out(t)$, dimana $net(t)$ adalah *net input* dan $out(t)$ adalah vektor - vektor keluaran.

6. Peraturan Aktivasi (*Activation Rule*)

   Secara umum peraturan aktivasi didefinisikan : $a(t+1) = F(a(t), net(t)_1, net(t)_2, ...)$. Fungsi aktivasi (F) adalah fungsi yang *differentiable* dan biasanya berupa fungsi *sigmoid*, fungsi *semi-linear* atau fungsi *threshold* (Gambar 2.6).

7. Peraturan Belajar (*Learning Rule*)

   Peraturan belajar digunakan untuk mengubah elemen - elemen dari matriks W dan parameter - parameter lain yang dimiliki oleh jaringan. Dua tipe utama dari *learning* yaitu *supervised* dan *unsupervised*. Pada *supervised learning* diperlukan pasangan setiap masukan dengan keluaran yang diinginkan, pasangan tersebut disebut *training pair*. Satu vektor masukan diaplikasikan kemudian keluarannya dihitung dan dibandingkan dengan target keluarannya, hasil selisih dikembalikan ke jaringan dan bobotnya disesuaikan berdasarkan suatu algoritma *learning* yang cenderung meminimumkan *error*. Sedangkan *unsupervised learning* tidak diperlukan target keluaran. Algoritma *learning* mengubah bobot jaringan untuk menghasilkan keluaran yang konsisten.

   Algoritma *learning* saat ini sebagian besar dikembangkan berdasarkan konsep *Hebbian Learning* dari Donald O. Hebb (1961), dimana secara simbol dapat

dituliskan sebagai berikut : $W_{ij}(t+1) = W_{ij}(t) + \eta.out_i.out_j$, dimana $W_{ij}(t)$ adalah bobot dari unit *i* ke unit *j* sebelum penyesuaian, $W_{ij}(t-1)$ adalah bobot dari unit *i* ke *j* sesudah penyesuaian, η adalah *learning rate*, out_i adalah keluaran dari unit *i* dan masukan dari unit *j*, out_j adalah keluaran dari unit *j*.

8. Lingkungan Luar (*External Environment*)

   Lingkungan luar berinteraksi dengan jaringan dimana dapat menyediakan masukan ke jaringan dan menerima keluaran dari jaringan tersebut.

2.3.5. Model Jaringan Saraf Tiruan

Banyak sekali terdapat model - model Jaringan Saraf Tiruan (JST), diantaranya : *Backpropagation, Counterpropagation,* dan *Bidirectional Associative Memory (BAM). Backpropagation* menunjukkan sebuah jaringan *multilayer* dengan menggunakan *supervised learning.* Sebagai penegasan kembali, *backpropagation* ditetapkan untuk melatih bagian lapisan tersembunyi *weights* dalam mengatasi batu penghalang utama dalam melanjutkan perkembangan JST. Sedangkan *Counterpropagation* menggunakan sebuah metode *unsupervised learning*, dimana beberapa aplikasi lebih dapat dipraktekkan daripada *backpropagation*. Sistem BAM menggunakan sebuah model yang membangun dua lapisan jaringan yang sederhana, dan teknik dari penggabungan untuk mengatasi kapasitas limitasi. [Blum, 1992, p53-54]

Ada berbagai cara mengklasifikasikan model - model JST yang ada, salah satunya dengan mengelompokkan berdasarkan tiga aspek penting dalam JST [Rao, 1993, p7-8], antara lain :

a. *Structure*

Aspek ini menghubungkan ke beberapa lapisan jaringan yang ada dan fungsi lapisan tersebut, seperti lapisan masukan dan lapisan keluaran.Struktur juga meliputi bagaimana antarhubungan dibuat antara neuron di dalam jaringan dan apa fungsinya.

b. *Encoding*

Aspek ini berhubungan dengan paradigma yang digunakan untuk menentukan dan memperbaiki bobot antar *neurons*.

c. *Recall*

Aspek ini berhubungan dengan bagaimana cara mendapatkan keluaran yang diharapkan dari masukan yang diberikan. Aspek ini merupakan bagian dari proses desain.

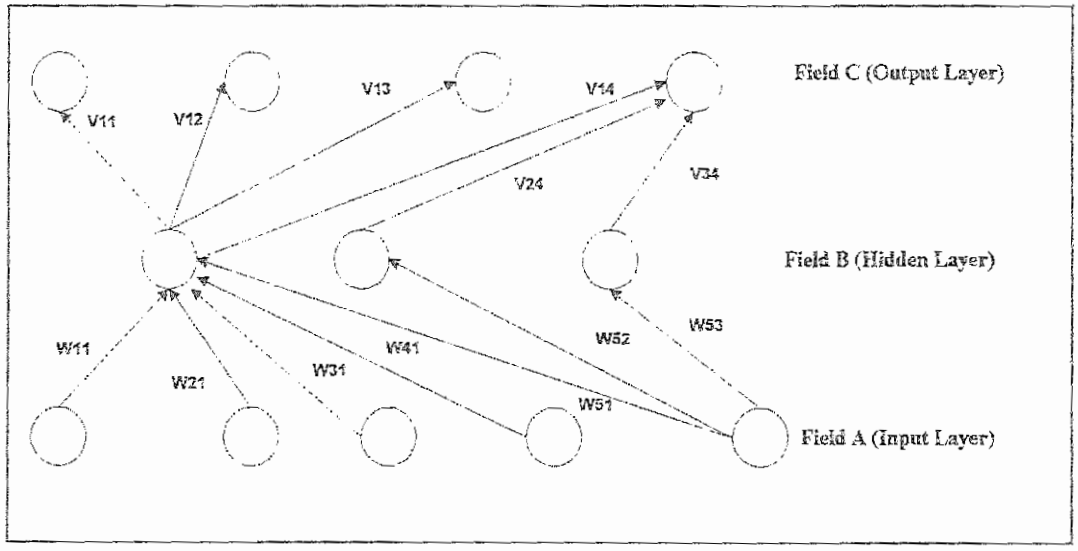
2.3.6. Backpropagation

Backpropagation yang sering disebut dengan *Generalized Delta Rule*, merupakan suatu metode pelatihan yang meminimalisasikan total *error* pada suatu keluaran yang dihitung oleh suatu jaringan. Jaringan *backpropagation* merupakan JST beberapa lapisan yang paling sedikitnya memiliki *input layer*, *hidden layer* dan *output layer*. Pada setiap lapisan terdapat *nodes* , yang biasanya secara *feedforward* pada lapisan masukan terhubung dengan *nodes*

yang ada pada lapisan tersembunyi, dan *nodes* yang ada pada lapisan tersembunyi terhubung sepenuhnya dengan *nodes* yang ada pada lapisan keluaran. [Fausett, 1994, p289].

Melatih suatu jaringan dengan menggunakan metode *backpropagation* meliputi tiga tahapan, yaitu: proses *feedforward* pada pola masukan untuk pelatihan, penghitungan dan *backpropagation* dari *error* yang ada, serta penyesuaian bobot. [Fausett, 1994, p290].

Walaupun jaringan *single-layer* memiliki keterbatasan dalam *mapping* proses pembelajaran, jaringan *multilayer* (dengan satu atau lebih lapisan tersembunyi). Arsitektur dari jaringan *backpropagation* dengan tiga layer dapat dilihat pada Gambar 2.8.



Gambar 2.8 Jaringan *Backpropagation*

Jaringan *backprogation* mengawasi pelatihan dengan menggunakan sebuah bilangan terbatas dari sepasang pola yang terdiri dari sebuah pola masukan dan

target pola keluaran. Sebuah pola masukan terdapat pada lapisan masukan. Keluaran dari saraf lapisan tersembunyi dihasilkan oleh bias dan juga fungsi sebuah *threshold* dengan menentukan pengerakkan oleh bobot dan masukan. [Rao, 1993, p103-105].

Algoritma *backpropagation* terdiri dari dua tahap, yaitu pemetaan vektor masukan menjadi vektor keluaran dan perhitungan *error*. Untuk fungsi aktivasi digunakan fungsi *sigmoid*.

Misalkan I adalah jumlah unit pada lapisan masukan (*input layer*), J adalah jumlah unit pada lapisan tersembunyi (*hidden layer*) dan K adalah jumlah unit pada lapisan keluaran (*output layer*). Dimana P *training pairs* $\{x_1, d_1, x_2, d_2, \ldots, x_P, d_P\}$, x_i adalah vektor masukan *(Ix1)*, d_i adalah vektor yang diharapkan (Kx1) dan $i = 1, 2, \ldots, P$. Sedangkan y adalah keluaran dari lapisan tersembunyi dan z adalah keluaran dari lapisan keluaran. Nilai aktivasi dari lapisan tersembunyi adalah a dan pada lapisan keluaran adalah b. V adalah bobot dari lapisan masukan ke lapisan tersembunyi dan W adalah bobot dari lapisan tersembunyi ke lapisan keluaran. Nilai bias pada lapisan keluaran adalah τ dan nilai bias pada lapisan tersembunyi adalah θ.

Bobot V dan W akan diinisialisasikan secara acak dengan nilai antara -1 dan 1. V_{ij} = random (-1,1), utuk $i = 1,2,\ldots,I$ dan $j = 1,2,\ldots,J$. W_{jk} = random (-1,1) untuk $j = 1,2,\ldots,J$ dan $k = 1,2,\ldots,K$. Kemudian inisialisasikan nilai ambang batas / bias secara acak antara -1 dan 1. θ_j = random (-1,1) untuk $j = 1,2,\ldots, J$. τ_k = random (-1,1) untuk $k = 1,2,\ldots, K$.

Penghitungan aktivasi dan keluaran pada lapisan tersembunyi dengan mempropagasikan aktivasi dari unit - unit pada lapisan masukan ke unit - unit pada lapisan tersembunyi :

$$a_j = \left[\sum_{i=1}^{I} V_{ij} x_i\right] + \theta_j \qquad \text{untuk j = 1,2,...,J} \qquad (2.5)$$

$$y_j = \frac{1}{1+e^{-a_j}} \qquad \text{untuk j = 1,2,...,J} \qquad (2.6)$$

Propagasikan aktivasi dari unit - unit pada lapisan tersembunyi ke unit - unit pada lapisan keluaran :

$$b_k = \left[\sum_{j=1}^{J} W_{jk} y_j\right] + \tau_k \qquad \text{untuk k = 1,2,...,K} \qquad (2.7)$$

$$z_k = \frac{1}{1+e^{-b_k}} \qquad \text{untuk k = 1,2,...,K} \qquad (2.8)$$

Penghitungan kesalahan pada lapisan keluaran berdasarkan pada *actual output* (z) dan keluaran yang diharapkan (d) yang dinotasikan dengan $\delta 2$:

$$\delta 2_k = y_j(1-z_k)(d_k - z_k) \qquad \text{untuk k = 1,2,...,K} \qquad (2.9)$$

Penghitungan kesalahan pada lapisan tersembunyi yang dinotasikan dengan $\delta 1$:

$$\delta 1_j = y_j\left(1-y_j\right)\sum_{k=1}^{K} \delta 2_k W_{jk} \qquad \text{untuk j = 1,2,...,J} \qquad (2.10)$$

Penghitungan dalam mengatur bobot W :

$$\Delta W_{jk} = \eta y_j \delta 2_k \qquad \text{untuk j = 1,2,...,J dan k = 1,2,...,K} \qquad (2.11)$$

Penghitungan dalam mengatur bobot V :

$$\Delta V_{ij} = \eta x_i \delta 1_j \qquad \text{untuk i = 1,2,...,I dan j = 1,2,...,J} \qquad (2.12)$$

Menyesuaikan nilai bias untuk lapisan keluaran $(\Delta\tau)$ dan nilai bias untuk lapisan tersembunyi $(\Delta\theta)$.

$$\Delta\tau_k = \eta\delta 2_k \qquad \text{untuk k = 1,2,...,K} \qquad (2.13)$$

$$\Delta\theta_j = \eta\delta 1_j \qquad \text{untuk j = 1,2,...,J} \qquad (2.14)$$

Dengan mengulangi penghitungan aktivasi dan keluaran pada lapisan tersembunyi dengan mempropagasikan aktivasi dari unit - unit pada lapisan masukan ke unit - unit pada lapisan tersembunyi sampai semua *training pair* telah di-*training*. Periode ini dikatakan sebagai satu *epoch*. Proses belajar akan selesai bila telah mencapai *error* yang diinginkan.

Algoritma tersebut berlaku untuk jaringan yang terdiri dari tiga lapisan (hanya satu lapisan tersembunyi).

Kecepatan belajar pada jaringan dapat ditingkatkan dengan memasukkan parameter *momentum* (α), sehingga modifikasi bobot pada lapisan masukan dan lapisan tersembunyi adalah :

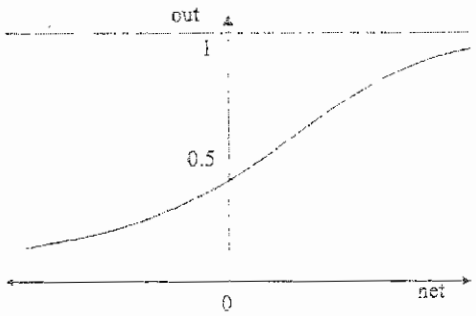
$$V_{ij}(t) = \eta x_i \delta 1_j + \alpha V_{ij}(t-1) \qquad (2.15)$$

dan modifikasi bobot pada lapisan tersembunyi dan lapisan keluaran adalah :

$$W_{jk}(t) = \eta y_j \delta 2_k + \alpha W_{jk}(t-1) \qquad (2.16)$$

dengan $x_i, y_i, \delta 1_j$ dan $\delta 2_k$ diukur pada waktu *t*.

Fungsi *sigmoid* digunakan untuk memodifikasi nilai bobot, karena fungsi ini adalah fungsi yang bersifat kontinu dan dapat diturunkan (Gambar 2.9).



$$out = F(net) = \frac{1}{1+e^{-net}}$$

$$F'(net) = \frac{\partial out}{\partial net} = out(1-out)$$

Gambar 2.9 Fungsi Sigmoid

Dengan adanya fungsi ini, maka keluaran pada unit keluaran adalah sebuah vektor dengan komponen nilai antara 0 dan 1.