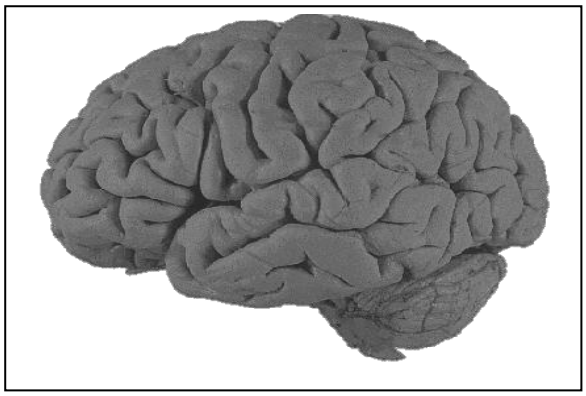
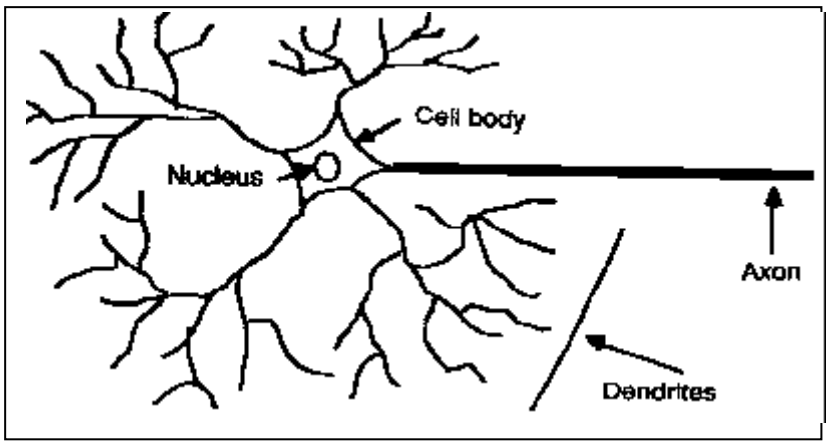
Jaringan Syaraf Tiruan
(Artificial Neural Networks)



BAB I PENDAHULUAN

1.1 Sejarah JST

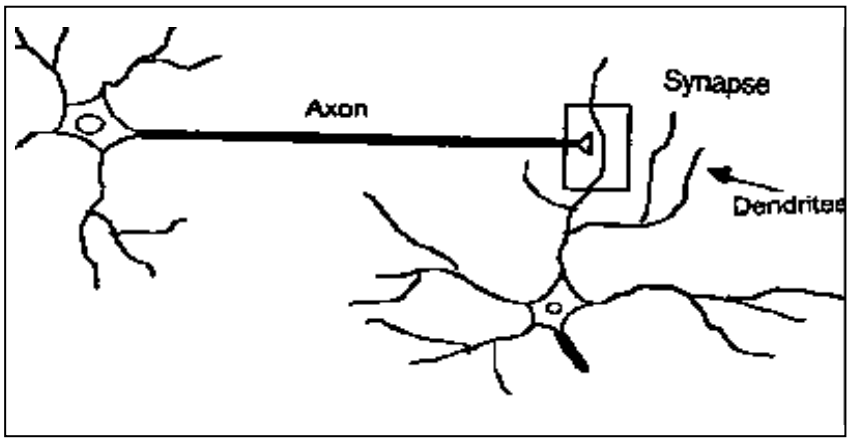
- JST : merupakan cabang dari Kecerdasan Buatan (Artificial Intelligence)
- JST : meniru cara kerja otak mahluk hidup yaitu *sel syaraf (neuron*)
- Otak manusia terdiri dari 10 milyar neuron yang saling berhubungan satu sama lain. Hubungan ini disebut dengan *synapses*
- Neuron secara garis besar dibagi 3 : *cell body, dendrites, dan axon*



Gambar 1.1. Struktur dari sebuah sel syaraf (neuron)

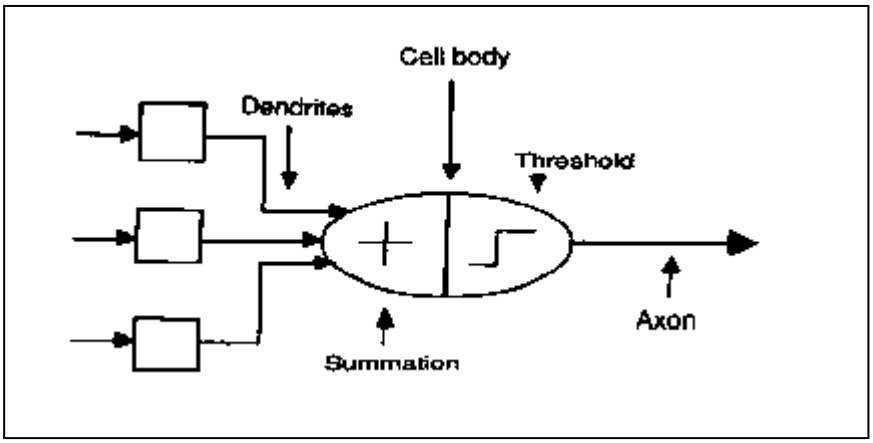
- Denrites : merupakan unit *input* yaitu sebagai tempat masuknya sinyal
- Cell body : berfungsi untuk memproses sinyal yang masuk
- Axon : merupakan unit *output* dari sinyal hasil proses cell body
- Hubungan antara 1 neuron dengan neuron yg lain lewat hubungan synapse
- Proses perjalanan sinyal yaitu :
  - mula-mula sinyal masuk lewat denrites menuju cell body
  - Kemudian sinyal diproses dalam cell body berdasarkan fungsi tertentu (Summation process). Jika sinyal hasil proses melebihi nilai ambang (*treshold*) tertentu maka sinyal tersebut akan membangkitkan neuron(*excited*) untuk meneruskan sinyal tsb., sedangkan jika di bawah treshold maka sinyal tersebut akan dihalangi (*inhibited*). Sinyal yg diteruskan akan menuju ke axon

dan akhirnya menuju ke neuron lainnya lewat *synapse* (yaitu celah sempit antara axon dari suatu neuron dan denrites dari neuron lainnya)



Gambar 1.2. Hubungan antara 2 buah neuron

- Model neuron digambarkan sbb:

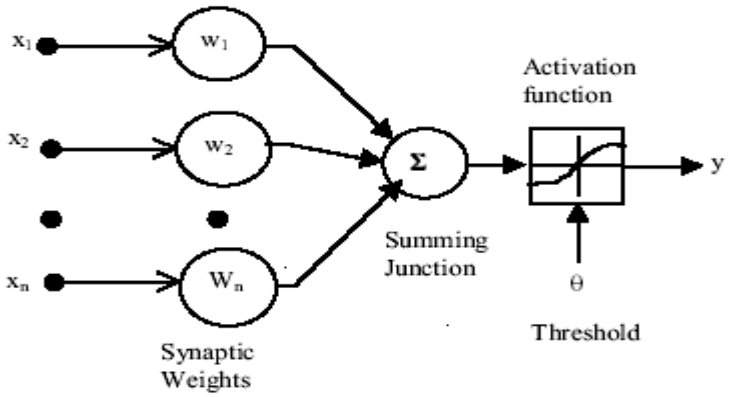


Gambar 1.3. Model neuron

- Model neuron inilah yang selanjutnya menjadi dasar dari **Jaringan Syaraf Tiruan** (*Artificial Neural Networks*)

1.2 Model JST

- Elemen dasar dari JST terdiri dari 3 bagian utama : bobot (*weight*), threshold, dan fungsi aktivasi



Gambar 1.4. Elemen dari JST

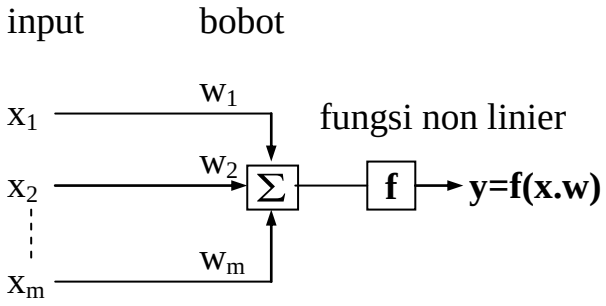
- input : $x_1,x_2,x_3,\ldots x_n$ adalah sinyal yg masuk ke sel syaraf
- Bobot(*weight*) : $w_1,w_2,w_3,\ldots w_n$ adalah faktor bobot yang berhubungan dengan masing-masing node. Setiap input akan dikalikan dengan bobot dari node-nya masing-masing, $x^T.w$. Tergantung dari fungsi aktivasi yang dipakai, nilai $x^T.w$ dapat membangkitkan (*excite*) node atau menghalangi (*inhibit*) node
- Threshold : nilai ambang internal dari node θ adalah besarnya offset yang mempengaruhi aktivasi dari output node y :

$$y = \sum_{i=1}^{n} X_i W_i - \theta \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

- Fungsi aktivasi : merupakan operasi matematik yang dikenakan pada sinyal output y. Ada beberapa fungsi aktivasi yang biasa dipakai dalam JST tergantung dari masalah yang akan diselesaikan.

1.3 Model Matematis JST

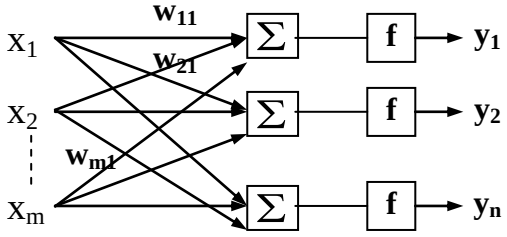
- Menyatakan topologi dari interkoneksi neuron-neuron dlm JST serta aturan-aturan yg dikenakan dalam jaringan tsb.
- Model matematis dari sebuah sel syaraf (neuron) diperlihatkan dlm gamb. 1.5 :



Gambar 1.5. Model matematis JST

- Jika dinyatakan dalam notasi matematika :
  $y = f(x_1w_1+x_2w_2+\ldots+x_mw_m)$ atau $y = f(\mathbf{x.w})$
  f = fungsi non linear atau fungsi aktivasi
- Ada beberapa model JST a.l. :
  - model JST satu lapisan
  - model JST banyak lapisan
  - model JST dua lapisan dgn umpan balik

a) Model JST satu lapisan



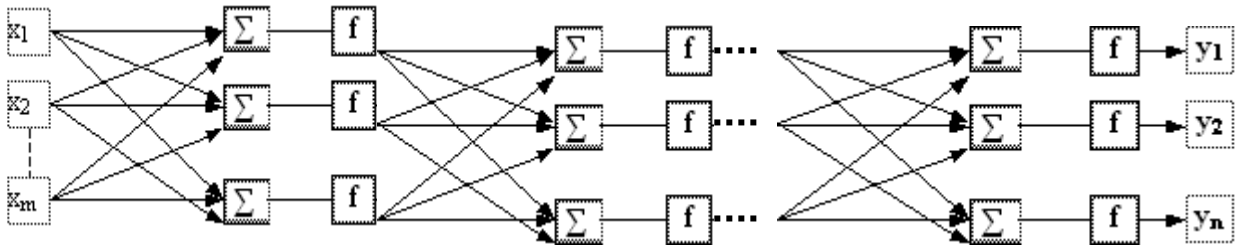
Gambar 1.6. Model JST satu lapisan

$y_1 = f(x_1w_{11}+x_2w_{21}+\ldots+x_mw_{m1})$

$y_2 = f(x_1w_{12}+x_2w_{22}+\ldots+x_mw_{m2})$

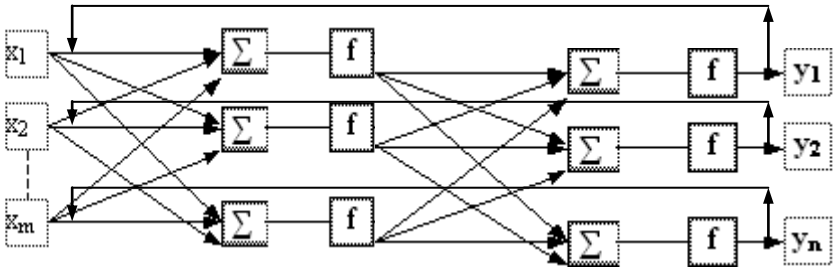
$y_n = f(x_1w_{1n}+x_2w_{2n}+\ldots+x_mw_{mn})$

b) Model JST banyak lapisan
- Dibentuk dari beberapa model JST satu lapisan
- Prinsip kerja hampir sama dgn model JST satu lapisan
- Output tiap lapisan sebelumnya merupakan input bagi lapisan sesudahnya



Gambar 1.7. Model JST banyak lapisan

c) Model JST dua lapisan dengan umpan balik
- Diperkenalkan oleh John Hopfield (1982)
- Model ini bersifat umpan balik yaitu output yg dihasilkan akan mempengaruhi input yg akan masuk lagi ke jaringan.

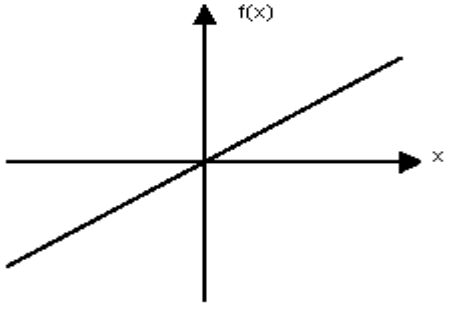


Gambar 1.8. Model JST 2 lapisan dengan umpan balik

1.4 Fungsi Aktivasi

- Beberapa fungsi aktivasi yang biasa dipakai dalam JST a.l. :

1.**Fungsi linear**
- Fungsi linear dinyatakan dengan persamaan : $y = f(x) = \alpha x$



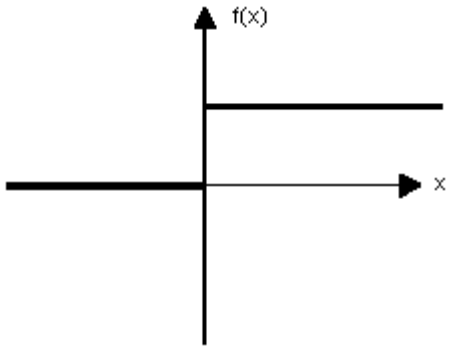
Gambar 1.9. Fungsi aktivasi linear

Dimana α adalah kemiringan dari fungsi. Jika $\alpha = 1$ maka fungsi aktivasi tsb adalah fungsi identitas

2.Fungsi threshold *(hard- limiter)*

- Fungsi threshold type : *biner* dan *bipolar*
- Fungsi threshold biner mempunyai output y yaitu :

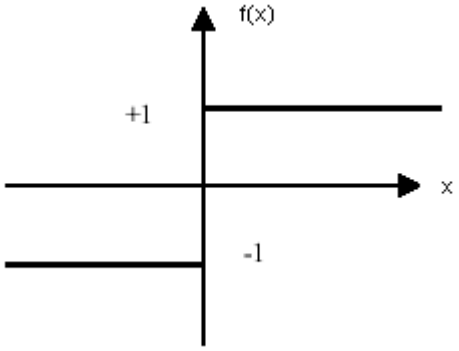
$$y = f(x) = \begin{cases} 0 \ jika \ x < 0 \\ 1 \ jika \ x \geq 0 \end{cases}$$



Gambar 1.10. Fungsi threshold biner

- Fungsi threshold bipolar mempunyai output y yaitu :

$$y = f(x) = \begin{cases} -1 \ jika \ x < 0 \\ +1 \ jika \ x \geq 0 \end{cases}$$

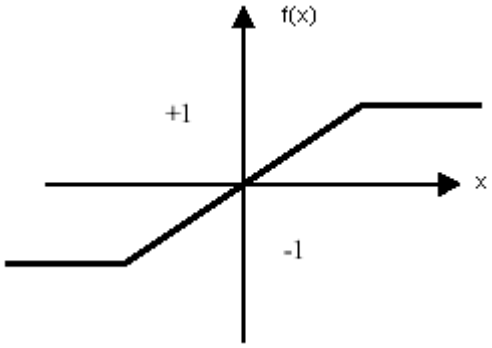


Gambar 1.11. Fungsi threshold bipolar

3.Fungsi linear piecewise

- Persamaan matematik dari fungsi ini adalah :

$$y = f(x) = \begin{cases} -1 \ jika & x < -1 \\ x \ jika & -1 \leq x \leq 1 \\ 1 \ jika & x \geq 1 \end{cases}$$

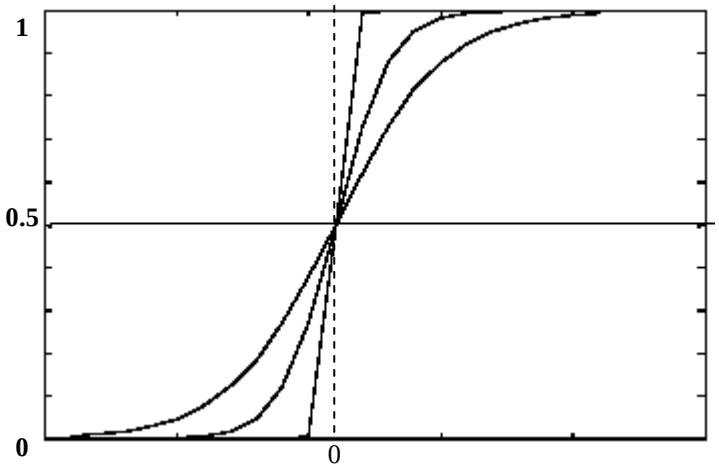


Gambar 1.12. Fungsi linear piecewise

4.Fungsi Biner Sigmoid

- Merupakan fungsi non linear yang paling banyak digunakan dalam JST
- Fungsi aktivasi sigmoid dapat ditulis seperti berikut :

$$y = f(x) = \frac{1}{1+e^{-\alpha x}} \quad untuk \quad 0 \le f(x) \le 1$$



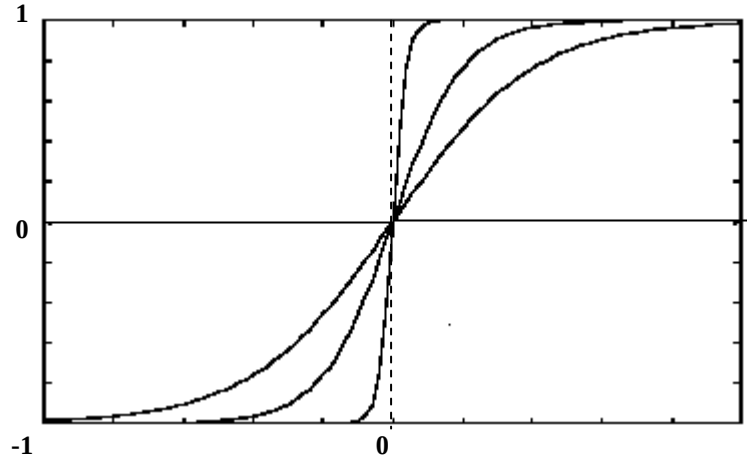
Gambar 1.13. Fungsi Biner Sigmoid untuk beberapa nilai α

- α adalah parameter bentuk dari fungsi sigmoid. Dengan mengubah harga α maka bentuk dari fungsi sigmoid akan berbeda-beda spt ditunjukkan gamb. 1.13

5.Fungsi Bipolar Sigmoid atau Tangen hiperbolik (tanh)

- fungsi ini dinyatakan dengan persamaan berikut ini :

$$y = f(x) = \frac{e^{\alpha x} - e^{-\alpha x}}{e^{\alpha x} + e^{-\alpha x}} \quad untuk \quad -1 \le f(x) \le 1$$

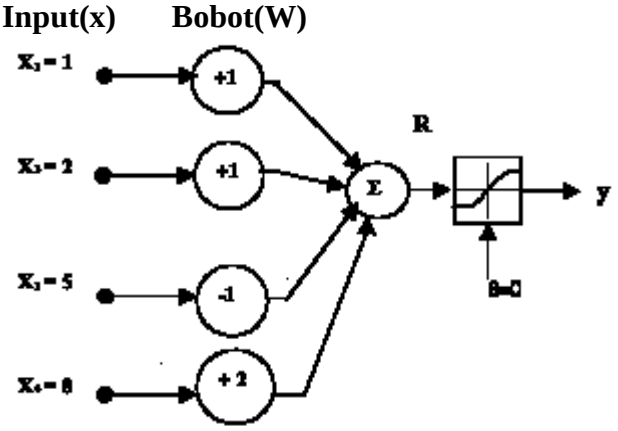


Gambar 1.14. Fungsi tanh untuk beberapa nilai α

- α adalah parameter bentuk dari fungsi tanh. Dengan mengubah harga α maka bentuk fungsi tanh akan berbeda-beda seperti ditunjukkan dalam gamb. 1.14

Contoh :

Andaikan kita mempunyai sebuah neural network dengan 4 input dan bobot seperti gambar 1.15.



Gambar 1.15. JST dengan 4 input dan bobot

maka output R dari neuron sebelum dikenai fungsi aktivasi adalah :

$$R = W^T.X = \begin{bmatrix} 1 & 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 5 \\ 8 \end{bmatrix} = 14$$

jika kita memakai fungsi aktivasi hard-limiter biner maka hasilnya :

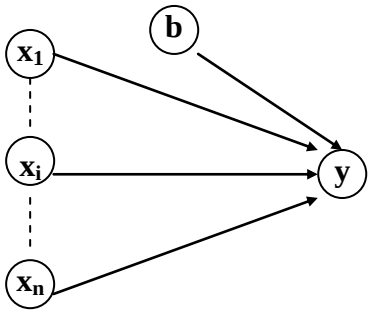
$$y = f(R) = f(14) = 1$$

dan jika dipakai fungsi aktivasi sigmoid maka :

$$y = f(R) = f(14) = \frac{1}{1+e^{-1.14}} = 1.5 * 2^{-8}$$

1.5 Bias dan Threshold

- Arsitektur paling dasar dari JST adalah JST satu lapisan : terdiri dari beberapa unit input dan satu unit output
- Biasanya dlm unit input ditambah suatu variabel yaitu bias(b) atau threshold (θ)



Gambar 1.16 JST satu lapisan dengan bias

- Jika kita memakai bias b maka nilai fungsi aktivasi akan menjadi :

$$y = f(net) = \begin{cases} 1 \; jika \; net \geq 0 \\ -1 \; jika \; net < 0 \end{cases}$$

dimana $net = b + \sum_i x_i w_i$

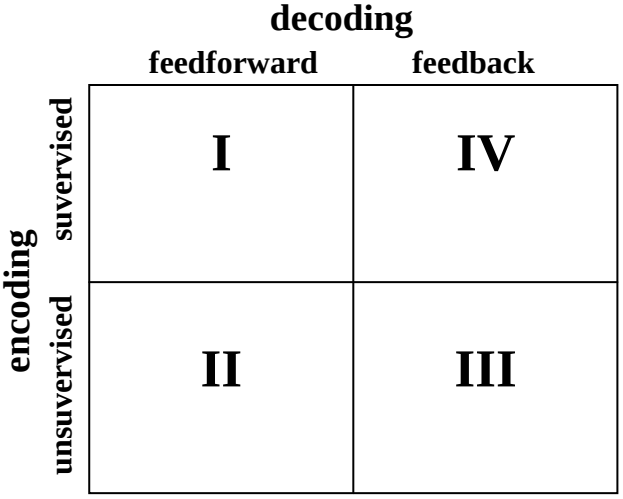
- Kadang kala JST tidak memakai bias tetapi menggunakan threshold (θ) sehingga nilai fungsi aktivasi menjadi :

$$y = f(net) = \begin{cases} 1 \; jika \; net \geq \theta \\ -1 \; jika \; net < \theta \end{cases}$$

dimana $net = \sum_i x_i w_i$

1.6 Prinsip Dasar Pelatihan JST

- Contoh yg baik untuk pelatihan pengenalan sebuah pola adalah pada manusia
- Misal : kita ingin mengenalkan buah jeruk pada seorang bayi dengan cara mendekatkan buah jeruk tsb pada bayi dan mengucapkan kata ”jeruk” secara terus menerus maka secara perlahan bayi tsb akan tahu (kenal) bahwa itu adalah buah jeruk.
- Jadi saat kita mengucapkan kata ”jeruk” maka kekuatan koneksi sinaptik dalam sel syaraf bayi juga akan meningkat dan teraktivasi
- Gambaran diatas menunjukkan bahwa sel syaraf manusia memiliki kemampuan untuk belajar
- Jaringan syaraf tiruan juga memiliki kemampuan untuk belajar hampir sama dgn syaraf pada manusia
- Kemampuan belajar dari JST bersifat terbatas shg jaringan ini juga punya kemampuan terbatas
- Kosko (1990) mengklasifikasikan JST menjadi 2 :
  -bagaimana JST menyimpan pengetahuan /encode (proses belajar)
  -bagaimana JST memproses dan merespon data yg masuk /decode
- Proses encode : 2
  -supervised (dibimbing)
  -unsupervised (tidak dibimbing)
- Proses decode : 2
  -feedforward (tanpa umpan balik)
  -feedback (umpan balik)



Gambar 1.16. Klasifikasi JST

- Kwadran I : supervised – feedforward
  Dalam proses belajar (penyimpanan pengetahuan) dibimbing dan dalam proses merespon data input maka tidak memberikan umpan balik
- Kwadran II : Unsupervised – feedforward
  Dalam proses belajar (penyimpanan pengetahuan) tidak dibimbing dan dalam proses merespon data input maka tidak memberikan umpan balik
- Kwadran III : Unsupervised – feedback
  Dalam proses belajar (penyimpanan pengetahuan) tidak dibimbing dan dalam proses merespon data input maka memberikan umpan balik
- Kwadran IV : supervised – feedback
  Dalam proses belajar (penyimpanan pengetahuan) dibimbing dan dalam proses merespon data input maka memberikan umpan balik
- Dalam JST-supervised, jaringan diberi masukan tertentu dan keluarannya ditentukan oleh pengajarnya. Saat proses belajar ini jaringan akan menyesuaikan bobot sinaptiknya
- Dalam JST-unsupervised, jaringan diberi masukan dan keluarannya akan diatur secara mandiri sesuai aturan yg dimiliki
- Kumpulan dari pasangan vektor input dan vektor output yg diinginkan (target) disebut pola pelatihan.
- Pasangan vektor input dan target dimasukkan bersama dgn nilai bobot koneksi (sinaptik) ke dalam jaringan. Proses ini dilakukan dalam proses belajar/latihan
- Proses belajar :
  -jaringan diberikan masukan / input serta pasangan output yg diinginkan (target)
  -jaringan melakukan perhitungan thd data input dan menghasilkan output sementara
  -membandingkan antara output sementara dgn output yg diinginkan (target) dan selisihnya dipakai untuk memperbaiki nilai bobot sinaptik
  -proses tsb akan diulang sampai kesalahan atau selisih dari output sementara dan target sekecil mungkin atau istilahnya adalan *konvergen*
- Proses belajar akan selesai apabila nilai matrik bobot koneksi (misal : W) yg dihasilkan dapat membuat sistem yg dpt memberikan pola yg sempurna walaupun pola masukan yg diberikan pada jaringan tidak lengkap atau terkena noise
- Ada banyak model JST sesuai dgn konfigurasi neuron dan algoritmanya masing-masing, seperti :
  -JST Hebb
  -JST Hopfield
  -JST Hamming
  -JST Kohonen dll
- Aplikasi JST
  -kedokteran : menyimpan data gejala, diagnosis, serta perawatan /obat

-pengenalan pola : pengenalan karakter tulisan, wajah, suara,sidik jari
-komunikasi : mengurangi suara noise pada komunikasi telepon
-ekonomi : peramalan saham

BAB II Jaringan Syaraf Hebb

- Hebb memperkenalkan aturan pembelajaran/pelatihan pada JST
- Pembelajaran dilakukan dgn memodifikasi kekuatan sinaptik (bobot) dgn cara sedemikan hingga : jika 2 neuron saling berhubungan dalam waktu dan kondisi yg sama (on) maka bobot dari kedua neuron tsb akan dinaikkan
- JST satu lapisan yg dilatih dengan aturan Hebb disebut dgn jaringan Hebb

2.1 Algoritma

langkah 0 : Bobot dan bias diberi nilai awal :

$w_i = 0 \quad (i = 1,2,\ldots,n) \quad ; b = 0$

langkah 1 : untuk tiap pasangan input dan target, s : t, lakukan langkah 2 sampai 4 :

langkah 2 : set aktivasi untuk unit input :

$x_i = s_i \qquad (i = 1,2,\ldots,n)$

langkah 3 : set aktivasi untuk unit output :

$y = t$

langkah 4 : perbaiki nilai bobot :

$w_i(baru) = w_i(lama) + x_i y \qquad (i = 1,2,\ldots,n)$

perbaiki nilai bias :

$b(baru) = b(lama) + y$

- Biasanya perubahan bobot disimbolkan dgn Δw

$\Delta w = x_i y$ sehingga $w_i(baru) = w_i(lama) + \Delta w$

Catatan :

nilai bias dari unit input biasanya secara eksplisit tidak dipakai pada aturan Hebb. Biasanya nilai bias selalu 1. Sebab tanpa nilai bias masalah tidak akan ada solusi.

2.2 Aplikasi

- Contoh aplikasi JST Hebb untuk solusi fungsi logika **AND**

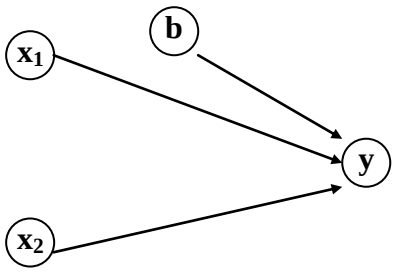
Contoh 2.1: jaringan Hebb untuk fungsi logika AND : input biner dan target biner

Input			target
x_1	x_2	b	t
1	1	1	1
1	0	1	0
0	1	1	0
0	0	1	0

Perubahan bobot : $\Delta w_1 = x_1 y$; $\Delta w_2 = x_2 y$

Perubahan bias : $\Delta b = y$

Hasil selengkapnya dapat dilihat dalam tabel 2.1



Gambar 2.1 Arsitektur JST Hebb untuk solusi fungsi logika AND

Input			target		Perubahan bobot-bias			Bobot dan bias		
x_1	x_2	b	t	y	Δw_1	Δw_2	Δb	w_1	w_2	b
								0	0	0
1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	0	1	1	1
0	1	1	0	0	0	0	0	1	1	1
0	0	1	0	0	0	0	0	1	1	1

Tabel 2.1 Hasil perhitungan JST Hebb untuk solusi fungsi logika AND dengan input biner dan target biner

Catatan :

Hasil akhir dari w_1 dan w_2 dapat juga ditentukan dari penjumlahan seluruh perubahan bobot (Δw) dari input pertama sampai terakhir :

$w_1 = 1 + 0 + 0 + 0 = 1$

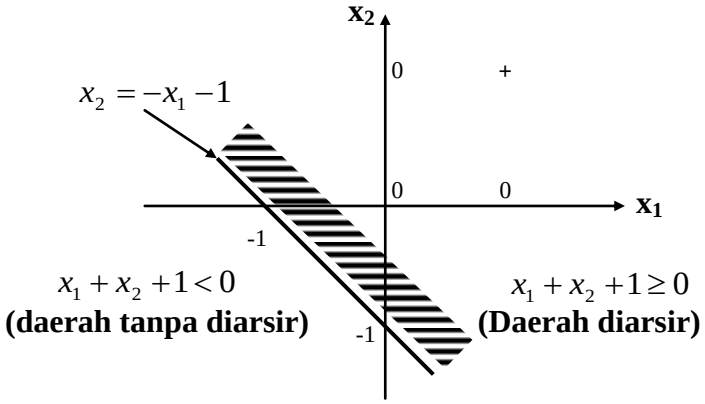
$w_2 = 1 + 0 + 0 + 0 = 1$

- Dari tabel 2.1, batas keputusan (decision boundary) dari fungsi logika AND dinyatakan dengan persamaan garis pemisah (separating line) :

$w_1 x_1 + w_2 x_2 + b = 0$

$1.x_1 + 1.x_2 + 1 = 0$

$x_1 + x_2 + 1 = 0$ atau $x_2 = -x_1 - 1$



Gambar 2.2 Batas keputusan fungsi AND : input dan target biner

- Dapat dilihat dari tabel 2.1, garis batas keputusan dari input pertama sampai input terakhir tidak mengalami perubahan yaitu : $x_2 = -x_1 - 1$ (gamb 2.2)
- Setelah mendapat nilai bobot dan bias maka kita melakukan *Testing* thd pola input dengan memakai bobot dan bias yg didapat dalam proses pelatihan

$w_1 = 1$, $w_2 = 1$, dan $b = 1$

input		Bobot		bias	$net = b + \sum_i x_i w_i$	output	target
x_1	x_2	w_1	w_2	b		y=f(net)	
1	1	1	1	1	3	1	1
1	0	1	1	1	2	1	0
0	1	1	1	1	2	1	0
0	0	1	1	1	1	1	0

Tabel 2.2 Hasil Testing logika AND : input dan target biner

- Tabel 2.2 menunjukkan bahwa hanya input yg pertama menghasilkan output(y) yg sesuai dengan target
- Hasil ini menunjukkan JST Hebb yg dilatih dg pasangan pola input biner dan target biner tidak menghasilkan JST yg sempurna.

Contoh 2.2: Jaringan Hebb untuk fungsi AND : input biner dan target *bipolar*

Input			target
x_1	x_2	b	t
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

Perubahan bobot : $\Delta w_1 = x_1 y$; $\Delta w_2 = x_2 y$
Perubahan bias : $\Delta b = y$

Hasil selengkapnya dapat dilihat dalam tabel 2.3

Input			Target		Perubahan bobot-bias			Bobot dan bias		
x_1	x_2	b	t	y	Δw_1	Δw_2	Δb	w_1	w_2	b
								0	0	0
1	1	1	1	1	1	1	1	1	1	1
1	0	1	-1	-1	-1	0	-1	0	1	0
0	1	1	-1	-1	0	-1	-1	0	0	-1
0	0	1	-1	-1	0	0	-1	0	0	-2

Tabel 2.3 Hasil perhitungan JST Hebb untuk solusi fungsi logika AND dengan input biner dan target *bipolar*

- Setelah mendapat nilai bobot dan bias maka kita melakukan *Testing* thd pola input dengan memakai bobot dan bias yg didapat dalam proses pelatihan

$w_1 = 0$, $w_2 = 0$, dan $b = -2$

input		Bobot		bias	$net = b + \sum_i x_i w_i$	output	target
x_1	x_2	w_1	w_2	b		y=f(net)	
1	1	0	0	-2	-2	-1	1
1	0	0	0	-2	-2	-1	-1
0	1	0	0	-2	-2	-1	-1
0	0	0	0	-2	-2	-1	-1

Tabel 2.4 Hasil Testing logika AND : input biner dan target *bipolar*

- Tabel 2.4 menunjukkan bahwa input yg pertama menghasilkan output(y) tidak sesuai dengan target, sedang input kedua, ketiga, dan keempat sesuai dg target
- Hasil ini menunjukkan JST Hebb yg dilatih dg pasangan pola input biner dan target bipolar tidak menghasilkan JST yg sempurna

Contoh 2.3: Jaringan Hebb untuk fungsi AND : input *bipolar* dan target *bipolar*

Input				target
x_1	x_2	b		t
1	1	1		1
1	-1	1		-1
-1	1	1		-1
-1	-1	1		-1

Perubahan bobot : $\Delta w_1 = x_1 y$; $\Delta w_2 = x_2 y$
Perubahan bias : $\Delta b = y$

Hasil selengkapnya dapat dilihat dalam tabel 2.4

Input			Target		Perubahan bobot-bias			Bobot dan bias		
x_1	x_2	b	t	y	Δw_1	Δw_2	Δb	w_1	w_2	b
								0	0	0
1	1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	-1	1	-1	0	2	0
-1	1	1	-1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	-1	1	1	-1	2	2	-2

Tabel 2.4 Hasil perhitungan JST Hebb untuk solusi fungsi logika AND dengan input *bipolar* dan target *bipolar*

- Hasil *Testing* thd pola input dengan memakai bobot dan bias yg didapat dalam proses pelatihan

  $w_1 = 2$, $w_2 = 2$, dan $b = -2$

Input		Bobot		bias	$net = b + \sum_i x_i w_i$	output	Target
x_1	x_2	w_1	w_2	b		y=f(net)	
1	1	2	2	-2	2	1	1
1	-1	2	2	-2	-2	-1	-1
-1	1	2	2	-2	-2	-1	-1
-1	-1	2	2	-2	-6	-1	-1

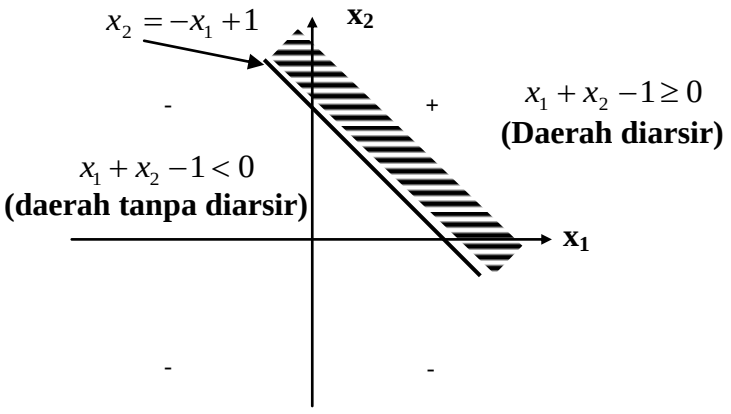
Tabel 2.5 Hasil Testing logika AND : input bipolar dan target *bipolar*

- Tabel 2.5 menunjukkan bahwa semua input menghasilkan output(y) sesuai dengan target
- Hasil ini menunjukkan JST Hebb yg dilatih dg pasangan pola input bipolar dan target bipolar menghasilkan JST yg sempurna
- Dari proses pelatihan (tabel 2.4) maka batas keputusan (decision boundary) dari fungsi logika AND yg dilatih dgn pola input bipolar dan target bipolar dinyatakan dengan persamaan garis pemisah (separating line) :

  $w_1 x_1 + w_2 x_2 + b = 0$

  $2.x_1 + 2.x_2 + (-2) = 0$

  $2x_1 + 2x_2 - 2 = 0$ atau $x_2 = -x_1 + 1$



Gambar 2.2 Batas keputusan fungsi AND : input dan target bipolar

- Dari contoh 2.1 dimana digunakan data biner (input biner dan target biner) maka *tidak terjadi perbaikan bobot* untuk pasangan pola pelatihan untuk unit input ”on” (x_i = 1) dan target ”off ” (t = 0) maupun untuk unit input dan target keduanya dalam keadaan ”off ” (x_i = 0 dan t = 0) :

  $\Delta w = x_i y = 0$

  sehingga

  $w_i(\text{baru}) = w_i(\text{lama}) + \Delta w$

  $w_i(\text{baru}) = w_i(\text{lama})$

- Sedangkan dalam contoh 2.3 digunakan data bipolar (input dan target bipolar) maka *terjadi perbaikan bobot* untuk semua pasangan pola pelatihan baik unit input ”on” (x_i = 1) dan target ”off ” (t = -1)

Contoh 2.4: Jaringan Hebb untuk mengklasifikasi pola input 2 dimensi dari 2 buah huruf yaitu : ”X” dan ”O”. Pola dari huruf tsb dinyatakan dgn :

```
#...#        .###.
.#.#.        #...#
..#..        #...#
.#.#.        #...#
#...#        .###.
```
Pola 1 Pola 2

jawab :

Dalam hal ini kita menganggap jaringan hanya mempunyai 1 output yaitu kelas X(untuk huruf ”X”) dan kelas bukan X (untuk huruf ”O”). Misal kelas X kita beri nilai target 1 sedangkan kelas bukan X kita beri target –1. Sedangkan setiap lambang ”#” kita beri nilai 1 dan lambang ”.” kita beri nilai –1. Vektor input untuk pola 1 dan pola 2 menjadi :

Pola	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
”X”	1	-1	-1	-1	1	-1	1	-1	1	-1	-1	-1	1	-1	-1
”O”	-1	1	1	1	-1	1	-1	-1	-1	1	1	-1	-1	-1	1

Pola	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	X_{22}	x_{23}	x_{24}	X_{25}	Target
”X”	-1	1	-1	1	-1	1	-1	-1	-1	1	1
”O”	1	-1	-1	-1	1	-1	1	1	1	-1	-1

Tabel 2.6 Vektor input dari kedua pola (”X” dan ”O”)

Bobot mula-mula :

$W_i = 0$ dimana i = 1,2,…,25

Sedangkan perubahan bobot (Δw_i) dan bias setelah diberikan input pola 1 dan 2 :

	Δw_1	Δw_2	Δw_3	Δw_4	Δw_5	Δw_6	Δw_7	Δw_8	Δw_9	Δw_{10}	Δw_{11}	Δw_{12}	Δw_{13}	Δw_{14}	Δw_{15}
”X”	1	-1	-1	-1	1	-1	1	-1	1	-1	-1	-1	1	-1	-1
”O”	1	-1	-1	-1	1	-1	1	1	1	-1	-1	1	1	1	-1

	Δw_{16}	Δw_{17}	Δw_{18}	Δw_{19}	Δw_{20}	Δw_{21}	Δw_{22}	Δw_{23}	Δw_{24}	Δw_{25}	b
”X”	-1	1	-1	1	-1	1	-1	-1	-1	1	1
”O”	-1	1	1	1	-1	1	-1	-1	-1	1	-1

Tabel 2.7 Perubahan bobot Δw_i dan bias b

Dan bobot akhir (w_i) dan bias b dapat ditentukan dari penjumlahan kedua perubahan bobot diatas sehingga :

w_1	w_2	w_3	w_4	w_5	w_6	w_7	W_8	w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}
2	-2	-2	-2	2	-2	2	0	2	-2	-2	0	2	0	-2

w_{16}	w_{17}	w_{18}	w_{19}	w_{20}	w_{21}	w_{22}	w_{23}	w_{24}	w_{25}	b
-2	2	0	2	-2	2	-2	-2	-2	2	0

Tabel 2.8 Nilai bobot w_i dan bias b akhir

Setelah mendapatkan bobot akhir (w_i) dan bias b, selanjutnya dapat dilakukan proses testing terhadap pola input. Pertama kita melakukan testing thd pola 1 (huruf ”X”) :

$$\sum x_i w_i + b = 42 + 0 = 42$$

Selanjutnya kita melakukan testing thd pola 2 (huruf ”O”) :

$$\sum x_i w_i + b = -42 + 0 = -42$$

Hasil testing selengkapnya dapat dilihat dalam tabel 2.9 :

Input			Bobot			Bias	$net = b + \sum_i x_i w_i$	Output	Target
x_1	…	x_{25}	w_1	…	w_{25}	b		y=f(net)	
1		-1	2		2	0	42	1	1
-1		-1	2		2	0	-42	-1	-1

Tabel 2.9 Hasil testing

Dari tabel 2.9 dapat dilihat hasil testing terhadap pola 1 (”X”) dan pola 2(”O”) menghasilkan output(y) yang sesuai dengan target.

BAB III PERCEPTRON

- Perceptron merupakan jaringan syaraf yg memiliki pengaruh yang paling luas dibandingkan model jaringan syaraf sebelumnya.
- Metode pembelajaran Perceptron lebih kuat dari pembelajaran JST Hebb.
- Prosedur pelatihan dari Perceptron terutama dalam proses iterasinya dapat menghasilkan bobot yang *konvergen.*Dan bobot ini bila ditest terhadap pola input dapat menghasilkan output yang sesuai target.
- Ada beberapa tipe Perceptron yang dikemukan oleh beberapa ahli seperti : Rosenblatt (1962), Minsky dan Papert (1969 dan 1988).
- Mulanya Perceptron punya 3 lapisan neuron : *unit sensor, unit asosiasi, dan unit renspon(output).* Mendekati model jaringan syaraf pada retina
- Unit sensor dan asosiasi memakai aktivasi biner sedangkan unit output memakai aktivasi 1, 0, atau –1.
- Fungsi aktivasi untuk unit asosiasi adalah fungsi step biner dgn nilai threshold θ tertentu.
- Sinyal yg dikirim dari unit asosiasi ke unit output adalah sinyal biner (1 atau 0).
- Output perceptron y = f(y_in), dimana fungsi aktivasinya adalah :

$$f(y\_in) = \begin{cases} 1 & jika \quad y\_in > \theta \\ 0 & jika \quad -\theta \le y\_in \le \theta \\ -1 & jika \quad y\_in < -\theta \end{cases} \qquad \text{.....(3.1)}$$

- Bobot dari unit asosiasi ke unit respon (output) diatur dengan aturan pembelajaran perceptron. Untuk tiap data input pelatihan jaringan akan menghitung respon dari unit output. Jaringan akan menentukan apakah terjadi error pada data input ini (dengan membandingkan antara nilai *output* yg dihitung dengan nilai *target*).
- Jaringan tidak akan menganggap ada kesalahan (error) jika outputnya bernilai 0 dan target bernilai –1. Sedangkan jika output bernilai 1 dan target bernilai –1 maka ini dianggap sebagai kesalahan (error).
- Jika terjadi kesalahan maka bobot akan diubah menurut persamaan :

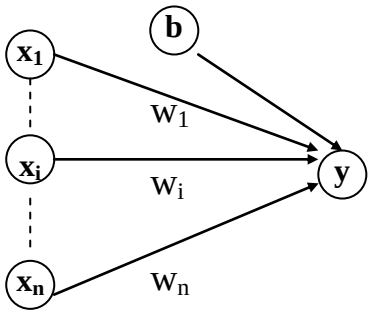
$$w_i(baru) = w_i(lama) + \alpha t x_i \qquad \text{.....(3.2)}$$

dimana nilai target t adalah 1 atau –1, α adalah learning rate.

- Pelatihan akan diteruskan sampai tidak terjadi kesalahan.

3.1 Arsitektur perceptron

- Arsitektur sederhana perceptron adalah seperti gambar 3.1.
- Jaringan ini dipakai untuk mengklasifikasi satu kelas tunggal. Yaitu menentukan apakah suatu pola input termasuk dalam satu kelas atau tidak. Jika masuk dalam satu kelas maka respon dari unit output adalah 1 sedangkan jika tidak maka responnya –1.



Gambar 3.1 Arsitektur perceptron untuk klasifikasi tunggal

3.1 Algoritma perceptron

- Algoritma ini berlaku untuk input biner atau bipolar dan target bipolar, serta nilai threshold θ, dan bias b.
- Langkah-langkah algoritma perceptron :

Langkah 0 : bobot dan bias diberi nilai awal
(biasanya bobot dan bias diberi nilai 0)
beri nilai learning rate $(0 < \alpha \leq 1)$
(biasanya α diberi nilai 1)

langkah 1 : selama kondisi berhenti bernilai salah lakukan langkah 2 – 6

langkah 2 : untuk tiap-tiap pasangan pola pelatihan s : t, lakukan langkah 3-5

langkah 3 : set aktivasi unit input :

$x_i = s_i$

langkah 4 : hitung respon dari unit output :

$$y\_in = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 & jika \;\; y\_in > \theta \\ 0 & jika \;\; -\theta \leq y\_in \leq \theta \\ -1 & jika \;\; y\_in < -\theta \end{cases}$$

langkah 5 : perbaiki bobot dan bias jika terjadi kesalahan pada pola ini :
jika $y \neq t$

$$w_i(baru) = w_i(lama) + \alpha t x_i$$

$$b(baru) = b(lama) + \alpha t$$

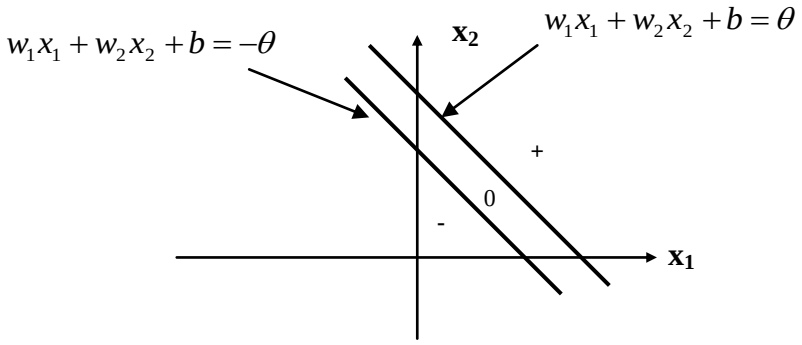
jika $y = t$

$$w_i(baru) = w_i(lama)$$

$$b(baru) = b(lama)$$

langkah 6 : test kondisi berhenti :
jika tidak ada bobot yang berubah dalam langkah 2 maka stop
jika ada maka lanjutkan kembali ke langkah 1

- Dari algoritma maka hanya bobot yang berkoneksi dgn unit input yang aktif yg akan berubah ($x_i \neq 0$)
- Algoritma ini biasanya dipakai untuk melakukan klasifikasi pola dengan cara pemisahan linear.
- Fungsi aktivasi yg dipakai akan mengatur batas antara daerah positif, negatif dan nol.
- Daerah nol adalah daerah yg terletak antara daerah positif dan negatif. Lebar daerah ini adalah θ
- Garis pemisah antara daerah positif dgn daerah nol dinyatakan dgn persamaan :
  $w_1x_1 + w_2x_2 + b = \theta$
- Garis pemisah antara daerah negatif dgn daerah nol dinyatakan dgn persamaan :
  $w_1x_1 + w_2x_2 + b = -\theta$



Gambar 3.2 Batas keputusan perceptron

- Shg daerah positif dinyatakan dgn pertidaksamaan :
  $w_1x_1 + w_2x_2 + b > \theta$
- Dan daerah negatif dinyatakan dgn pertidaksamaan :
  $w_1x_1 + w_2x_2 + b < -\theta$

3.2 Aplikasi perceptron

- Contoh 3.1 :
  Aplikasi perceptron untuk menyelesaikan fungsi logika AND dengan input biner dan target bipolar

Input			target
x_1	x_2	b	t
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

Nilai $\alpha = 0.7$
Bobot (w) dan bias (b) diberi nilai 0
Nilai $\theta = 0.3$

Pelatihan dgn algoritma perceptron :

Iterasi pertama :

Data pertama : (1 1)

$$y\_in = b + \sum_i x_i w_i = b + x_1 w_1 + x_2 w_2$$

= 0 + (1).(0) + (1).(0) = 0

hasil aktivasi = y = 0 (-0.3 <y_in<0.3)

target t = 1

y ≠ t

bobot baru :

w_i(baru) = w_i(lama) + αtx_i

w_1 = 0+(0.7)(1)(1) = 0.7

w_2 = 0+(0.7)(1)(1) = 0.7

bias baru :

b(baru) = b(lama) + αt

b = 0 + (0.7)(1) = 0.7

Data kedua : (1 0)

y_in = 0.7 + 0.7 + 0 = 1.4

hasil aktivasi = y = 1 (y_in > 0.3)

target t = -1

y ≠ t

bobot baru :

w_1 = 0.7+(0.7)(-1)(1) = 0

w_2 = 0.7+(0.7)(-1)(0) = 0.7

bias baru :

b = 0.7 + (0.7)(-1) = 0

Data ketiga : (0 1)

y_in = 0 + 0 + 0.7 = 0.7

hasil aktivasi = y = 1 (y_in > 0.3)

target t = -1

y ≠ t

bobot baru :

w_1 = 0+(0.7)(-1)(0) = 0

w_2 = 0.7+(0.7)(-1)(1) = 0

bias baru :

b = 0 + (0.7)(-1) = -0.7

Data keempat : (0 0)

y_in = -0.7 + 0 + 0 = -0.7

hasil aktivasi = y = -1 (y_in < -0.3)

target t = -1

y = t

Iterasi kedua :

Data pertama : (1 1)

y_in = -0.7 + 0 + 0 = -0.7

hasil aktivasi = y = -1 (y_in <- 0.3)

target t = 1

y ≠ t

bobot baru :

w_1 = 0+(0.7)(1)(1) = 0.7

w_2 = 0+(0.7)(1)(1) = 0.7

bias baru :

b = -0.7 + (0.7)(1) = 0

Data kedua : (1 0)

y_in = 0 + 0.7 + 0 = 0.7

hasil aktivasi = y = 1 (y_in > 0.3)

target t = -1

y ≠ t

bobot baru :

w_1 = 0.7+(0.7)(-1)(1) = 0

w_2 = 0.7+(0.7)(-1)(0) = 0.7

bias baru :

b = 0 + (0.7)(-1) = -0.7

Data ketiga : (0 1)

y_in = -0.7 + 0 + 0.7 = 0

hasil aktivasi = y = 0 (-0.3<y_in < 0.3)

target t = -1

y ≠ t

bobot baru :

w_1 = 0+(0.7)(-1)(0) = 0

w_2 = 0.7+(0.7)(-1)(1) = 0

bias baru :

b = -0.7 + (0.7)(-1) = -1.4

Data keempat : (0 0)

y_in = -1.4 + 0 + 0 = -1.4

hasil aktivasi = y = -1 (y_in < -0.3)

target t = -1

y = t

Iterasi ketiga :

Data pertama : (1 1)
y_in = -1.4 + 0 + 0 = -1.4
hasil aktivasi = y = -1 (y_in <- 0.3)
target t = 1
$y \neq t$
bobot baru :
$w_1 = 0+(0.7)(1)(1) = 0.7$
$w_2 = 0+(0.7)(1)(1) = 0.7$
bias baru :
b = -1.4 + (0.7)(1) = -0.7

Data kedua : (1 0)
y_in = -0.7 + 0.7 + 0 = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3)
target t = -1
$y \neq t$
bobot baru :
$w_1 = 0.7+(0.7)(-1)(1) = 0$
$w_2 = 0.7+(0.7)(-1)(0) = 0.7$
bias baru :
b = -0.7 + (0.7)(-1) = -1.4

Data ketiga : (0 1)
y_in = -1.4 + 0 + 0.7 = -0.7
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Data keempat : (0 0)
y_in = -1.4 + 0 + 0 = -1.4
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Iterasi keempat :

Data pertama : (1 1)
y_in = -1.4 + 0 + 0.7 = -0.7
hasil aktivasi = y = -1 (y_in <- 0.3)
target t = 1
$y \neq t$
bobot baru :
$w_1 = 0+(0.7)(1)(1) = 0.7$
$w_2 = 0.7+(0.7)(1)(1) = 1.4$
bias baru :
b = -1.4 + (0.7)(1) = -0.7

Data kedua : (1 0)
y_in = -0.7 + 0.7 + 0 = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3)
target t = -1
$y \neq t$
bobot baru :
$w_1 = 0.7+(0.7)(-1)(1) = 0$
$w_2 = 1.4+(0.7)(-1)(0) = 1.4$
bias baru :
b = -0.7 + (0.7)(-1) = -1.4

Data ketiga : (0 1)
y_in = -1.4 + 0 + 1.4 = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3)
target t = -1
$y \neq t$
bobot baru :
$w_1 = 0+(0.7)(-1)(0) = 0$
$w_2 = 1.4+(0.7)(-1)(1) = 0.7$
bias baru :
b = -1.4 + (0.7)(-1) = -2.1

Data keempat : (0 0)
y_in = -2.1 + 0 + 0 = -2.1
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Iterasi kelima :

Data pertama : (1 1)
y_in = -2.1 + 0 + 0.7 = -1.4
hasil aktivasi = y = -1 (y_in <- 0.3)
target t = 1
$y \neq t$
bobot baru :
$w_1 = 0+(0.7)(1)(1) = 0.7$
$w_2 = 0.7+(0.7)(1)(1) = 1.4$
bias baru :
b = -2.1 + (0.7)(1) = -1.4

Data kedua : (1 0)
y_in = -1.4 + 0.7 + 0 = -0.7
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Data ketiga : (0 1)
y_in = -1.4 + 0 + 1.4 = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3)
target t = -1
$y \neq t$
bobot baru :
$w_1 = 0.7+(0.7)(-1)(0) = 0.7$
$w_2 = 1.4+(0.7)(-1)(1) = 0.7$
bias baru :
b = -1.4 + (0.7)(-1) = -2.1

Data keempat : (0 0)
y_in = -2.1 + 0 + 0 = -2.1
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Iterasi keenam :

Data pertama : (1 1)
y_in = -2.1 + 0.7 + 0.7 = -0.7
hasil aktivasi = y = -1 (y_in <- 0.3)
target t = 1
$y \neq t$
bobot baru :
$w_1 = 0.7+(0.7)(1)(1) = 1.4$
$w_2 = 0.7+(0.7)(1)(1) = 1.4$
bias baru :
b = -2.1 + (0.7)(1) = -1.4

Data kedua : (1 0)
y_in = -1.4 + 1.4 + 0 = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3)
target t = -1
$y \neq t$
bobot baru :
$w_1 = 0.7+(0.7)(-1)(0) = 0.7$
$w_2 = 1.4+(0.7)(-1)(0) = 1.4$
bias baru :
b = -1.4 + (0.7)(-1) = -2.1

Data ketiga : (0 1)
y_in = -2.1 + 0 + 1.4 = -0.7
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Data keempat : (0 0)
y_in = -2.1 + 0 + 0 = -2.1
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

y = t

Iterasi ketujuh :

Data pertama : (1 1)
y_in = -2.1 + 0.7 + 1.4 = 0
hasil aktivasi = y = 0 (-0.3<y_in< 0.3)
target t = 1
y ≠ t
bobot baru :
w_1 = 0.7+(0.7)(1)(1) = 1.4
w_2 = 1.4+(0.7)(1)(1) = 2.1
bias baru :
b = -2.1 + (0.7)(1) = -1.4

Data kedua : (1 0)
y_in = -1.4 + 1.4 + 0 = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3)
target t = -1
y ≠ t
bobot baru :
w_1 = 1.4+(0.7)(-1)(0) = 0.7
w_2 = 2.1+(0.7)(-1)(0) = 2.1
bias baru :
b = -1.4 + (0.7)(-1) = -2.1

Data ketiga : (0 1)
y_in = -2.1 + 0 + 2.1 = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3)
target t = -1
y ≠ t
bobot baru :
w_1 = 0.7+(0.7)(-1)(0) = 0.7
w_2 = 2.1+(0.7)(-1)(1) = -1.4
bias baru :
b = -2.1 + (0.7)(-1) = -2.8

Data keempat : (0 0)
y_in = -2.8 + 0 + 0 = -2.8
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1

Iterasi kedelapan :

Data pertama : (1 1)
y_in = -2.8 + 0.7 + 1.4 = -0.7
hasil aktivasi = y = -1 (y_in <- 0.3)
target t = 1
y ≠ t
bobot baru :
w_1 = 0.7+(0.7)(1)(1) = 1.4
w_2 = 1.4+(0.7)(1)(1) = 2.1
bias baru :
b = -2.8 + (0.7)(1) = -2.1

Data kedua : (1 0)
y_in = -2.1 + 1.4 + 0 = -0.7
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Data ketiga : (0 1)
y_in = -2.1 + 0 + 2.1 = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3)
target t = -1
y ≠ t
bobot baru :
w_1 = 1.4+(0.7)(-1)(0) = 1.4
w_2 = 2.1+(0.7)(-1)(1) = 1.4
bias baru :
b = -2.1 + (0.7)(-1) = -2.8

Data keempat : (0 0)
y_in = -2.8 + 0 + 0 = -2.8
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Iterasi kesembilan :

Data pertama : (1 1)
y_in = -2.8 + 1.4 + 1.4 = 0
hasil aktivasi = y = 0 (-0.3<y_in< 0.3)
target t = 1
y ≠ t
bobot baru :
w_1 = 1.4+(0.7)(1)(1) = 2.1
w_2 = 1.4+(0.7)(1)(1) = 2.1
bias baru :
b = -2.8 + (0.7)(1) = -2.1

Data kedua : (1 0)
y_in = -2.1 + 2.1 + 0 = 0
hasil aktivasi = y = 0 (-0.3<y_in < 0.3)
target t = -1
y ≠ t
bobot baru :
w_1 = 2.1+(0.7)(-1)(1) = 1.4
w_2 = 2.1+(0.7)(-1)(0) = 2.1
bias baru :
b = -2.1 + (0.7)(-1) = -2.8

Data ketiga : (0 1)
y_in = -2.8 + 0 + 2.1 = -0.7
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Data keempat : (0 0)
y_in = -2.8 + 0 + 0 = -2.8
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Iterasi kesepuluh :

Data pertama : (1 1)
y_in = -2.8 + 1.4 + 2.1 = 0.7
hasil aktivasi = y = 1 (y_in > 0.3)
target t = 1
y = t

Data kedua : (1 0)
y_in = -2.8 + 1.4 + 0 = -1.4
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

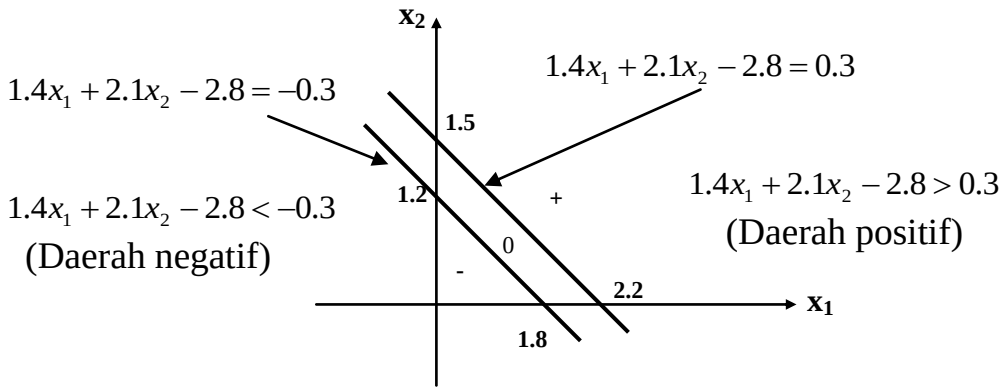
Data ketiga : (0 1)
y_in = -2.8 + 0 + 2.1 = -0.7
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

Data keempat : (0 0)
y_in = -2.8 + 0 + 0 = -2.8
hasil aktivasi = y = -1 (y_in < -0.3)
target t = -1
y = t

- Pada iterasi kesepuluh tidak terjadi perubahan bobot shg proses pelatihan dapat dihentikan. Hasil akhir dari proses pembelajaran ini adalah :
  $w_1 = 1.4$; $w_2 = 2.1$; b = -2.8

- Shg persamaan garis pemisah antara daerah positif dan nol adalah :
  $1.4x_1 + 2.1x_2 – 2.8 = 0.3$

- Shg persamaan garis pemisah antara daerah negatif dan nol adalah :
  $1.4x_1 + 2.1x_2 – 2.8 = -0.3$



Gambar 3.3 Garis batas keputusan Perceptron : logika AND dgn input biner targer bipolar

- Hasil *Testing* thd pola input dengan memakai bobot dan bias yg didapat dalam proses pelatihan :

  $w_1 = 1.4$, $w_2 = 2.1$, dan $b = -2.8$

Input		Bobot		Bias	$net = b + \sum_i x_i w_i$	output	target
x_1	x_2	w_1	w_2	b		y=f(net)	
1	1	1.4	2.1	-2.8	0.7	1	1
1	0	1.4	2.1	-2.8	-1.4	-1	-1
0	1	1.4	2.1	-2.8	-0.7	-1	-1
0	0	1.4	2.1	-2.8	-2.8	-1	-1

Tabel 3.1 Hasil Testing logika AND : input biner dan target *bipolar*

- Soal latihan :

  Tentukan solusi fungsi logika AND dengan input biner dan target bipolar yang dilatih dengan algoritma perceptron.

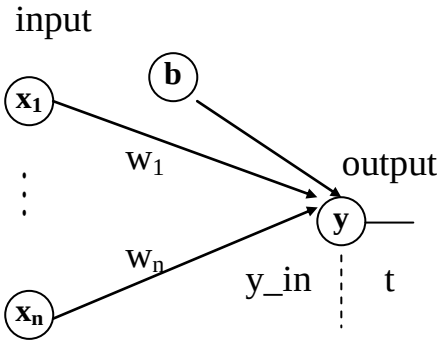
Input			target
x_1	x_2	b	t
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

Dimana :
Nilai $\alpha = 1$ dan nilai $\theta = 0.2$
Hitung bobot dan bias akhir. Selanjutnya gambarkan garis batas keputusannya.

BAB IV Delta rule, Adaline, dan Madaline

4.1 Delta rule

- Metode pembelajaran Delta rule biasanya dipakai dalam pelatihan pada model jaringan Adaline dan Madaline



Gambar 4.1 Arsitektur Adaline dgn satu unit output

- Tujuan utama dari delta rule adalah untuk memperbaiki bobot-bobot (w) yg menghubungkan antara unit input dan output sehingga selisih antara input jaringan untuk unit output (y_in) dan nilai target (t) menjadi minimal.
- Delta rule untuk adaline dengan satu unit output dirumuskan :

$$\Delta w_i = \alpha(t - y\_in)x_i \quad \text{.....(4.1)}$$

dimana, $(t - y\_in)$ → selisih

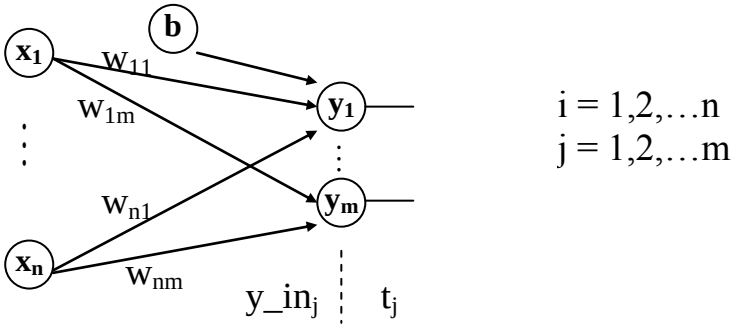
$$y\_in = \sum_{i=1}^{n} x_i w_i$$

- Sedangkan Delta rule untuk adaline dgn beberapa unit output dirumuskan :

$$\Delta w_{ij} = \alpha(t_j - y\_in_j)x_i \quad \text{.....(4.2)}$$

dimana,

$$y\_in_j = \sum_{i=1}^{n} x_i w_{ij}$$

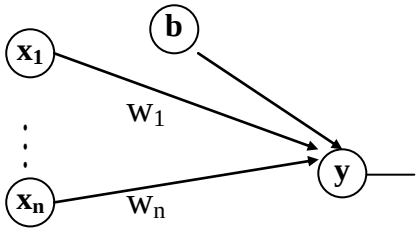


Gambar 4.2 Arsitektur Adaline dgn beberapa unit output

4.2 ADALINE (adaptive linear neuron)

- Diperkenalkan oleh Widrow dan Hoff (1960)

- Jaringan adaline biasanya dilatih dengan metode Delta rule atau sering disebut metode *Least Mean Square* (LMS)
- Adaline adalah unit tunggal (neuron) yg menerima input dari beberapa unit.
- Arsitektur dari adaline



Gambar 4.3 Arsitektur sebuah Adaline

- Beberapa adaline yg menerima sinyal dari unit input yg sama dapat dipadukan dalam jaringan satu lapisan seperti dalam jaringan Perceptron.
- Jika adaline dikombinasikan sedemikian hingga output dari beberapa adaline menjadi input bagi adaline yg lain. Jaringan ini membentuk jaringan banyak lapisan yg sering disebut dgn *Madaline* (many adaline).
- Algoritma Adaline :

Langkah 0 : inisialisasi bobot (biasanya nilai acak yg kecil)
Set nilai learning rate α

Langkah 1 : selama kondisi berhenti bernilai salah, lakukan langkah 2 – 6

Langkah 2 : untuk setiap pasangan pola pelatihan bipolar s : t kerjakan langkah 3 – 5

Langkah 3 : set aktivasi unit input , i = 1,2,...,n

$x_i = s_i$

Langkah 4 : tentukan input jaringan pada unit output :

$$y\_in = b + \sum_i x_i w_i$$

Langkah 5 : perbaiki nilai bias dan bobot :

w(baru) = w(lama) + α(t – y_in)x_i

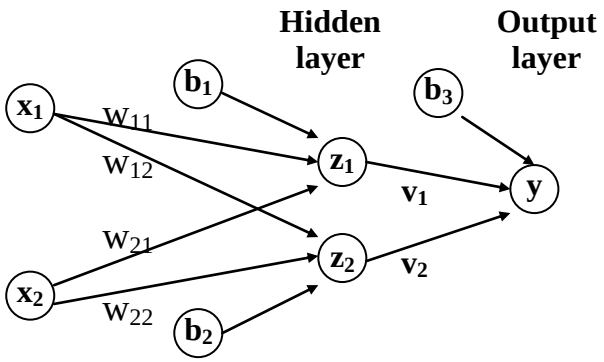
b(baru) = b(lama) + α(t – y_in)

Langkah 6 : tes kondisi berhenti. Jika perubahan bobot tertinggi yg terjadi pada langkah 2 nilainya lebih kecil dari toleransi tertentu maka hentikan jika tidak lanjutkan.

4.3 MADALINE

- Madaline adalah kumpulan dari banyak adaline yg membentuk jaringan banyak lapisan atau jaringan satu lapisan dgn beberapa output.
- Contoh yg diberikan pada Perceptron dan delta rule untuk beberapa output menunjukkan tidak ada perubahan mendasar pada proses pelatihan jika beberapa unit adaline dikombinasikan dalam jaringan satu lapisan.

- Salah satu contoh madaline : jaringan 2 lapisan yg terdiri dari 1 lapisan tersembunyi (terdiri dari 2 unit adaline) dan 1 lapisan output (terdiri dari 1 unit adaline).Lihat gambar 4.4
-



Gambar 4.4 Arsitektur madaline : hidden layer (2 unit adaline) dan output layer (1 unit adaline)

- ada 2 buah algoritma pelatihan Madaline yaitu : MR I dan MR II

4.3.1 Algoritma MR I

- Dalam algoritma MR I : bobot v_1 dan v_2 dan bias b_3 ditentukan sedemikian sehingga output y akan bernilai 1 jika salah satu (atau keduanya) z_1 atau z_2 bernilai 1. Sedangkan y akan bernilai –1 jika z_1 dan z_2 keduanya bernilai –1.Jadi unit y melakukan fungsi logika OR pada sinyal z_1 dan z_2.
- Bobot dan bias pada unit y ditentukan yaitu :

  $v_1 = ½$
  $v_2 = ½$
  $b_3 = ½$

- Fungsi aktivasi untuk z_1,z_2, dan y adalah :

$$f(x) = \begin{cases} 1 \; jika \; x \geq 0 \\ -1 \; jika \; x < 0 \end{cases}$$

- Algoritma MR I terdiri dari langkah-langkah yaitu :

Langkah 0 : inisialisasi bobot
bobot v_1 , v_2 dan bias b_3 ditentukan dgn nilai spt yg telah ditentukan diatas,sedangkan bobot dan bias yg lain di set dgn nilai acak kecil.
set learning rate α dgn nilai kecil

langkah 1 : selama kondisi berhenti bernilai salah, kerjakan langkah 2 – 8 :
langkah 2 : untuk setiap pola pelatihan bipolar s : t, kerjakan langkah 3 – 7 :
langkah 3 : set aktivasi dari unit input :
$x_i = s_i$
langkah 4 : hitung input jaringan untuk tiap-tiap hidden unit :
$z\_in_1 = b_1 + x_1w_{11} + x_2w_{21}$
$z\_in_2 = b_2 + x_1w_{12} + x_2w_{22}$
langkah 5 : tentukan output untuk tiap-tiap hidden unit :
$z_1 = f(z\_in_1)$
$z_2 = f(z\_in_2)$
langkah 6 : tentukan output jaringan :

$y\_in = b_3 + z_1v_1 + z_2v_2$

$y = f(y\_in)$

langkah 7 : tentukan kesalahan dan perbaiki bobot :

jika t = y, maka tidak ada bobot yang diperbaiki

jika $t \neq y$:

jika t =1, maka perbaiki bobot pada z_j, unit input yg paling dekat dgn 0

$b_j(baru) = b_j(lama) + \alpha(1-z\_in_j)$

$w_{ij}(baru) = w_{ij}(lama) + \alpha(1-z\_in_j)x_i$

jika t =-1, maka perbaiki bobot pada semua unit z_k, input jaringan yg bernilai positif

$b_k(baru) = b_k(lama) + \alpha(-1-z\_in_k)$

$w_{ik}(baru) = w_{ik}(lama) + \alpha(-1-z\_in_k)x_i$

langkah 8 : tes kondisi berhenti

jika sudah tidak terjadi perubahan bobot atau jika nomer maksimum dari iterasi perubahan bobot sudah dikerjakan semuanya(langkah 2) maka hentikan. Jika tidak lanjutkan.

4.3.2 Algoritma MR II

- Algoritma MR II hampir sama dgn MR I, bedanya terletak pada langkah ke-7.
- Algoritma MR II : bobot v_1 dan v_2 dan bias b_3 ditentukan sedemikian sehingga output y akan bernilai 1 jika salah satu (atau keduanya) z_1 atau z_2 bernilai 1. Sedangkan y akan bernilai –1 jika z_1 dan z_2 keduanya bernilai –1.Jadi unit y melakukan fungsi logika OR pada sinyal z_1 dan z_2.
- Bobot dan bias pada unit y ditentukan yaitu :

$v_1 = ½$

$v_2 = ½$

$b_3 = ½$

- Fungsi aktivasi untuk z_1,z_2, dan y adalah :

$$f(x) = \begin{cases} 1 \; jika \; x \geq 0 \\ -1 \; jika \; x < 0 \end{cases}$$

- Algoritma MR II terdiri dari langkah-langkah yaitu :

Langkah 0 : inisialisasi bobot

bobot v_1 , v_2 dan bias b_3 ditentukan dgn nilai spt yg telah ditentukan diatas,sedangkan bobot dan bias yg lain di set dgn nilai acak kecil.

set learning rate α dgn nilai kecil

langkah 1 : selama kondisi berhenti bernilai salah, kerjakan langkah 2 – 8 :

langkah 2 : untuk setiap pola pelatihan bipolar s : t, kerjakan langkah 3 – 7 :

langkah 3 : set aktivasi dari unit input :

$x_i = s_i$

langkah 4 : hitung input jaringan untuk tiap-tiap hidden unit :

$z\_in_1 = b_1 + x_1w_{11} + x_2w_{21}$

$z\_in_2 = b_2 + x_1w_{12} + x_2w_{22}$

langkah 5 : tentukan output untuk tiap-tiap hidden unit :

$z_1 = f(z\_in_1)$

$z_2 = f(z\_in_2)$

langkah 6 : tentukan output jaringan :

$y\_in = b_3 + z_1v_1 + z_2v_2$

$y = f(y\_in)$

langkah 7 : tentukan kesalahan dan perbaiki bobot :

jika t = y, maka tidak ada bobot yang diperbaiki

jika $t \neq y$, kerjakan langkah 7a – b untuk tiap unit hidden dgn input jaringan cukup dekat dgn 0 (antara –0.25 sampai 0.25). Mulai dengan yg paling dekat dgn 0 kemudian dilanjutkan dgn terdekat berikutnya, dst.

Langkah 7a : perbaiki output unit (dari 1 menjadi –1, atau sebaliknya)

Langkah 7b : hitung ulang output jaringan.

Jika kesalahan berkurang :

Perbaiki bobot pada unit ini (gunakan nilai output yg terakhir sebagai target dan pakai Delta Rule)

langkah 8 : tes kondisi berhenti

jika sudah tidak terjadi perubahan bobot atau jika nomer maksimum dari iterasi perubahan bobot sudah dikerjakan semuanya(langkah 2) maka hentikan. Jika tidak lanjutkan

4.4 Aplikasi

- Contoh 4.1 : pelatihan madaline untuk fungsi logika XOR

Contoh ini menggambarkan pemakaian Algoritma MR I untuk melatih Madaline untuk menyelesaikan masalah fungsi logika XOR. Disini hanya akan diperlihatkan perhitungan untuk perubahan bobot untuk iterasi yg pertama dari input pertama sampai keempat. Iterasi selanjutnya dicoba sendiri.

Pola pelatihan adalah :

x_1	x_2	t
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

Iterasi pertama :

langkah 0 : inisialisasi bobot :

bobot pada z_1

w_{11}	w_{21}	b_1
0.05	0.2	0.3

bobot pada z_2

w_{12}	w_{22}	b_2
0.1	0.2	0.15

bobot pada y

v_1	v_2	b_3
0.5	0.5	0.5

langkah 1 : mulai pelatihan

langkah 2 : untuk pasangan pelatihan pertama (1,1) : -1

langkah 3 : $x_1 = 1$, $x_2 = 1$

langkah 4 : $z\_in_1 = 0.3+0.05+0.2 = 0.55$

$z\_in_2 = 0.15+0.1+0.2 = 0.45$

langkah 5 : $z_1 = 1$, $z_2 = 1$

langkah 6 : $y\_in = 0.5+0.5+0.5 = 1.5$

$y = 1$

langkah 7 : $t - y = -1 - 1 = -2 \neq 0$, jadi kesalahan terjadi.

Karena t =-1, dan kedua unit z memiliki nilai input jaringan positif (z_1=0.55, z_2 = 0.45), maka

ubah bobot pada z_1 yaitu :

$b_1(baru) = b_1(lama) + \alpha(-1-z\_in_1)$
$= 0.3 + (0.5)(-1.55)$
$= -0.475$

$w_{11}(baru) = w_{11}(lama) + \alpha(-1-z\_in_1)x_1$
$= 0.05+ (0.5)(-1.55)$
$= -0.725$

$w_{21}(baru) = w_{21}(lama) + \alpha(-1-z\_in_1)x_2$
$= 0.2+ (0.5)(-1.55)$
$= -0.575$

ubah bobot pada z_2 yaitu :

$b_2(baru) = b_2(lama) + \alpha(-1-z\_in_2)$
$= 0.15 + (0.5)(-1.45)$
$= -0.575$

$w_{12}(baru) = w_{12}(lama) + \alpha(-1-z\_in_2)x_1$
$= 0.1+ (0.5)(-1.45)$
$= -0.625$

$w_{22}(baru) = w_{22}(lama) + \alpha(-1-z\_in_2)x_2$
$= 0.2+ (0.5)(-1.45)$
$= -0.525$

input kedua :

langkah 0 : inisialisasi bobot :

bobot pada z_1

w_{11}	w_{21}	b_1
-0.725	-0.575	-0.475

bobot pada z_2

w_{12}	w_{22}	b_2
-0.625	-0.525	-0.575

bobot pada y

v_1	v_2	b_3
0.5	0.5	0.5

langkah 1 : mulai pelatihan

langkah 2 : untuk pasangan pelatihan kedua (1,-1) : 1

langkah 3 : $x_1 = 1$, $x_2 = -1$

langkah 4 : $z\_in_1 = -0.475+-0.725+0.575 = -0.625$

$z\_in_2 = -0.575+-0.625+0.525 = -0.675$

langkah 5 : $z_1 = -1$, $z_2 = -1$

langkah 6 : $y\_in = 0.5+-0.5+-0.5 = -0.5$

$y = -1$

langkah 7 : $t - y = 1 + 1 = 2 \neq 0$, jadi kesalahan terjadi.

Karena t =1, dan kedua unit z memiliki nilai input jaringan negatif maka perbaiki bobot pada z_1 yaitu :

$b_1(baru) = b_1(lama) + \alpha(1-z\_in_1)$
$= -0.475 + (0.5)(1.625)$
$= 0.3375$

$w_{11}(baru) = w_{11}(lama) + \alpha(1-z\_in_1)x_1$
$= -0.725+ (0.5)(1.625)$
$= 0.0875$

$w_{21}(baru) = w_{21}(lama) + \alpha(1-z\_in_1)x_2$
$= -0.575+ (0.5)(1.625)(-1)$
$= -1.3875$

ubah bobot pada z_2 yaitu :

$b_2(baru) = b_2(lama) + \alpha(1-z\_in_2)$
$= -0.575 + (0.5)(1.675)$
$= 0.2625$

$w_{12}(baru) = w_{12}(lama) + \alpha(1-z\_in_2)x_1$
$= -0.625+ (0.5)(1.675)$
$= 0.2125$

$w_{22}(baru) = w_{22}(lama) + \alpha(1-z\_in_2)x_2$
$= -0.525+ (0.5)(1.675)(-1)$
$= -1.3625$

input ketiga :

langkah 0 : inisialisasi bobot :

bobot pada z_1

w_{11}	w_{21}	b_1
0.0875	-1.3875	0.3375

bobot pada z_2

w_{12}	w_{22}	b_2
0.2125	-1.3625	0.2625

bobot pada y

v_1	v_2	b_3
0.5	0.5	0.5

langkah 1 : mulai pelatihan

langkah 2 : untuk pasangan pelatihan ketiga (-1,1) : 1

langkah 3 : $x_1 = -1$, $x_2 = 1$

langkah 4 : $z\_in_1 = 0.3375+-0.0875+-1.3875 = -1.1375$
$z\_in_2 = 0.2625+-0.2125+-1.3625 = -1.3125$
langkah 5 : $z_1 = -1$, $z_2 = -1$
langkah 6 : $y\_in = 0.5+-0.5+-0.5 = -0.5$
$y = -1$
langkah 7 : $t - y = 1 + 1 = 2 \neq 0$, jadi kesalahan terjadi.
Karena t =-1, dan kedua unit z memiliki nilai input jaringan negatif maka ubah bobot pada z_1 yaitu :

$b_1(baru) = b_1(lama) + \alpha(1-z\_in_1)$
$= 0.3375 + (0.5)(2.1375)$
$= 1.40625$
$w_{11}(baru) = w_{11}(lama) + \alpha(1-z\_in_1)x_1$
$= 0.0875+ (0.5)(2.1375)(-1)$
$= -0.98125$
$w_{21}(baru) = w_{21}(lama) + \alpha(1-z\_in_1)x_2$
$= -1.3875+ (0.5)(2.1375)$
$= -0.31875$

ubah bobot pada z_2 yaitu :

$b_2(baru) = b_2(lama) + \alpha(1-z\_in_2)$
$= 0.2625 + (0.5)(2.3125)$
$= 1.41875$
$w_{12}(baru) = w_{12}(lama) + \alpha(1-z\_in_2)x_1$
$= 0.2125+ (0.5)(2.3125)(-1)$
$= -0.94375$
$w_{22}(baru) = w_{22}(lama) + \alpha(1-z\_in_2)x_2$
$= -1.3625+ (0.5)(2.3125)$
$= -0.20625$

input keempat :
langkah 0 : inisialisasi bobot :
bobot pada z_1

w_{11}	w_{21}	b_1
-0.98125	-0.31875	1.40625

bobot pada z_2

w_{12}	w_{22}	b_2
-0.94375	-0.20625	1.41875

bobot pada y

v_1	v_2	b_3
0.5	0.5	0.5

langkah 1 : mulai pelatihan
langkah 2 : untuk pasangan pelatihan pertama (-1,-1) : -1
langkah 3 : $x_1 = -1$, $x_2 = -1$
langkah 4 : $z\_in_1 = 1.40625+0.98125+0.31875 = 2.70625$
$z\_in_2 = 1.41875+0.94375+0.20625 = 2.56875$

langkah 5 : $z_1 = 1$, $z_2 = 1$
langkah 6 : y_in = 0.5+0.5+0.5 = 1.5
y = 1
langkah 7 : t - y = -1 - 1 = -2 ≠ 0, jadi kesalahan terjadi.
Karena t =-1, dan kedua unit z memiliki nilai input jaringan positif maka ubah bobot pada z_1 yaitu :

b_1(baru) = b_1(lama) + α(-1-z_in_1)
= 1.40625 + (0.5)(-3.70625)
= -0.446875
w_{11}(baru)= w_{11}(lama) + α(-1-z_in_1)x_1
= -0.98125+ 1.853125
= 0.871875
w_{21}(baru)= w_{21}(lama) + α(-1-z_in_1)x_2
= -0.31875+ 1.853125
= 1.5534375

ubah bobot pada z_2 yaitu :

b_2(baru) = b_2(lama) + α(-1-z_in_2)
= 1.41875 + -1.784375
= -0.365625
w_{12}(baru)= w_{12}(lama) + α(-1-z_in_2)x_1
= -0.94375+ 1.784375
= 0.840625
w_{22}(baru)= w_{22}(lama) + α(-1-z_in_2)x_2
= -0.20625+ 1.784375
= 1.578125

karena dalam iterasi yg pertama ini masih terjadi perbaikan bobot maka iterasi dilanjutkan ke iterasi yg kedua, ketiga,…,sampai kondisi berhenti tercapai. Silahkan hitung sendiri! Kalau tidak ada kekeliruan dalam perhitungan maka kondisi berhenti akan tercapai setelah 4 kali iterasi (4 epoch). Dalam hal ini hasil akhir bobot dan bias yaitu :

w_{11} = -0.73 w_{12} = 1.27
w_{21} = 1.53 w_{22} = -1.33
b_1 = -0.99 b_2 = -1.09

- Hasil *Testing* thd pola input dengan memakai bobot dan bias yg didapat dalam proses pelatihan :

Untuk pola input pertama (1,1) dengan target (–1)
$z\_in_1 = b_1 + w_{11}x_1 + w_{21}x_2$ = -0.99 + (-0.73)(1)+(1.53)(1) = -0.19
$z\_in_2 = b_2 + w_{12}x_1 + w_{22}x_2$ = -1.99 + (1.27)(1)+(-1.33)(1) = -2.05
$z_1 = f(z\_in_1) = f(-0.19) = -1$
$z_2 = f(z\_in_2) = f(-2.05) = -1$
$y\_in = b_3 + v_1z_1 + v_2z_2 = 0.5 + (0.5)(-1) + (0.5)(-1) = -0.5$
$y = f(y\_in) = f(-0.5) = -1$
jadi output y=-1 sesuai dengan target =-1
Untuk pola input kedua (1,-1) dengan target (1)

$z\_in_1 = b_1 + w_{11}x_1 + w_{21}x_2 = -0.99 + (-0.73)(1)+(1.53)(-1) = -3.25$
$z\_in_2 = b_2 + w_{12}x_1 + w_{22}x_2 = -1.99 + (1.27)(1)+(-1.33)(-1) = 0.61$
$z_1 = f(z\_in_1) = f(-3.25) = -1$
$z_2 = f(z\_in_2) = f(0.61) = 1$
$y\_in = b_3 + v_1z_1 + v_2z_2 = 0.5 + (0.5)(-1) + (0.5)(1) = 0.5$
$y = f(y\_in) = f(0.5) = 1$

jadi output y=1 sesuai dengan target =1

Untuk pola input ketiga (-1,1) dengan target (1)

$z\_in_1 = b_1 + w_{11}x_1 + w_{21}x_2 = -0.99 + (-0.73)(-1)+(1.53)(1) = 1.27$
$z\_in_2 = b_2 + w_{12}x_1 + w_{22}x_2 = -1.99 + (1.27)(-1)+(-1.33)(1) = -4.59$
$z_1 = f(z\_in_1) = f(1.27) = 1$
$z_2 = f(z\_in_2) = f(-4.59) = -1$
$y\_in = b_3 + v_1z_1 + v_2z_2 = 0.5 + (0.5)(1) + (0.5)(-1) = 0.5$
$y = f(y\_in) = f(0.5) = 1$

jadi output y=1 sesuai dengan target =1

Untuk pola input keempat (-1,-1) dengan target (-1)

$z\_in_1 = b_1 + w_{11}x_1 + w_{21}x_2 = -0.99 + (-0.73)(-1)+(1.53)(-1) = -1.79$
$z\_in_2 = b_2 + w_{12}x_1 + w_{22}x_2 = -1.99 + (1.27)(-1)+(-1.33)(-1) = -1.93$
$z_1 = f(z\_in_1) = f(-1.79) = -1$
$z_2 = f(z\_in_2) = f(-1.93) = -1$
$y\_in = b_3 + v_1z_1 + v_2z_2 = 0.5 + (0.5)(-1) + (0.5)(-1) = -0.5$
$y = f(y\_in) = f(-0.5) = -1$

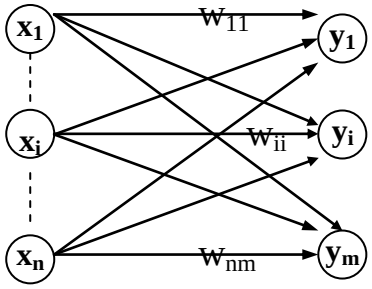
jadi output y =-1 sesuai dengan target = -1

BAB V Jaringan Heteroassociative Memory

- Jaringan syaraf HM adalah jaringan dimana bobot-bobotnya ditentukan sedemikian shg jaringan dpt menyimpan sekumpulan pola P. Tiap-tiap kumpulan pola adalah pasangan vektor (s(p), t(p)), dgn p = 1,2,…,P.
- Tiap vektor s(p) terdiri dari n komponen, dan tiap target t(p) terdiri dari m komponen
- Bobot dpt dicari dg memakai *aturan Hebb*(bab II) atau *aturan Delta* (bab IV)

5.1 Arsitektur

- Arsitektur dari Jaringan Heteroassociative memory :



Gambar 5.1 Arsitektur Heteroassociative Memory

5.2 Algoritma pelatihan

- Algoritma pelatihan HM biasanya memakai algoritma Hebb atau Delta rule. Algoritma Hebb untuk pengelompokan pola adalah :

langkah 0 : Bobot diberi nilai awal :

$w_{ij} = 0 \quad (i = 1,2,\ldots,n ; j = 1,2,\ldots,m)$

langkah 1 : untuk tiap pasangan input dan target, s : t, lakukan langkah 2 - 4 :

langkah 2 : set aktivasi untuk unit input :

$x_i = s_i \qquad (i = 1,2,\ldots,n)$

langkah 3 : set aktivasi untuk unit output :

$y_j = t_j$

langkah 4 : perbaiki nilai bobot :

$w_{ij}(\text{baru}) = w_{ij}(\text{lama}) + x_i y_j$

- Algoritma Hebb juga dapat ditentukan dgn perkalian matrik (outer product) antara pasangan vektor input(s) dan output (t) untuk mendapatkan bobot w :

$s = (s_1, \ldots, s_i, \ldots, s_n) \qquad t = (t_1, \ldots, t_j, \ldots, t_m)$

jika $S = s^T$ dan $T = t$ maka :

$$ST = \begin{bmatrix} s_1 \\ \vdots \\ s_i \\ \vdots \\ s_n \end{bmatrix} \begin{bmatrix} t_1 & \cdots & t_j & \cdots & t_m \end{bmatrix} = \begin{bmatrix} s_1 t_1 & \cdots & s_1 t_j & \cdots & s_1 t_m \\ \vdots & . & \vdots & . & \vdots \\ s_i t_1 & \cdots & s_i t_j & \cdots & s_i t_m \\ \vdots & . & \vdots & . & \vdots \\ s_n t_1 & \cdots & s_n t_j & \cdots & s_n t_m \end{bmatrix}$$

hasil kali ST adalah sama dengan matrik bobot w untuk menyimpan pasangan pola pelatihan s : t. Bobot w bisa juga dinyatakan dgn

$$w_{ij} = \sum_{p=1}^{P} s_i(p) t_j(p)$$

atau dalam bentuk yg lebih umum :

$$\mathrm{w} = \sum_{p=1}^{P} s^T(p) t(p)$$

5.3 Algoritma testing

○Algoritma testing jaringan HM yg dilatih dgn algoritma Hebb :

Langkah 0 : inisialisasi bobot dgn aturan Hebb atau Delta

Langkah 1 : untuk tiap vektor input, kerjakan langkah 2 – 4

Langkah 2 : set aktivasi dari unit input sama dgn vektor input sekarang

Langkah 3 : hitung input jaringan untuk unit output :

$$y\_in_j = \sum_{i=1}^{n} x_i w_{ij}$$

Langkah 4 : tentukan aktivasi dari unit output :

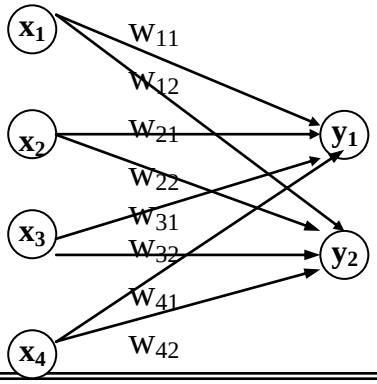
$$y_j = \begin{cases} 1 & jika\ y\_in_j > 0 \\ 0 & jika\ y\_in_j = 0 \\ -1 & jika\ y\_in_j < 0 \end{cases}$$ **(**untuk target bipolar)

$$y_j = \begin{cases} 1 & jika\ y\_in_j > 0 \\ -1 & jika\ y\_in_j \le 0 \end{cases}$$ **(**untuk target biner)

○Contoh 5.1:

Misalkan sebuah jaringan HM dilatih untuk menyimpan pola dari vektor input s = (s_1, s_2, s_3, s_4) dan vektor output t = (t_1, t_2) yaitu :

s_1	s_2	s_3	s_4	t_1	t_2
1	0	0	0	1	0
1	1	0	0	1	0
0	0	0	1	0	1
0	0	1	1	0	1



Gambar 5.2 Jaringan HM : 4 input dan 2 output

Pelatihan :

Langkah 0 : semua bobot diberi nilai 0

Langkah 1 : untuk pasangan pola yg pertama s : t (1,0,0,0) (1,0)

Langkah 2 : $x_1 = 1$; $x_2 = 0$; $x_3 = 0$; $x_4 = 0$

Langkah 3 : $y_1 = 1$; $y_2 = 0$

Langkah 4 : $w_{11}(\text{baru}) = w_{11}(\text{lama}) + x_1y_1 = 0 + 1 = 1$

Semua bobot yg lain tetap 0

Langkah 1 : untuk pasangan pola yg kedua s : t (1,1,0,0) (1,0)

Langkah 2 : $x_1 = 1$; $x_2 = 1$; $x_3 = 0$; $x_4 = 0$

Langkah 3 : $y_1 = 1$; $y_2 = 0$

Langkah 4 : $w_{11}(\text{baru}) = w_{11}(\text{lama}) + x_1y_1 = 1 + 1 = 2$

$w_{21}(\text{baru}) = w_{21}(\text{lama}) + x_2y_1 = 0 + 1 = 1$

Semua bobot yg lain tetap 0

Langkah 1 : untuk pasangan pola yg ketiga s : t (0,0,0,1) (0,1)

Langkah 2 : $x_1 = 0$; $x_2 = 0$; $x_3 = 0$; $x_4 = 1$

Langkah 3 : $y_1 = 0$; $y_2 = 1$

Langkah 4 : $w_{42}(\text{baru}) = w_{42}(\text{lama}) + x_4y_2 = 0 + 1 = 1$

Semua bobot yg lain tetap tidak berubah

Langkah 1 : untuk pasangan pola yg keempat s : t (0,0,1,1) (0,1)

Langkah 2 : $x_1 = 0$; $x_2 = 0$; $x_3 = 1$; $x_4 = 1$

Langkah 3 : $y_1 = 0$; $y_2 = 1$

Langkah 4 : $w_{32}(\text{baru}) = w_{32}(\text{lama}) + x_3y_2 = 0 + 1 = 1$

$w_{42}(\text{baru}) = w_{42}(\text{lama}) + x_4y_2 = 1 + 1 = 2$

Semua bobot yg lain tetap tidak berubah

Sehingga matrik bobotnya adalah :

$$W = \begin{bmatrix} 2 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 2 \end{bmatrix}$$

cara lain untuk mendapatkan matrik bobot tersebut adalah dengan memakai perkalian matrik (outer product). Adapun langkahnya adalah :

untuk pasangan pola yg pertama s = (1,0,0,0) ; t = (1,0)

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

untuk pasangan pola yg kedua s = (1,1,0,0) ; t = (1,0)

$$\begin{bmatrix}1\\1\\0\\0\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix}=\begin{bmatrix}1 & 0\\1 & 0\\0 & 0\\0 & 0\end{bmatrix}$$

untuk pasangan pola yg ketiga s = (0,0,0,1) ; t = (0,1)

$$\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix}=\begin{bmatrix}0 & 0\\0 & 0\\0 & 0\\0 & 1\end{bmatrix}$$

untuk pasangan pola yg keempat s = (0,0,1,1) ; t = (0,1)

$$\begin{bmatrix}0\\0\\1\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix}=\begin{bmatrix}0 & 0\\0 & 0\\0 & 1\\0 & 1\end{bmatrix}$$

matrik bobot untuk menyimpan semua pasangan pola adalah hasil penjumlahan dari setiap matrik pasangan pola diatas :

$$w=\begin{bmatrix}1 & 0\\0 & 0\\0 & 0\\0 & 0\end{bmatrix}+\begin{bmatrix}1 & 0\\1 & 0\\0 & 0\\0 & 0\end{bmatrix}+\begin{bmatrix}0 & 0\\0 & 0\\0 & 0\\0 & 1\end{bmatrix}+\begin{bmatrix}0 & 0\\0 & 0\\0 & 1\\0 & 1\end{bmatrix}=\begin{bmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{bmatrix}$$

- Selanjutnya kita lakukan *Testing* terhadap pola input dgn menggunakan bobot w hasil pelatihan.Dgn memakai algoritma testing dan fungsi aktivasi pola biner:

Langkah 0 : $w=\begin{bmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{bmatrix}$

Langkah 1 : pola input pertama, lakukan langkah 2 – 4

Langkah 2 : x = (1,0,0,0)

Langkah 3 : $y\_in_1 = x_1w_{11}+ x_2w_{21}+ x_3w_{31}+ x_4w_{41}$
$= 1(2) + 0(1) + 0(0) + 0(0) = 2$
$y\_in_2 = x_1w_{12}+ x_2w_{22}+ x_3w_{32}+ x_4w_{42}$
$= 1(0) + 0(0) + 0(0) + 0(0) = 0$

langkah 4 : $y_1 = f(y\_in_1) = f(2) = 1$
$y_2 = f(y\_in_2) = f(0) = 0$
(ini menunjukkan output yg benar untuk input pertama)

Langkah 1 : pola input kedua, lakukan langkah 2 – 4

Langkah 2 : x = (1,1,0,0)

Langkah 3 : $y\_in_1 = x_1w_{11} + x_2w_{21} + x_3w_{31} + x_4w_{41}$
$= 1(2) + 1(1) + 0(0) + 0(0) = 3$
$y\_in_2 = x_1w_{12} + x_2w_{22} + x_3w_{32} + x_4w_{42}$
$= 1(0) + 1(0) + 0(0) + 0(0) = 0$
langkah 4 : $y_1 = f(y\_in_1) = f(3) = 1$
$y_2 = f(y\_in_2) = f(0) = 0$
(ini menunjukkan output yg benar untuk input kedua)

Langkah 1 : pola input ketiga, lakukan langkah 2 – 4

Langkah 2 : x = (0,0,0,1)
Langkah 3 : $y\_in_1 = x_1w_{11} + x_2w_{21} + x_3w_{31} + x_4w_{41}$
$= 0(2) + 0(1) + 0(0) + 1(0) = 0$
$y\_in_2 = x_1w_{12} + x_2w_{22} + x_3w_{32} + x_4w_{42}$
$= 0(0) + 0(0) + 0(1) + 1(2) = 2$
langkah 4 : $y_1 = f(y\_in_1) = f(0) = 0$
$y_2 = f(y\_in_2) = f(2) = 1$
(ini menunjukkan output yg benar untuk input ketiga)

Langkah 1 : pola input keempat, lakukan langkah 2 – 4

Langkah 2 : x = (0,0,1,1)
Langkah 3 : $y\_in_1 = x_1w_{11} + x_2w_{21} + x_3w_{31} + x_4w_{41}$
$= 0(2) + 0(1) + 1(0) + 1(0) = 0$
$y\_in_2 = x_1w_{12} + x_2w_{22} + x_3w_{32} + x_4w_{42}$
$= 0(0) + 0(0) + 1(1) + 1(2) = 3$
langkah 4 : $y_1 = f(y\_in_1) = f(0) = 0$
$y_2 = f(y\_in_2) = f(3) = 1$
(ini menunjukkan output yg benar untuk input keempat

- Proses Testing thd pola input juga dpt dilakukan dgn menggunakan matrik yaitu :

Langkah 0 : $w = \begin{bmatrix} 2 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 2 \end{bmatrix}$

Langkah 1 : untuk vektor input pertama :
Langkah 2 : x = (1,0,0,0)
Langkah 3 : x W = (y_in_1, y_in_2)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 2 \end{bmatrix} = (2,0)$$

Langkah 4 : f(2) = 1 ; f(0) = 0
y = (1,0)
atau semua proses dari langkah 2 – 4 dinyatakan dgn :

x W = $(y\_in_1, y\_in_2) \longrightarrow y$

$$\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}2 & 0\\ 1 & 0\\ 0 & 1\\ 0 & 2\end{bmatrix} = (2,0) \longrightarrow (1,0)$$

atau singkatnya :

(1,0,0,0) W = $(2,0) \longrightarrow$ (1,0)

sehingga hasil testing untuk 3 vektor input lainnya adalah :

(1,1,0,0) W = $(3,0) \longrightarrow$ (1,0)

(0,0,0,1) W = $(0,2) \longrightarrow$ (0,1)

(0,0,1,1) W = $(0,3) \longrightarrow$ (0,1)

- Contoh 5.2 : Coba lakukan testing thd matrik bobot w yg didapat pada contoh 5.1 dgn data input yg mirip dgn data input pelatihan yaitu x = (0,1,0,0).

  Jawab :

  Pola input x (0,1,0,0) berbeda dgn vektor pelatihan s=(1,1,0,0) pada komponen yg pertama. Hasil testing dari pola input ini :

  (0,1,0,0) w = (1,0) $\longrightarrow$ (1,0)

  Hasil testing ini menunjukkan bahwa jaringan masih dpt mengelompokkan data testing ini ke dalam salah satu pola dalam pelatihan

- Contoh 5.2 : Coba lakukan testing thd matrik bobot w yg didapat pada contoh 5.1 dgn data input yg mirip dgn data input pelatihan yaitu x = (0,1,0,0).

  Jawab :

  Pola input x (0,1,0,0) berbeda dgn vektor pelatihan s=(1,1,0,0) pada komponen yg pertama. Hasil testing dari pola input ini :

  (0,1,0,0) w = (1,0) $\longrightarrow$ (1,0)

  Hasil testing ini menunjukkan bahwa jaringan masih dpt mengelompokkan data testing ini ke dalam salah satu target pola dalam pelatihan

- Contoh 5.3 : Coba lakukan testing thd matrik bobot w yg didapat pada contoh 5.1 dgn data input yg tidak mirip dgn data input pelatihan yaitu x = (0,1,1,0).

  Jawab :

  Pola input x (0,1,1,0) berbeda dgn vektor pelatihan s=(1,1,0,0) pada 2 komponen yaitu komponen pertama dan ketiga. Hasil testing dari pola input ini :

  (0,1,1,0) w = (1,1) $\longrightarrow$ (1,1)

  Hasil testing ini menunjukkan bahwa jaringan tidak dpt mengelompokkan data testing ini ke dalam salah satu target pola dalam pelatihan. Jadi jika ada 2 buah komponen yg berbeda pada vektor input maka jaringan tidak akan mampu mengelompokkan data input tsb ke dalam salah satu target pelatihan.

- Contoh 5.4:

Misalkan jaringan HM dlm contoh 5.1 dilatih untuk menyimpan pola vektor input dan output bipolar yaitu :

s_1	s_2	s_3	s_4	t_1	t_2
1	-1	-1	-1	1	-1

1	1	-1	-1	1	-1
-1	-1	-1	1	-1	1
-1	-1	1	1	-1	1

Jika kita pakai cara perkalian matrik (outer product). Adapun langkahnya adalah :

untuk pasangan pola yg pertama s = (1,-1,-1,-1) ; t = (1,-1)

$$\begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

untuk pasangan pola yg kedua s = (1,1,-1,-1) ; t = (1,-1)

$$\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

untuk pasangan pola yg ketiga s = (-1,-1,-1,1) ; t = (-1,1)

$$\begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}$$

untuk pasangan pola yg keempat s = (-1,-1,1,1) ; t = (-1,1)

$$\begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

matrik bobot untuk menyimpan semua pasangan pola adalah hasil penjumlahan dari setiap matrik pasangan pola diatas :

$$w = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & -4 \\ 2 & -2 \\ -2 & 2 \\ -4 & 4 \end{bmatrix}$$

- Salah satu kelebihan memakai pola bipolar adalah dalam menyatakan bentuk gangguan (noise) dalam data input. Jika kita anggap ada dua bentuk noise yaitu : "*data salah* (mistake)" dan "*data hilang* (missing)" dan jika noise dinyatakan dalam pola bipolar yaitu :

1 → ya

–1 → tidak

0 → ragu-ragu (unsure)

jadi untuk kasus *data hilang* (missing) :

ya = ragu ; tidak = ragu atau 1 dinyatakan dgn 0 ; -1 dinyatakan dgn 0

sedang untuk kasus *data salah* (mistake) :

ya = tidak ; tidak = ya atau 1 dinyatakan dgn –1 ; –1 dinyatakan dgn 1

- Salah satu kelebihan memakai pola bipolar adalah dalam menyatakan bentuk gangguan (noise) dalam data input. Jika kita anggap ada dua bentuk noise yaitu : "*data salah* (mistake)" dan "*data hilang* (missing)" dan jika noise dinyatakan dalam pola bipolar yaitu :
- Misal jika kita testing matrik bobot W yg didapatkan dalam contoh 5.4.(bipolar) dengan input "data salah" pada 2 komponen (data pertama dan ketiga) yaitu (-1,1,1-1)

  (-1,1,1,-1) W = (0,0) → (0,0)

  hal ini menunjukkan respon yg masih salah dari jaringan thd pola input tsb.
- Dan misal jika kita testing matrik bobot W yg didapatkan dalam contoh 5.4.(bipolar) dengan input "data hilang" pada 2 komponen (data pertama dan ketiga) yaitu (0,1,0-1) maka :

  (0,1,0,-1) W = (6,-6) → (1,-1)

  hal ini menunjukkan respon yg benar dari jaringan thd pola input tsb.
- Contoh 5.5. : Misal jika kita melatih jaringan syaraf HM dgn 3 pasangan pola pelatihan spt di bawah ini. Input (s) terdiri dari 63 komponen dan output (t) terdiri dari 15 komponen.

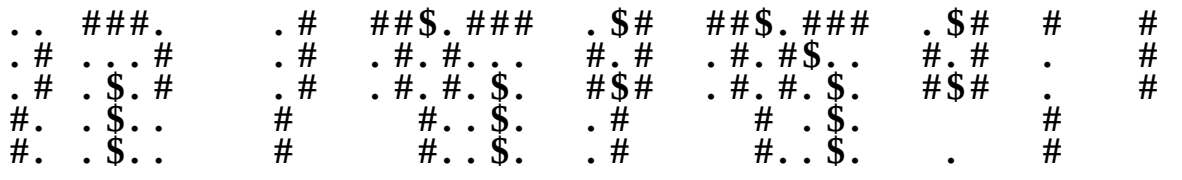
Pola 1		Pola 2		Pola 3	
. . . #. .	.	#####	..	. . ###	. .
. . . #. .	.	#. . . .	#.	. #. . .	# .
. . #. #.	. .	#. #. . .	. ##	#. #. . .	. ## # #
. . #. #.	. #	. ##. . .	#. #	. ##. . .	. . # . .
. . ###.	. #	#######	. . #	#. #. . .	. . # . .
. # . . . #	. #	. ##. . .	#. #	. ##. . .	. . # . .
. # . . . #	. #	. ##. . .	. ##	#. #. . .	. ## # #
#.	#	#. . . .	#.	. #. . .	# .
#.	#	#####	. .	. . ###	. .
s(1)	t(1)	s(2)	t(2)	s(3)	t(3)

dimana : "#" = 1 dan "." = -1

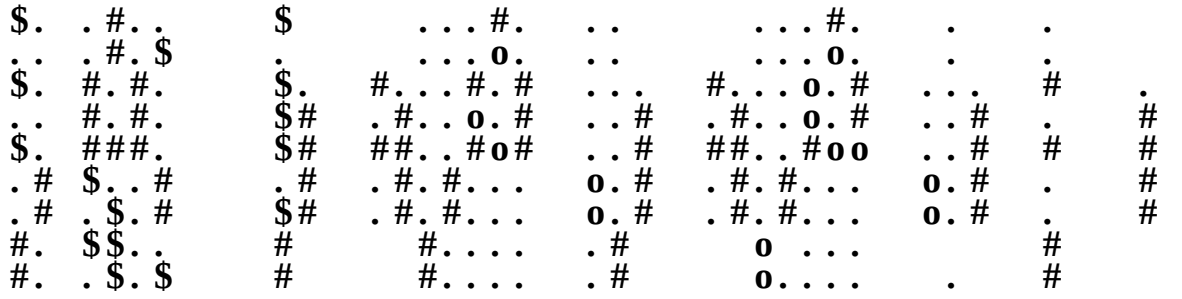
a)Gambarkan arsitektur jaringan HM dan tentukan matrik bobot W dgn pelatihan. Kemudian lakukan testing thd pola input dgn bobot W tersebut. Bagaimana hasilnya?

b)Buktikan hasil testing dari pola-pola berikut ini :

$. . #. .	.	$. . #.	. $	$. . #.	. $
. . . #. .	.	. . . #.	. .	. . . #.	$.
. . #. #.	. .	#. $. #. #	. . .	#. $. #. #	. . . # .
. . #. #.	. #	. #. . #. #	. . #	. #. . #. #	. . # . #



input output input output input output

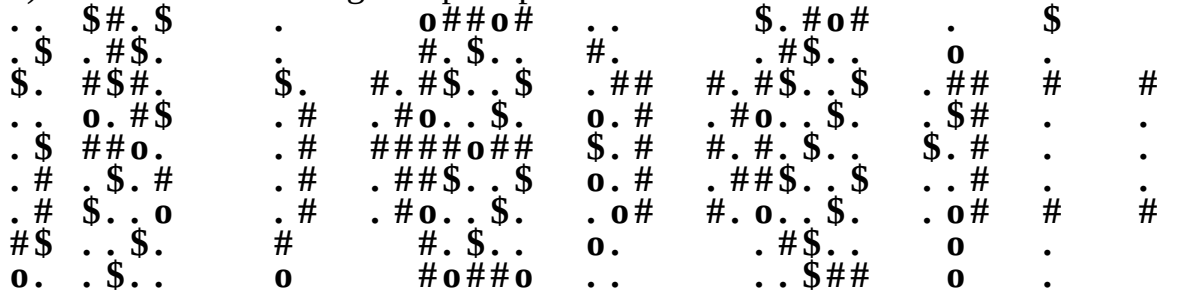


input output input output input output

dimana noise ”$” dan ”o” dilambangkan dengan :

$ = 1 ; o = -1

c)Buktikan hasil testing dari pola-pola berikut ini :



input output input output input output

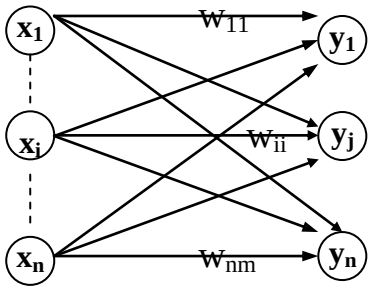
dimana noise ”$” dan ”o” dilambangkan dengan :

$ = 1 ; o = -1

BAB VI Jaringan Autoassociative Memory

- Jaringan syaraf AM merupakan bentuk khusus dari Jaringan HM
- Jaringan AM, vektor input dan target dari pola pelatihan adalah identik
- Proses pelatihan disebut dgn *proses penyimpanan vektor*
- Vektor yg sudah tersimpan ini dapat dikenali dari vektor input yg tidak sempurna (noise)
- Unjuk kerja jaringan dpt dinilai dari kemampuannya untuk menghasilkan pola-pola yg sudah tersimpan walaupun diberikan input yang tidak sempurna (bernoise atau hilang sebagian)
- Umumnya kemampuan jaringan yg dilatih dengan pola vektor bipolar akan lebih baik dari pelatihan dengan vektor biner
- Seringkali diagonal dari matrik bobot hasil pelatihan nilainya dibuat 0. Hal ini untuk meningkatkan kemampuan jaringan khususnya jika jaringan dipakai untuk menyimpan lebih dari satu vektor

6.1 Arsitektur



Gambar 6.1 Arsitektur Autoassociative Memory

6.2 Algoritma pelatihan

- Algoritma pelatihan AM biasanya memakai algoritma Hebb
- Vektor input dan output adalah sama (jml unit output = jml unit input)

langkah 0 : Bobot diberi nilai awal :

$w_{ij} = 0 \quad (i = 1,2,\ldots,n ; j = 1,2,\ldots,m)$

langkah 1 : untuk tiap vektor yg akan disimpan, lakukan langkah 2 - 4 :

langkah 2 : set aktivasi untuk unit input :

$x_i = s_i \qquad (i = 1,2,\ldots,n)$

langkah 3 : set aktivasi untuk unit output :

$y_j = t_j$

langkah 4 : perbaiki nilai bobot :

$w_{ij}(\text{baru}) = w_{ij}(\text{lama}) + x_i y_j$

- biasanya dalam praktek bobot W dapat dihitung dgn rumus :

$$w = \sum_{p=1}^{P} s^T(p)\, s(p)$$

6.3. Aplikasi

- sesudah pelatihan jaringan AM dpt dipakai untuk mengetes apakah sebuah vektor input dpt dikenali atau tidak oleh jaringan
- Jaringan akan dpt mengenali sebuah vektor jika dia bisa menghasilkan pola aktivasi pada unit output yg sama dengan vektor yang tersimpan dalam jaringan tsb.
- Langkah-langkah testing adalah :

langkah 0 : set bobot (dgn aturan Hebb atau Outer Product)
langkah 1 : untuk tiap vektor input yg akan ditesting, lakukan langkah 2 - 4 :
langkah 2 : set aktivasi untuk unit input sama dengan vektor input
langkah 3 : hitung input jaringan utuk masing-masing unit output

$$y\_in_j = \sum_i x_i w_{ij} \quad ; j = 1,2,...n$$

langkah 4 : gunakan fungsi aktivasi (j = 1,2,…,m)

$$y_j = f(y\_in_j) = \begin{cases} 1 \; jika \; y\_in_j > 0 \\ -1 \; jika \; y\_in_j \le 0 \end{cases}$$

contoh 6.1 :

Andaikan kita ingin menyimpan sebuah vektor input (1,1,1,-1) ke dalam jaringan A.M. Setelah pelatihan kita ingin mengetes apakah jaringan dapat mengenali vektor input tsb.

Langkah 0 : vektor s = (1,1,1,-1) disimpan dengan matrik bobot :

$$w = \sum_{p=1}^{P} s^T(p)s(p) = w = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

Hasil testing dari vektor input thd matrik bobot w hasil pelatihan :

Langkah 1 : untuk tiap-tiap vektor input :

Langkah 2 : x = (1,1,1,-1)

Langkah 3 : y_in = (4,4,4,-4)

Langkah 4 : y = f(4,4,4,-4) = (1,1,1,-1)

Ternyata hasil output y sama dgn vektor input. Berarti jaringan dpt mengenali vektor input tsb.

Proses testing juga bisa dilakukan dengan cara :

(1,1,1,-1) W = (4,4,4,-4) →(1,1,1,-1)

Contoh 6.2 :

Menguji jaringan AM dgn beberapa vektor input dgn satu komponen salah(mistake). Adapun vektor tsb adalah : (-1,1,1,-1), (1,-1,1,-1), (1,1,-1,-1), dan (1,1,1,1)

Hasil testing :

(-1,1,1,-1)W = (2,2,2,-2) →(1,1,1,-1)
(1,-1,1,-1)W = (2,2,2,-2) →(1,1,1,-1)
(1,1,-1,-1)W = (2,2,2,-2) →(1,1,1,-1)
(1,1,1,1) W = (2,2,2,-2) →(1,1,1,-1)

Contoh 6.3 :

Ujilah jaringan AM tsb dgn vektor input dimana sebuah komponennya hilang. Adapun vektor-vektor tsb adalah : (0,1,1,-1), (1,0,1,-1), (1,1,0,-1) dan (1,1,1,0)

Hasil testing :

(0,1,1,-1)W = (3,3,3,-3) →(1,1,1,-1)
(1,0,1,-1)W = (3,3,3,-3) →(1,1,1,-1)
(1,1,0,-1)W = (3,3,3,-3) →(1,1,1,-1)
(1,1,1,0) W = (3,3,3,-3) →(1,1,1,-1)

6.4. Kapasitas Penyimpanan

○Adalah banyaknya pola yg dpt disimpan dlm jaringan AM

Contoh 6.4 :

Penyimpanan 2 vektor ortogonal dalam jaringan AM

Vektor 1 : (1,1,-1,-1)
Vektor 2 : (-1,1,1,-1)
Jika w_1 dan w_2 adalah matrik bobot dari vektor 1 dan vektor 2 :

$$w = \sum_{p=1}^{P} s^T(p)\, s(p)$$

$$w_1 = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix}$$

2 vektor x dan y dikatakan ortogonal jika memenuhi :

$$xy^T = \sum_i x_i y_i = 0$$

Hasil testing :

Vektor 1 : (1,1,-1,-1)
(1,1,-1,-1) W = (2,2,-2,-2) →(1,1,-1,-1)
Vektor 2 : (-1,1,1,-1)
(-1,1,1,-1) W = (-2,2,2,-2) →(-1,1,1,-1)

Jadi kedua vektor tsb dpt dikenali oleh jaringan

Contoh 6.5 :

Penyimpanan 3 vektor yg ortogonal satu sama lain dlm jaringan AM

Vektor 1 : (1,1,-1,-1)
Vektor 2 : (-1,1,1,-1)
Jika w_1 ,w_2 , dan w_3 adalah matrik bobot dari vektor 1,vektor 2, dan vektor :

$$w = \sum_{p=1}^{P} s^T(p)\, s(p)$$

$$w_1 = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix} \quad w_3 = \begin{bmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{bmatrix}$$

$$w = w_1 + w_2 + w_3 = \begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

Hasil testing :

Vektor 1 : (1,1,-1,-1)
(1,1,-1,-1) W = (1,1,-1,-1)→(1,1,-1,-1)
Vektor 2 : (-1,1,1,-1)
(-1,1,1,-1) W = (-1,1,1,-1)→(-1,1,1,-1)
Vektor 3 : (-1,1,-1,1)
(-1,1,-1,1) W = (-1,1,-1,1)→(-1,1,-1,1)

Jadi ketiga vektor tsb dapat dikenali oleh jaringan

Contoh 6.6 :

Menyimpan 2 vektor input pelatihan yang non orthogonal. Adapun vektor-vektor tsb : (1,-1,-1,1) dan (1,1,-1,1)

$$w_1 = \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix} \qquad w_2 = \begin{bmatrix} 0 & 1 & -1 & 1 \\ 1 & 0 & -1 & 1 \\ -1 & -1 & 0 & -1 \\ 1 & 1 & -1 & 0 \end{bmatrix}$$

$$w = w_1 + w_2 = \begin{bmatrix} 0 & 0 & -2 & 2 \\ 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & -2 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

Hasil testing :

Vektor 1 : (1,-1,-1,1)
(1,-1,-1,1) W = (0,0,-2,4) →(-1,-1,-1,1)
Vektor 2 : (1,1,-1,1)
(1,1,-1,1) W = (4,0,-2,4) →(1,-1,-1,1)

Hasil ini menunjukkan vektor input pelatihan tidak dapat dikenali.

Latihan :

Seandainya kita punya 4 vektor input pelatihan : (1,1,-1,-1), (-1,1,1,-1),(-1,1,-1,1), dan (1,1,1,1). Hitunglah hasil pelatihan dari vektor tsb. Selanjutnya tentukan hasil testing dari matrik bobot hasil pelatihan thd vektor input pelatihan.

Teorema :

Kapasitas penyimpanan jaringan AM bergantung pada jumlah komponen dari vektor tersimpan dan hubungan antara vektor tersimpan tsb. Lebih banyak vektor yang dapat disimpan jika vektor-vektor tsb saling ortogonal satu sama lain. Menurut Szu (1989) dibuktikan bahwa :

Jika kita mempunyai (n-1) vektor bipolar yang saling ortogonal, dimana masing-masing vektor memiliki n komponen maka semua (n-1) vektor tsb dapat disimpan dalam jaringan AM (diagonal matrik bobot dibuat 0)

6.5 Jaringan Autoassociative dengan iterasi (iterative Autoassociative Net)

- Dalam beberapa hal jaringan AM tidak dapat memberikan respon/output dengan cepat thd vektor input, akan tetapi memberi respon yang cukup mirip dgn pola yang tersimpan dan selanjutnya memakai respon yang pertama ini sbg input untuk jaringan lagi (iterasi)

Contoh 6.7 :

Jika kita menyimpan vektor input (1,1,1,-1) maka akan didapatkan W :

$$w = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

a)Jika kita lakukan testing dgn memakai vektor input tsb dimana 3 komponennya hilang yaitu : (1,0,0,0) maka hasil testing :

(1,0,0,0) W = (0,1,1,-1)→iterasi

(0,1,1,-1)W = (3,2,2,-2)→(1,1,1,-1)

Jadi vektor (1,0,0,0) dapat dikenali setelah melewati 2 kali iterasi

b)Jika kita beri vektor input : (0,1,0,0) maka hasil testing :

(0,1,0,0) W = (1,0,1,-1)→iterasi

(karena belum sama dgn vektor yg tersimpan)

(1,0,1,-1)W = (2,3,2,-2)→(1,1,1,-1) →stop

(karena sudah sama dgn vektor yg tersimpan)

Jadi vektor (0,1,0,0) juga dpt dikenali setelah 2 kali iterasi

c)Jika kita beri vektor input : (0,0,0,1) maka hasil testing :

(0,0,0,1) W = (-1,-1,-1,0)→iterasi

(-1,-1,-1,0)W = (-2,-2,-2,3)→(-1,-1,-1,1) →iterasi

(-1,-1,-1,1)W = (-3,-3,-3,3)→(-1,-1,-1,1) →stop

(vektor aktivasi sudah sama dgn vektor aktivasi pada iterasi sebelumnya)

Jadi vektor (0,0,0,1), hasil testing menunjukkan respon –1 dikali vektor tersimpan (1,1,1,-1). Vektor tersimpan dan nilai negatif dari vektor tsb adalah titik stabil dari jaringan.

d)Jika kita beri vektor input dgn 2 komponen salah (-1,-1,1,-1), hasil testing :

(-1,-1,1,-1) W = (1,1,-1,1)→iterasi

(1,1,-1,1) W = (-1,-1,1,-1)→ stop

(karena kembali menghasilkan vektor input semula)

Jika iterasi dilanjutkan maka pasti akan menghasilkan 2 vektor aktivasi yg dihasilkan sebelumnya secara berulang-ulang.

Catatan :

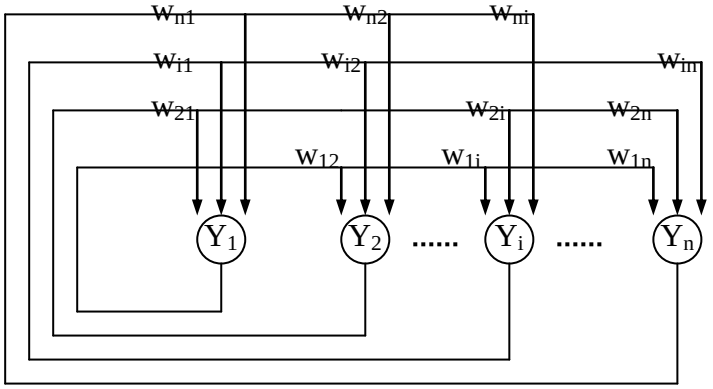
untuk vektor bipolar dgn 2 k komponen, dimana k buah komponennya salah akan menghasilkan vektor yg ortogonal dgn vektor tersimpan. Dalam contoh diatas vektor (-1,-1,1,-1) ortogonal dgn (-1,-1,-1,1)

BAB VII Jaringan Hopfield diskret

- Dikembangkan oleh John Hopfield (ahli fisika, pemenang hadiah nobel 1982)
- Struktur jaringannya terkoneksi secara penuh yaitu setiap unit terhubung dgn setiap unit yang lain
- Jaringan memiliki bobot simetris tanpa ada konenksi pada diri sendiri shg

  $W_{ij} = W_{ji}$ dan $w_{ii} = 0$

7.1 Arsitektur



Gambar 7.1 Arsitektur Jaringan Hopfield diskret

- Hanya satu unit mengubah aktivasinya pada satu waktu (berdasarkan sinyal yg diterima oleh unit tsb dari unit yang lain)
- Tiap-tiap unit melanjutkan menerima sinyal eksternal yg berasal dari unit yang lain dalam jaringan .

7.2 Algoritma Pelatihan

- Ada beberapa versi jaringan Hopfield diskret
- Versi pertama memakai vektor input biner (1982) :

  Untuk menyimpan pola biner s(p), p = 1,…,P dimana

  $s(p) = (s_1(p), \ldots, s_i(p), \ldots, s_n(p))$

  matrik bobot $w = \{w_{ij}\}$

$$w_{ij} = \sum_p [2s_i(p) - 1][2s_j(p) - 1] \; untuk \; i \neq j$$

$$w_{ij} = 0 \; untuk \; i = j$$

- Versi kedua memakai vektor input bipolar (1984) :

$$w_{ij} = \sum_p s_i(p) s_j(p) \; untuk \; i = j$$

$$= 0 \; untuk \; i = j$$

7.3 Algoritma Aplikasi (testing)

langkah 0 : inisialisasi bobot untuk menyimpan pola (gunakan aturan Hebb)

selama aktivasi tidak konvergen, kerjakan langkah 1- 7 :
langkah 1 : untuk tiap-tiap vektor input x , kerjakan langkah 2 – 6 :
langkah 2 : set aktivasi awal jaringan sama dgn vektor input eksternal x :
$y_i = x_i$ (i = 1,2, … ,n)
langkah 3 : lakukan langkah 4-6 untuk tiap-tiap unit Y_i
(unit-unit diubah secara acak)
langkah 4 : hitung input jaringan :

$$y\_in = x_i + \sum_j y_j w_{ij}$$

langkah 5 : tentukan aktivasi (sinyal output) :

$$y_i = \begin{cases} 1 \ \ jika \ \ y\_in_i > \theta_i \\ y_i \ \ jika \ \ y\_in_i = \theta_i \\ 0 \ \ jika \ \ y\_in_i < \theta_i \end{cases}$$

langkah 6 : kirimkan nilai y_i ke semua unit yang lain
langkah 7 : testing apakah sudah konvergen
catatan :
-nilai θ_i biasanya diambil sama dengan 0
-urutan perubahan dari unit-unit dilakukan secara acak
contoh 7.1 :

Seperti contoh sebelumnya, kita ingin menyimpan vektor biner (1,1,10) atau

dalam bentuk bipolar (1,1,1,-1) ke dalam jaringan Hopfield. Selanjutnya jika kita testing dengan vektor input dimana 2 komponennya salah yaitu (0,0,1,0). Unit akan mengubah aktivasinya dalam urutan acak. Misalkan dlm hal ini urutan perubahan aktivasi unit adalah Y_1, Y_4, Y_3, Y_2.
Langkah 0 : inisialisasi bobot untuk menyimpan pola :

$$w = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

langkah 1 : vektor input x = (0,0,1,0)
langkah 2 : y = (0,0,1,0)
langkah 3 : Pilih unit Y_1 untuk mengubah aktivasinya :
langkah 4 : $y\_in_1 = x_1 + \sum_j y_j w_{j1} = 0 + 1$
langkah 5 : $y\_in_1 > 0 \rightarrow y_1 = 1$
langkah 6 : $y = (1,0,1,0)$
langkah 3 : Pilih unit Y_4 untuk mengubah aktivasinya :
langkah 4 : $y\_in_4 = x_4 + \sum_j y_j w_{j4} = 0 + -2$

langkah 5 : $y\_in_4 < 0 \rightarrow y_4 = 0$

langkah 6 : $y = (1,0,1,0)$

langkah 3 : Pilih unit Y_3 untuk mengubah aktivasinya :

langkah 4 : $y\_in_3 = x_3 + \sum_j y_j w_{j3} = 1+1$

langkah 5 : $y\_in_3 > 0 \rightarrow y_3 = 1$

langkah 6 : $y = (1,0,1,0)$

langkah 3 : Pilih unit Y_2 untuk mengubah aktivasinya :

langkah 4 : $y\_in_2 = x_2 + \sum_j y_j w_{j2} = 0+2$

langkah 5 : $y\_in_2 > 0 \rightarrow y_2 = 1$

langkah 6 : $y = (1,1,1,0)$

langkah 7 : Testing apakah sudah konvergen

Karena beberapa aktivasi sudah berubah selama satu siklus iterasi, maka sekurang-kurangnya perlu dilakukan satu kali iterasi lagi. Dan nantinya dapat dilihat iterasi selanjutnya tidak akan mengubah aktivasi dari setiap unit. Jaringan sudah konvergen thd vektor yang tersimapan.

7.4 Fungsi Energi

- Hopfield (1984) membuktikan bahwa jaringan akan konvergen menuju sebuah titik stabil dng memperhitungkan sebuah fungsi energi atau disebut fungsi *Lyapunov*.
- Fungsi energi adalah suatu fungsi yg punya batas bawah dan fungsi yg nilainya tidak bertambah (non-increasing) dari keadaan sistem
- Keadaan sistem adalah vektor aktivasi dari unit-unit dalam sistem
- Jadi jika fungsi energi dpt ditemukan maka jaringan akan konvergen pada sekumpulan aktivasi yg stabil
- Fungsi energi untuk jaringan Hopfield diskret dinyatakan :

$$E = -0.5 \sum_{i \neq j} \sum_j y_i y_j w_{ij} - \sum_i x_i y_i + \sum_i \theta_i y_i$$

- Jika aktivasi dari jaringan berubah sebesar Δy_i, maka energi akan berubah sebesar ΔE,

$$\Delta E = -\left[\sum_j y_j w_{ij} + x_i - \theta_i \right] \Delta y_i$$

persamaan ini berdasarkan pada kenyataan bahwa hanya satu unit dapat mengubah aktivasinya pada satu waktu

- Ada 2 kasus dimana perubahan Δy_i akan terjadi dalam aktivasi neuron Y_i

-Jika y_i positif, maka y_i akan berubah menjadi nol jika

$$x_i + \sum_j y_j w_{ij} < \theta_i$$

dlm hal ini terjadi perubahan negatif pada y_i, shg untuk kasus ini $\Delta E < 0$ (negatif)
-Jika y_i nol, maka y_i akan berubah jadi positif jika

$$x_i + \sum_j y_j w_{ij} > \theta_i$$

dlm hal ini terjadi perubahan positif pada y_i, shg untuk kasus ini $\Delta E < 0$ (negatif)
Jadi $\Delta E < 0$ (negatif)

- Δy_i bernilai positif hanya jika $\left[x_i + \sum_j y_j w_{ij} - \theta_i \right]$ bernilai positif dan

  Δy_i bernilai negatif hanya jika $\left[x_i + \sum_j y_j w_{ij} - \theta_i \right]$ bernilai negatif dan
- Shg nilai energi tidak dapat bertambah besar (non-increasing)
- Karena energi terbatas, maka jaringan akan mencapai titik stabil sedemikian hingga energi tidak akan bertambah thd iterasi selanjutnya
- Aspek penting dari algoritma Hopfield adalah perubahan energi hanya bergantung pada perubahan aktivasi dari satu unit dan matrik bobot simetris dgn komponen diagonal bernilai nol

7.5 Kapasitas Penyimpanan

- Hopfield menemukan bahwa banyaknya pola biner yg dapat disimpan dalam jaringan adalah :

  $$P \approx 0.15n$$

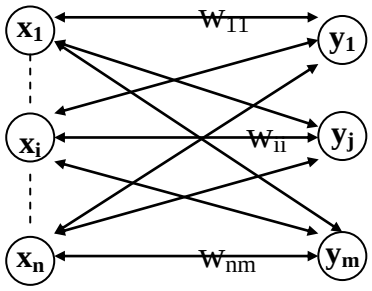
  dimana n = jumlah neuron dlm jaringan
- McEliece, Posner, Rodemich, dan Venkatesh (1987) menemukan bahwa banyaknya pola biner yg dapat disimpan dalam jaringan adalah :

  $$P \approx \frac{n}{2\log_2 n}$$

BAB VIII Bidirectional Associative Memory (B A M)

- BAM merupakan salah bentuk dari jaringan Heteroassociative memory, yg dikembangkan oleh Kosko (1988)
- BAM menyimpan pola pelatihan dalan matrik n x m yang merupakan perkalian dari vektor input dan target pelatihan (outer product)
- Arsitektur BAM terdiri dari 2 lapisan yg dihubungkan oleh lintasan koneksi bobot
- Jaringan akan beriterasi, mengirimkan sinyal pulang pergi antara 2 lapisan sampai semua neuron menjadi stabil (semua aktivasi neuron konstan)
- BAM dpt memberikan respon thd input dari kedua lapisan
- Bobot bersifat 2 arah (bidirectional) dan kita akan menyebut kedua lapisan sbg lapisan X dan lapisan Y
- Ada 3 jenis BAM : biner, bipolar , dan kontinu

8.1 Arsitektur BAM



Gambar 8.1 Arsitektur Bidirectional Associative Memory

- Lapisan x terdiri dari n unit dan lapisan y terdiri dari m unit
- Koneksi antara kedua lapisan bersifat 2 arah, yaitu jika matrik bobot untuk sinyal yg dikirim dari lapisan x ke lapisan y adalah w, maka matrik bobot untuk sinyal yg dikirim dari lapisan y ke lapisan x adalah w^T

8.2 Algoritma

8.2.1 BAM Diskret

- 2 bentuk pola pelatihan BAM yaitu : biner dan bipolar sangat berhubungan, dimana bobot dihitung dgn penjumlahan dari perkalian vektor pola pelatihan
- Fungsi aktivasi adalah fungsi tangga dgn threshold tidak nol
- Telah dibuktikan oleh Kosko(1988) dan Haines-Nielsen(1988) bahwa vektor bipolar dpt meningkatkan kinerja dari jaringan

Penentuan Nilai Bobot

- Matrik bobot ditentukan dgn aturan Hebb.
- Untuk pasangan vektor pelatihan *biner* maka bobot ditentukan dgn :

$$w_{ij} = \sum_p (2s_i(p) - 1)(2t_j(p) - 1)$$

- Dan jika vektor pelatihan *bipolar* maka bobot ditentukan dgn :

$$w_{ij} = \sum_p s_i(p) t_j(p)$$

Fungsi Aktivasi

- Fungsi aktivasi untuk BAM adalah fungsi tangga (step)
- Untuk vektor input *biner* maka fungsi aktivasi pada lapisan y :

$$y_j = \begin{cases} 1 \;\; jika\, y\_in_j > 0 \\ y_j \;\; jika\, y\_in_j = 0 \\ 0 \;\; jika\, y\_in_j < 0 \end{cases}$$

- Dan fungsi aktivasi pada lapisan x :

$$x_i = \begin{cases} 1 \;\; jika\, x\_in_i > 0 \\ x_i \;\; jika\, x\_in_i = 0 \\ 0 \;\; jika\, x\_in_i < 0 \end{cases}$$

- Sementar untuk vektor input *bipolar* maka fungsi aktivasi pada lapisan y :

$$y_j = \begin{cases} 1 \;\; jika\, y\_in_j > \theta_j \\ y_j \;\; jika\, y\_in_j = \theta_j \\ -1 \; jika\, y\_in_j < \theta_j \end{cases}$$

- Dan fungsi aktivasi pada lapisan x :

$$x_i = \begin{cases} 1 \;\; jika\, x\_in_i > 0 \\ x_i \;\; jika\, x\_in_i = 0 \\ 0 \;\; jika\, x\_in_i < 0 \end{cases}$$

Algoritma

Langkah 0 : inisialisasi bobot untuk menyimpan pola pelatihan
Inisialisasi semua aktivasi menjadi 0

Langkah 1 : untuk tiap input testing lakukan langkah 2-6

Langkah 2a : berikan pola input x ke lapisan X
(set aktivasi lap. X dgn pola input sekarang)

Langkah 2b : berikan pola input y ke lapisan Y
(salah satu pola input dpt bernilai nol)

Langkah 3 : selama aktivasi belum konvergen, lakukan langkah 4-6

Langkah 4 : ubah aktivasi dari unit-unit dlm lapisan Y.

Hitung input jaringan :

$$y\_in_j = \sum_i w_{ij} x_i$$

Hitung aktivasi :

$$y_j = f(y\_in_j)$$

kirim sinyal ke lapisan X

Langkah 5 : ubah aktivasi dari unit-unit dlm lapisan X.

Hitung input jaringan :

$$x\_in_i = \sum_j w_{ij} x_j$$

Hitung aktivasi :

$$x_i = f(x\_in_i)$$

kirim sinyal ke lapisan Y

Langkah 6 : Tes apakah sudah konvergen: jika vektor aktivasi x dan y sudah mencapai kestabilan maka stop pelatihan, jika tidak lanjutkan dan kembali ke langkah 1

8.2.2 BAM Kontinu

- Dalam BAM kontinu (Kosko,1988), sinyal input ditransformasikan menuju output secara kontinu dan halus(smooth) dalam range [0,1], dengan memakai fungsi aktivasi sigmoid logistik
- Untuk vektor input biner (s(p),t(p)), p =1,2,…P, bobot ditentukan dgn :

$$w_{ij} = \sum_p (2s_i(p) - 1)(2t_j(p) - 1)$$

- Fungsi aktivasi adalah sigmoid logistik yaitu :

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

- Dimana input jaringan untuk setiap unit :

$$y\_in_j = b_j + \sum_i x_i w_{ij}$$

Rumusan yg sama juga dapat diterapkan pada lapisan X

Aplikasi

Contoh 7.1 : Jaringan BAM untuk mengelompokkan huruf-huruf dgn kode biner sederhana. Misal 2 huruf yg dinyatakan dgn pola bipolar :

. # .	. # #
# . #	# . .
# # #	# . .
# . #	# . .
# #	. # #
(-1,1)	(1,1)

matrik bobot untuk menyimpan kedua pola tsb dinyatakan dgn perkalian dari vektor pola pelatihan yaitu :

vektor “A” → (-1,1)

(-1 1 -1,1 -1 1, 1 1 1, 1 -1 1,1 -1 1) →(-1,1)

$$\begin{bmatrix}-1\\1\\-1\\1\\-1\\1\\1\\1\\1\\1\\1\\-1\\1\\1\\-1\\1\end{bmatrix}\begin{bmatrix}-1 & 1\end{bmatrix}=\begin{bmatrix}1 & -1\\-1 & 1\\1 & -1\\-1 & 1\\1 & -1\\-1 & 1\\-1 & 1\\-1 & 1\\-1 & 1\\-1 & 1\\1 & -1\\-1 & 1\\-1 & 1\\1 & -1\\-1 & 1\end{bmatrix}$$

vektor “C” → (1,1)

(-1 1 1,1 -1 -1, 1 -1 -1, 1 -1 -1,-1 1 1) →(1,1)

$$\begin{bmatrix}-1\\1\\1\\1\\-1\\-1\\1\\-1\\-1\\1\\-1\\-1\\-1\\-1\\1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix}=\begin{bmatrix}-1 & -1\\1 & 1\\1 & 1\\1 & 1\\-1 & -1\\-1 & -1\\1 & 1\\-1 & -1\\-1 & -1\\1 & 1\\-1 & -1\\-1 & -1\\-1 & -1\\1 & 1\\1 & 1\end{bmatrix}$$

Jadi Bobot W untuk menyimpan kedua pola adalah :

$$w = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} -1 & -1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ -1 & -1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -2 \\ 0 & 2 \\ 2 & 0 \\ 0 & 2 \\ 0 & -2 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ -2 & 0 \\ 0 & 2 \\ 0 & -2 \\ -2 & 0 \\ -2 & 0 \\ 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Jika kita lakukan testing dgn vektor input “A”

(-1 1 -1,1 -1 1, 1 1 1, 1 -1 1,1 -1 1) w = (-14,16)→(-1,1)

Jika kita lakukan testing dgn vektor input “C”

(-1 1 1,1 -1 -1, 1 -1 -1, 1 -1 -1,-1 1 1) w = (14,18)→(1,1)

Untuk melihat bahwa jaringan bersifat 2 arah (bidirectional) maka vektor y kita perlakukan sbg input . Dalam hal ini sinyal akan dikirim dari lapisan Y ke X, shg vektor bobot adalah transpose dari W :

$$w^T = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & -2 & 0 & -2 & -2 & 0 & 0 & -2 & -2 & 2 & 0 \\ -2 & 2 & 0 & 2 & -2 & 0 & 2 & 0 & 0 & 2 & -2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

untuk vektor input yg berkaitan dgn “A”, yaitu (-1,1) :

$$(-1 \quad 1)W^T = (-1 \quad 1)\begin{bmatrix} 0 & 0 & 2 & 0 & 0 & -2 & 0 & -2 & -2 & 0 & 0 & -2 & -2 & 2 & 0 \\ -2 & 2 & 0 & 2 & -2 & 0 & 2 & 0 & 0 & 2 & -2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$= (-2 \quad 2 \quad -2 \quad 2 \quad -2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad -2 \quad 2 \quad 2 \quad -2 \quad 2) \rightarrow$$
$$(-1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad 1 \quad 1 \quad -1 \quad 1)$$

ini adalah pola “A”

Jika kita menginputkan vektor “C”, yaitu (1,1) :

$$(1 \quad 1)W^T = (1 \quad 1)\begin{bmatrix} 0 & 0 & 2 & 0 & 0 & -2 & 0 & -2 & -2 & 0 & 0 & -2 & -2 & 2 & 0 \\ -2 & 2 & 0 & 2 & -2 & 0 & 2 & 0 & 0 & 2 & -2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$= (-2 \quad 2 \quad 2 \quad 2 \quad -2 \quad -2 \quad 2 \quad -2 \quad -2 \quad 2 \quad -2 \quad -2 \quad -2 \quad 2 \quad 2) \rightarrow$$
$$(-1 \quad 1 \quad 1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1)$$

ini adalah pola “C”