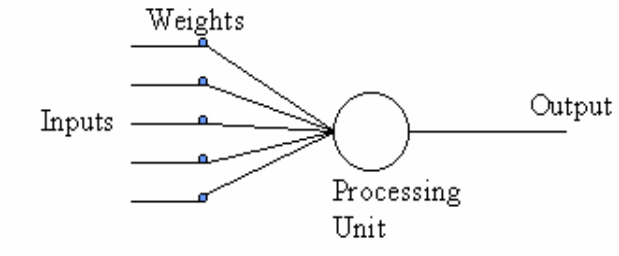
BAB II

NEURAL NETWORK (NN)

2.1 *Neural Network* (NN)

Secara umum *Neural Network* (NN) adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan jaringan syaraf manusia. NN ini merupakan sistem adaptif yang dapat merubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Secara sederhana NN adalah sebuah alat pemodelan data statistik non-linear. NN dapat digunakan untuk memodelkan hubungan yang kompleks antara input dan output untuk menemukan pola-pola pada data.

Secara mendasar, sistem pembelajaran merupakan proses penambahan pengetahuan pada NN yang sifatnya kontinuitas sehingga pada saat digunakan pengetahuan tersebut akan dieksploitasikan secara maksimal dalam mengenali suatu objek. *Neuron* adalah bagian dasar dari pemrosesan suatu *Neural Network*. Dibawah ini merupakan bentuk dasar dari suatu *neuron.*



Gambar 2.1 Bentuk dasar *neuron*

Keterangan Gambar 2.1 di atas adalah sebagai berikut.

1. *Input* merupakan masukan yang digunakan baik saat pembelajaran maupun dalam mengenali suatu objek.
2. *Weight*, beban yang selalu berubah setiap kali diberikan *input* sebagai proses pembelajaran.
3. *Processing Unit* merupakan tempat berlangsungnya proses pengenalan suatu objek berdasarkan pembebanan yang diberikan.
4. *Output*, keluaran dari hasil pengenalan suatu objek.

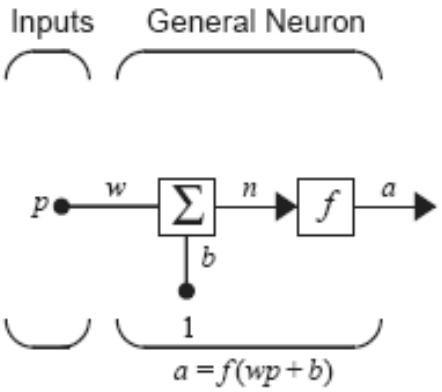
Keuntungan penggunaan *Neural Network.*

1. Perangkat yang mampu untuk mengenali suatu objek secara non-linier.
2. Mempermudah pemetaan *input* menjadi suatu hasil tanpa mengetahui proses sebenarnya.
3. Mampu melakukan pengadaptasian terhadap pengenalan suatu objek.
4. Perangkat yang memiliki toleransi terhadap suatu kesalahan dalam pengenalan suatu objek.
5. *Neural Network* mampu diimplementasikan pada suatu *Hardware* atau perangkat keras.

2.2 *Neuron* Model

2.2.1 *Single-Input Neuron*

Sebuah *neuron* dengan sebuah masukan (*single-input*) dapat dilihat pada Gambar 2.2 dibawah. Dalam gambar dapat dilihat sebuah *Input* (p) dikalikan dengan *weight* (w) kemudian ditambahkan dengan bobot *bias* (b) dan menghasilkan suatu keluaran (n), suatu nilai keluaran ini biasanya disebut dengan *net input*, selanjutnya masuk ke dalam suatu fungsi transfer (*f*) dan akan menghasilkan *output neuron* (a).



Gambar 2.2 *Single-Input Neuron*

Keluaran dari *neuron* dapat dihitung dengan rumus :

$$a = f\,(wp + b) \tag{2.1}$$

Sebagai contoh, jika bobot (*w*)= 3, input (*p*) = 2 dan bobot bias (*b*) = -1.5

maka :

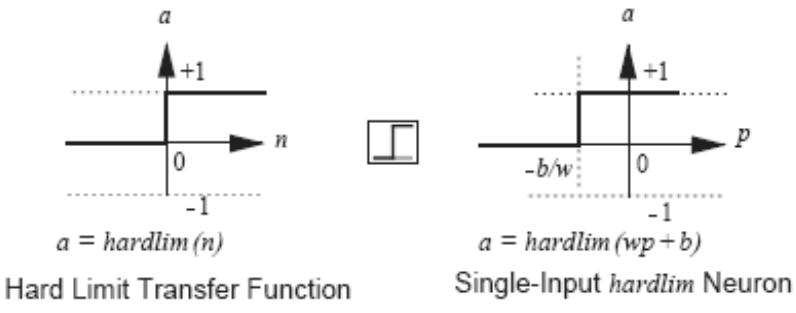
$$a = f(3(2) - 1.5 = f\,(4.5) \tag{2.2}$$

Keluaran aktual dari suatu *Neural Network* tergantung kepada suatu fungsi transfer yang digunakan. Biasanya, untuk suatu fungsi transfer dipilih oleh seorang yang merancang akan menggunakan beberapa aturan sehingga masukan atau keluaran *neuron* akan melakukan pembelajaran untuk memenuhi beberapa tujuan khusus. Sebagaimana akan diuraikan dalam begian berikut, di bawah dapat dilihat beberapa fungsi transfer untuk tujuan yang berbeda, seperti berikut.

2.2.2 *Transfer Functions*

Fungsi transfer untuk Gambar 2.2 diatas mungkin saja suatu fungsi linear atau bisa juga suatu fungsi non-linear. Sebuah fungsi transfer tertentu dapat dipilih untuk memecahkan beberapa masalah yang spesifik, sehingga dapat disimpulkan bahwa suatu *neuron* sedang memecahkan suatu masalah tersebut.

Fungsi transfer *hard limit*, yang dapat dilihat di bagian kiri pada Gambar 2.3 di bawah, menunjukkan set keluaran dari *neuron* pada harga nol (0) jika argument dari fungsi kurang dari 0, atau 1 jika argument lebih dari atau sama dengan nol (0). Maka fungsi *Hard Limit* ini akan menciptakan sebuah *neuron* yang mengklasifikasikan *input* kedalam dua kategori yang berbeda.



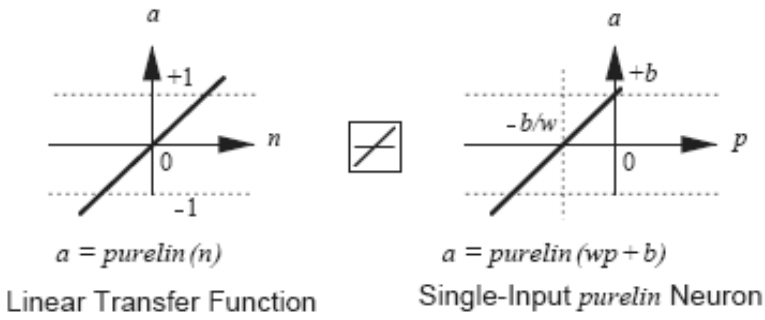
Ganbar 2.3 Fungsi Transfer *Hard Limit*

Gambar pada sisi kanan dari Gambar 2.3, menggambarkan karakteristik *input/output* dari *single-input neuron* yang menggunakan fungsi transfer *hardlim*. Disini kita melihat efek dari *weight* dan *bias*. Pada Gambar 2.3 di atas dapat dilihat bentuk dari fungsi transfer *hardlim* yang diperlihatkan diantara kedua gambar. Secara umum, biasanya bentuk itu yang digunakan untuk menggantikan diagram jaringan yang menunjukkan suatu fungsi transfer tertentu yang sedang digunakan.

Selanjutnya fungsi transfer linear, untuk fungsi transfer ini nilai keluaran sama dengan nilai masukannya:

$$a = n \tag{2.3}$$

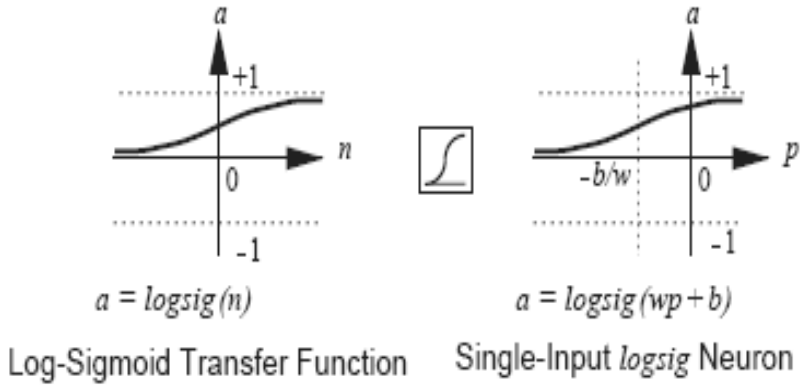
ilustrasi untuk fungsi transfer linear ini dapat dilihat pada Gambar 2.4 dibawah.



Gambar 2.4 Fungsi Transfer *Linear*

Pada Gambar 2.4 bagian kiri dapat diperlihatkan bahwa keluaran (a) sama dengan masukan (p), pada bagian kanan diperlihatkan suatu karakteristik dari suatu *single-input linear neuron* dengan *bias*.

Fungsi transfer *log-sigmoid* dapat dilihat pada Gambar 2.5 dibawah ini.



Gambar 2.5 Fungsi Transfer *Log-Sigmoid*

Fungsi transfer ini menangani masukan (yang mungkin mempunyai nilai minus maupun plus yang tidak terbatas) dan menghimpit keluaran pada *range* 0 sampai 1, sesuai dengan pernyataan :

$$a = \frac{1}{1 + exp^{-n}} \tag{2.4}$$

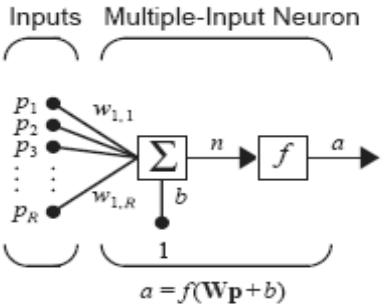
Fungsi transfer *log-sigmoid* pada umumnya digunakan pada *multilayer network* dengan menggunakan pelatihan *backpropagation algoritm*. Fungsi transfer lain dapat dilihat pada Tabel II.1 dibawah.

Tabel II.1 T*ransfer Functions*

Name	**Input/Output Relation**	**Icon**	**Function Mathlab**
Hard Limit	a = 0, n < 0 a = 1, n ≥ 0		Hardlim
Symmetrical Hard Limit	a = -1, n < 0 a = +1, n ≥ 0		Hardlims
Linear	a = n		purelin
Saturation Linear	a = 0, n < 0 a = n, 0 ≤ n ≤ 1 a = 1, n > 0		Satlin
Symmetrical Saturation Linear	a = -1, n < -1 a = n, -1 ≤ n ≤ 1 a = 1, n > 1		Satlins
Log-Sigmoid	$a = \frac{1}{1 + exp^{-n}}$		Logsig
Hyperbolic Tangent Sigmoid	$a = \frac{2}{1 + \exp(-2n)} - 1$		Tansig
Positive Linear	a = 0, n < 0 a = n, 0 ≤ n		Poslin
Competitive	a =1 neuron with max n a =0 all other neuron		compet

2.2.3 *Multiple Input Neuron*

Secara khas, suatu *neuron* mempunyai *input* lebih dari satu. Suatu *neuron* dengan *R input* diperlihatkan dalam Gambar 2.6. masukan tunggal $p_1,p_2,\ldots,p_R$ masing-masing di beri beban oleh unsur-unsur yang bersesuaian dengan $w_{1,1}w_{1,2},\ldots..w_{1,R}$ dari matriks bobot **W.**



Gambar 2.6 *Multiple-Input Neuron*

Neuron mempunyai suatu *bias*, dimana *bias* tersebut dijumlahkan dengan bobot masukan untuk membentuk suatu jaringan :

$$n = w_{1,1}\,p_1 + w_{1,2}p_2 + \ldots + w_{1,R}\,p_R + b \qquad (2.5)$$

Pernyataan ini mungkin di tulis dalam bentuk matriks :

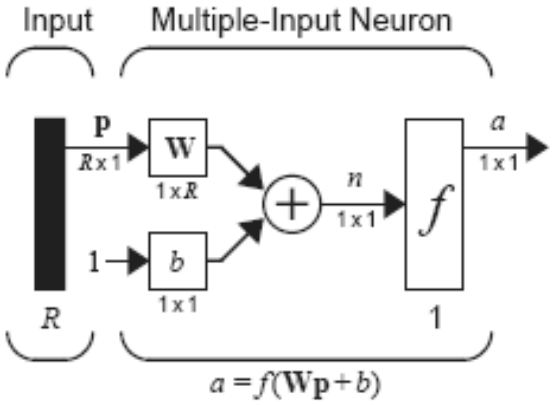
$$n = Wp + b \qquad (2.6)$$

Dimana acuan matriks **W** untuk kasus *single-neuron* mempunyai satu baris. Sekarang keluaran *neuron* dapat ditulis menjadi :

$$a = f\,(Wp + b) \qquad (2.7)$$

Neural Network bisa digambarkan dengan acuan matriks. Kita telah mengadopsi satu konversi tertentu dalam memberikan indeks dari unsur-unsur bobot matriks. Pertama menunjukkan tujuan tertentu untuk bobot tersebut. Kedua menunjukkan sumber sinyal dan dimasukkan ke dalam *neuron*. Dengan begitu, indeks dalam $w_{1,2}$ menyatakan bahwa bobot ini merepresentasikan koneksi di *neuron* pertama dari sumber kedua. Tentu saja, konvensi ini lebih bermanfaat jika ada lebih dari satu *neuron*.

Jika kita bermaksud untuk menggunakan suatu jaringan dengan beberapa *neuron*, dengan masing-masing neuron mempunyai beberapa *input*. selanjutnya, kita bermaksud meggunakan neuron tersebut dengan lapisan layer lebih dari satu. Maka sebuah *neuron* dengan banyak masukan menggunakan notasi ini dapat dilihat pada Gambar 2.7 di bawah ini.



Gambar 2.7 *Neuron* dengan R *Input*

Yang diperlihatkan dalam Gambar 2.7, masukan vektor p dipresentasikan oleh garis tebal pada sisi kiri. Dimensi dari p diperlihatkan sebagai variabel dari R x 1,

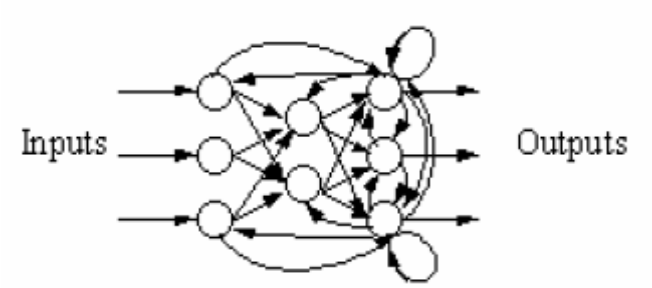
indikasi dari *input* merupakan *single-vector* dari elemen R. *input* tersebut menunjuk pada bobot matriks **W** pada kolom R, tetapi pada kotak *single neuron* ini hanya memiliki satu garis. Satu masukan *neuron* yang konstan dan dikalikan dengan *bias* b. dan menghasilkan suatu keluaran (net *input*), yang selanjutnya menuju fungsi transfer *f*. yang masuk fungsi ini adalah penambahan dari *bias* dan hasil dari w_p. Keluaran *neuron a* pada fungsi terbebut bernilai skalar. Dan jika memiliki banyak keluaran jaringan, maka akan menghasilkan suatu vektor.

Dimensi dalam suatu variabel pada blok diagram di atas akan selalu diikutsertakan, jadi bisa dikatakan bahwa yang sedang dibicarakan adalah tentang vektor, skalar atau matriks.

Jumlah input yang masuk pada suatu jaringan ditentukan oleh spesifikasi masalah eksternal. Sebagai contoh, jika ingin merancang suatu *Neural Network* seperti memprediksi kondisi dan syarat-syarat sebelum melakukan terbang layang (*kite-flying*), seperti temperatur, kecepatan angin dan kelembaban, kemudian jadilah tiga input yang akan masuk kedalam jaringan tersebut.

2.3 Arsitektur *Neural Network*

Bentuk dasar arsitektur suatu *Neural Network* adalah sebagai berikut :



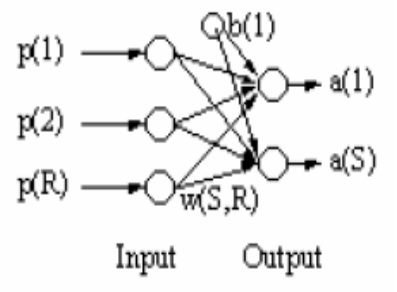
Gambar 2.8 Arsitektur Dasar *Neural Network*

Secara umum, terdapat tiga jenis *Neural Network* yang sering digunakan berdasarkan jenis *network*-nya, yaitu :

1. *Single-Layer Neural Network*
2. *Multilayer Perceptron Neural Network*
3. *Recurrent Neural Networks*

2.3.1 *Single-Layer Neural Network*

Neural Network jenis ini memiliki koneksi pada *input*nya secara langsung ke jaringan *output*.

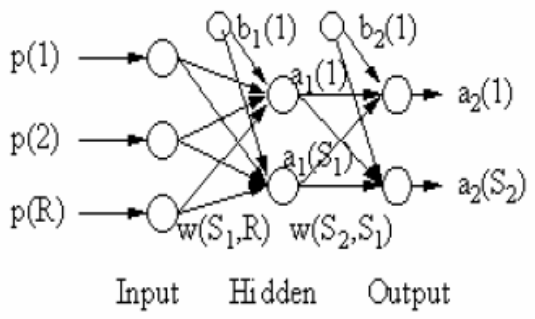


Gambar 2.9 *Single-layer Neural Network*

Jenis *Neural Network* ini sangatlah terbatas, hanya digunakan pada kasus-kasus yang sederhana.

2.3.2 *Multilayer Perceptron Neural Network*

Jenis *Neural Network* ini memiliki *layer* yang dinamakan "*hidden*", ditengah *layer input* dan *output*. *Hidden* ini bersifat *variable*, dapat digunakan lebih dari satu *hidden layer*.



Gambar 2.10 *Multilayer Perceptron Neural Network*

2.4 Proses Pembelajaran pada *Neural Network*

Proses pembelajaran merupakan suatu metoda untuk proses pengenalan suatu objek yang sifatnya kontinuitas yang selalu direspon secara berbeda dari setiap proses pembelajaran tersebut. Tujuan dari pembelajaran ini sebenarnya untuk memperkecil tingkat suatu *error* dalam pengenalan suatu objek. Secara mendasar, *Neural Network* memiliki sistem pembelajaran yang terdiri atas beberapa jenis berikut :

1. *Supervised Learning*
2. *Unsupervised Learning*

2.5.1 *Supervised Learning*

Sistem pembelajaran pada metoda *supervised learning* adalah sistem pembelajaran yang mana, setiap pengetahuan yang akan diberikan kepada sistem, pada awalnya diberikan suatu acuan untuk memetakan suatu masukan menjadi suatu keluaran yang diinginkan. Proses pembelajaran ini akan terus dilakukan selama kondisi *error* atau kondisi yang diinginkan belum tercapai. Adapun setiap perolehan *error* akan dikalkulasikan untuk setiap pemrosesan hingga data atau nilai yang diinginkan telah tercapai.

2.5.2 *Unsupervised Learning*

Sistem pembelajaran pada *Neural Network*, sepenuhnya memberikan hasil pada komputasi dari setiap pemrosesan, sehingga pada sistem ini tidak membutuhkan adanya acuan awal, agar perolehan nilai dapat dicapai. Meskipun secara mendasar, proses ini tetap mengkalkulasikan setiap langkah pada setiap kesalahannya dengan mengkalkulasikan setiap nilai *weight* yang didapat.