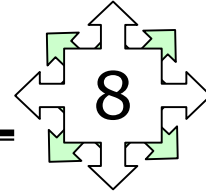


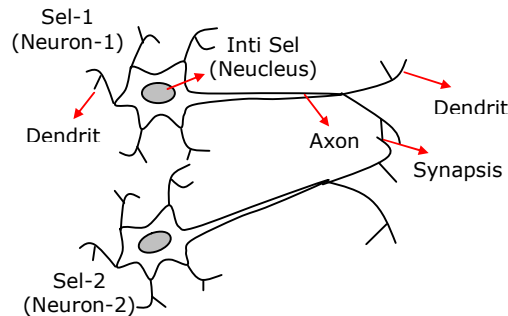
# JARINGAN SYARAF TIRUAN



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

## 8.1 OTAK MANUSIA

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



Gambar 8.1 Susunan syaraf manusia.

Gambar 8.1 menunjukkan susunan syaraf pada manusia. Setiap sel syaraf (neuron) akan memiliki satu inti sel, inti sel ini nanti yang akan bertugas untuk melakukan pemrosesan informasi. Informasi yang datang akan diterima oleh dendrit. Selain menerima informasi, dendrit juga menyertai axon sebagai keluaran dari suatu pemrosesan informasi. Informasi hasil olahan ini akan menjadi masukan bagi neuron lain yang mana antar dendrit kedua sel tersebut dipertemukan dengan synapsis. Informasi yang dikirimkan antar neuron ini berupa rangsangan yang dilewatkan melalui dendrit. Informasi yang datang dan diterima oleh dendrit akan dijumlahkan dan dikirim melalui axon ke dendrit akhir yang bersentuhan dengan dendrit dari neuron yang lain. Informasi ini akan diterima oleh neuron lain jika memenuhi batasan tertentu, yang sering dikenal dengan nama nilai ambang (*threshold*). Pada kasus ini, neuron tersebut dikatakan teraktivasi. Hubungan antar neuron terjadi secara adaptif, artinya struktur hubungan tersebut terjadi secara dinamis. Otak manusia selalu memiliki kemampuan untuk belajar dengan melakukan adaptasi.

## 8.2 SEJARAH

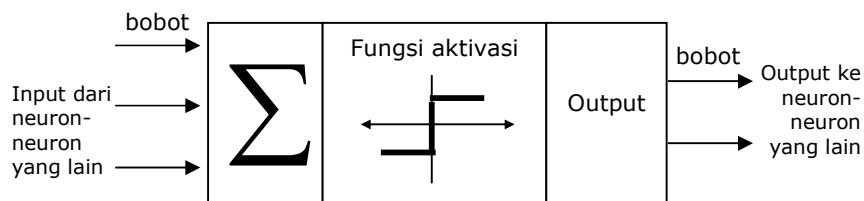
Mulai dari ditemukannya, jaringan syaraf tiruan telah mengalami tahap-tahap perkembangan, antara lain:

- \* Pada tahun 1940-an, para ilmuwan menemukan bahwa psikologi dari otak sama dengan mode pemrosesan yang dilakukan oleh peralatan komputer.

- ✳ Pada tahun 1943, McCulloch dan Pitts merancang model formal yang pertama kali sebagai perhitungan dasar neuron.
- ✳ Pada tahun 1949, Hebb menyatakan bahwa informasi dapat disimpan dalam koneksi-koneksi dan mengusulkan adanya skema pembelajaran untuk memperbaiki koneksi-koneksi antar neuron tersebut.
- ✳ Pada tahun 1954, Farley dan Clark mensetup model-model untuk relasi adaptif stimulus-respon dalam jaringan random.
- ✳ Pada tahun 1958, Rosenblatt mengembangkan konsep dasar tentang perceptron untuk klasifikasi pola.
- ✳ Pada tahun 1960, Widrow dan Hoff mengembangkan ADALINE untuk kendali adaptif dan pencocokan pola yang dilatih dengan aturan pembelajaran *Least Mean Square* (LMS).
- ✳ Pada tahun 1974, Werbos memperkenalkan algoritma *backpropagation* untuk melatih perceptron dengan banyak lapisan.
- ✳ Pada tahun 1975, Little dan Shaw menggambarkan jaringan syaraf dengan menggunakan model probabilistik.
- ✳ Pada tahun 1982, Kohonen mengembangkan metode pembelajaran jaringan syaraf yang tidak terawasi (*unsupervised learning*) untuk pemetaan.
- ✳ Pada tahun 1982, Grossberg mengembangkan teori jaringan yang diinspirasi oleh perkembangan psikologi. Bersama Carpenter, mereka mengenalkan sejumlah arsitektur jaringan, antara lain: Adaptive Resonance Theory (ART), ART2, dan ART3.
- ✳ Pada tahun 1982, Hopfield mengembangkan jaringan syaraf *recurrent* yang dapat digunakan untuk menyimpan informasi dan optimasi.
- ✳ Pada tahun 1985, algoritma pembelajaran dengan menggunakan mesin Boltzmann yang menggunakan model jaringan syaraf probabilistik mulai dikembangkan.
- ✳ Pada tahun 1987, Kosko mengembangkan jaringan *Adaptive Bidirectional Associative Memory* (BAM).
- ✳ Pada tahun 1988, mulai dikembangkan fungsi radial basis.

### 8.3 KOMPONEN JARINGAN SYARAF

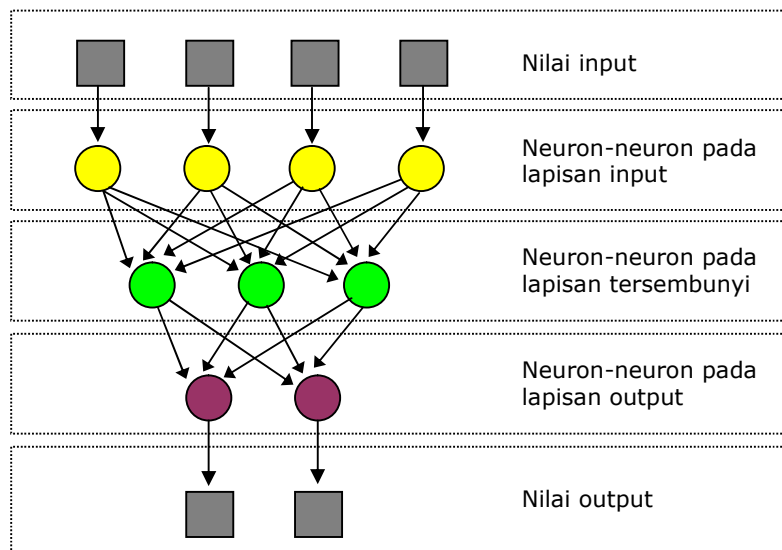
Ada beberapa tipe jaringan syaraf, namun demikian, hampir semuanya memiliki komponen-komponen yang sama. Seperti halnya otak manusia, jaringan syaraf juga terdiri-dari beberapa **neuron**, dan ada hubungan antara neuron-neuron tersebut. Neuron-neuron tersebut akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju ke neuron-neuron yang lain. Pada jaringan syaraf, hubungan ini dikenal dengan nama **bobot**. Informasi tersebut disimpan pada suatu nilai tertentu pada bobot tersebut. Gambar 8.2 menunjukkan struktur neuron pada jaringan syaraf.



Gambar 8.2 Struktur neuron jaringan syaraf.

Jika kita lihat, neuron buatan ini sebenarnya mirip dengan sel neuron biologis. Neuron-neuron buatan tersebut bekerja dengan cara yang sama pula dengan neuron-neuron biologis. Informasi (disebut dengan: **input**) akan dikirim ke neuron dengan bobot kedatangan tertentu. Input ini akan diproses oleh suatu fungsi perambatan yang akan menjumlahkan nilai-nilai semua bobot yang datang. Hasil penjumlahan ini kemudian akan dibandingkan dengan suatu nilai ambang (*threshold*) tertentu melalui **fungsi aktivasi** setiap neuron. Apabila input tersebut melewati suatu nilai ambang tertentu, maka neuron tersebut akan diaktifkan, tapi kalau tidak, maka neuron tersebut tidak akan diaktifkan. Apabila neuron tersebut diaktifkan, maka neuron tersebut akan mengirimkan **output** melalui bobot-bobot outputnya ke semua neuron yang berhubungan dengannya. Demikian seterusnya.

Pada jaringan syaraf, neuron-neuron akan dikumpulkan dalam lapisan-lapisan (*layer*) yang disebut dengan lapisan neuron (**neuron layers**). Biasanya neuron-neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya (kecuali lapisan input dan lapisan output). Informasi yang diberikan pada jaringan syaraf akan dirambatkan lapisan ke lapisan, mulai dari lapisan input sampai ke lapisan output melalui lapisan yang lainnya, yang sering dikenal dengan nama lapisan tersembunyi (**hidden layer**). Tergantung pada **algoritma pembelajarannya**, bisa jadi informasi tersebut akan dirambatkan secara mundur pada jaringan. Gambar 8.3 menunjukkan jaringan syaraf dengan 3 lapisan.



Gambar 8.3 Jaringan syaraf dengan 3 lapisan.

Gambar 8.3 bukanlah struktur umum jaringan syaraf. Beberapa jaringan syaraf ada juga yang tidak memiliki lapisan tersembunyi, dan ada juga jaringan syaraf dimana neuron-neuronnya disusun dalam bentuk matriks.

#### 8.4 ARSITEKTUR JARINGAN

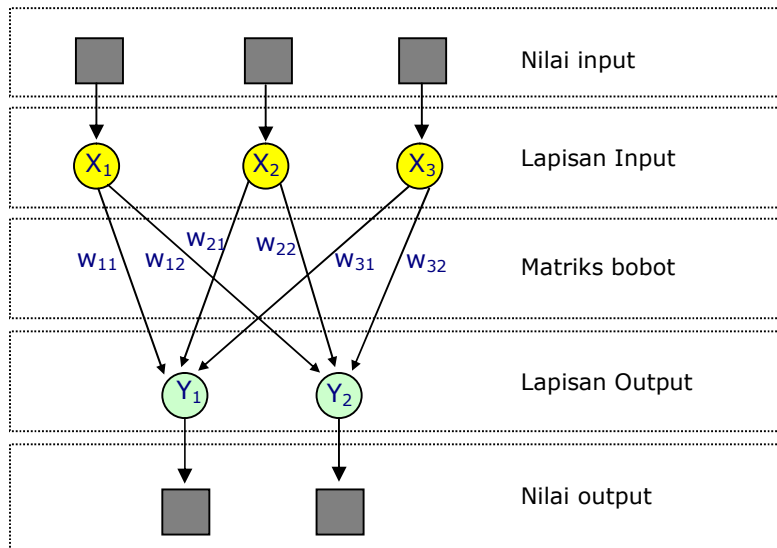
Seperti telah dijelaskan sebelumnya bahwa neuron-neuron dikelompokkan dalam lapisan-lapisan. Umumnya, neuron-neuron yang terletak pada lapisan yang sama akan memiliki keadaan yang sama. Faktor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Pada setiap lapisan yang sama, neuron-neuron akan memiliki fungsi aktivasi yang sama.

Apabila neuron-neuron dalam suatu lapisan (misalkan lapisan tersembunyi) akan dihubungkan dengan neuron-neuron pada lapisan yang lain (misalkan lapisan output), maka setiap neuron pada lapisan tersebut (misalkan lapisan tersembunyi) juga harus dihubungkan dengan setiap lapisan pada lapisan lainnya (misalkan lapisan output).

Ada beberapa arsitektur jaringan syaraf, antara lain:

#### a. Jaringan dengan lapisan tunggal (*single layer net*)

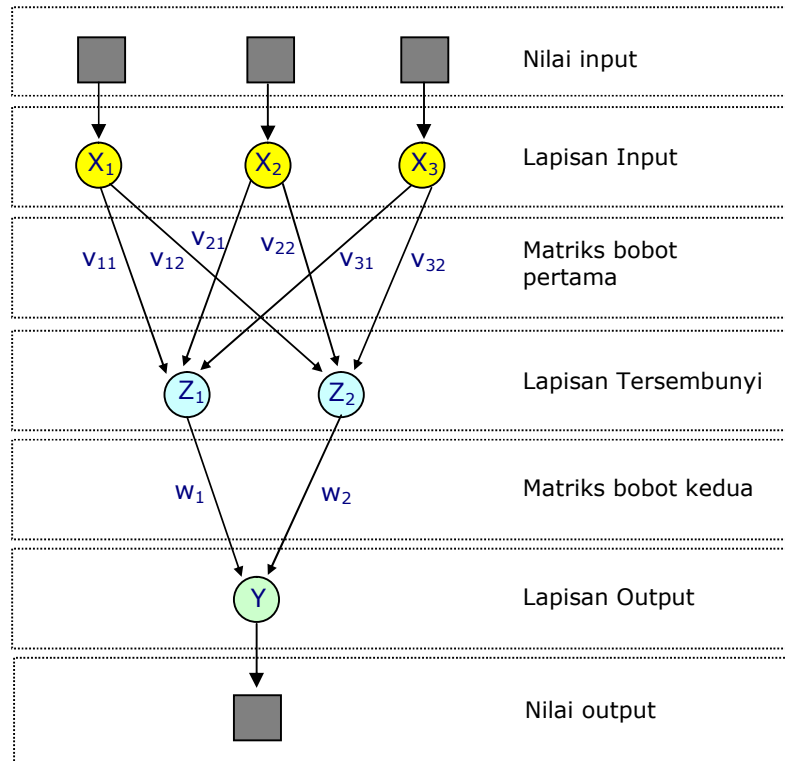
Jaringan dengan lapisan tunggal hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima input kemudian secara langsung akan mengolahnya menjadi output tanpa harus melalui lapisan tersembunyi (Gambar 8.4). Pada Gambar 8.4 tersebut, lapisan input memiliki 3 neuron, yaitu  $X_1$ ,  $X_2$  dan  $X_3$ . Sedangkan pada lapisan output memiliki 2 neuron yaitu  $Y_1$  dan  $Y_2$ . Neuron-neuron pada kedua lapisan saling berhubungan. Seberapa besar hubungan antara 2 neuron ditentukan oleh bobot yang bersesuaian. Semua unit input akan dihubungkan dengan setiap unit output.



Gambar 8.4 Jaringan syaraf dengan lapisan tunggal.

#### b. Jaringan dengan banyak lapisan (*multilayer net*)

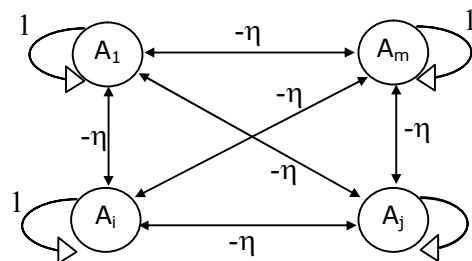
Jaringan dengan banyak lapisan memiliki 1 atau lebih lapisan yang terletak diantara lapisan input dan lapisan output (memiliki 1 atau lebih lapisan tersembunyi), seperti terlihat pada Gambar 8.5. Umumnya, ada lapisan bobot-bobot yang terletak antara 2 lapisan yang bersebelahan. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan dengan lapisan tunggal, tentu saja dengan pembelajaran yang lebih rumit. Namun demikian, pada banyak kasus, pembelajaran pada jaringan dengan banyak lapisan ini lebih sukses dalam menyelesaikan masalah.



Gambar 8.5 Jaringan syaraf dengan banyak lapisan.

### c. Jaringan dengan lapisan kompetitif (*competitive layer net*)

Umumnya, hubungan antar neuron pada lapisan kompetitif ini tidak diperlihatkan pada diagram arsitektur. Gambar 8.6 menunjukkan salah satu contoh arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot  $-\eta$ .



Gambar 8.6 Jaringan syaraf dengan lapisan kompetitif.

## 8.5 FUNGSI AKTIVASI

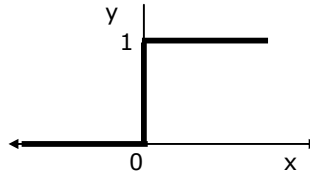
Ada beberapa fungsi aktivasi yang sering digunakan dalam jaringan syaraf tiruan, antara lain:

### a. Fungsi Undak Biner (*Hard Limit*)

Jaringan dengan lapisan tunggal sering menggunakan fungsi undak (*step function*) untuk mengkonversikan input dari suatu variabel yang bernilai kontinu ke suatu output biner (0 atau 1) (Gambar 8.7).

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

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



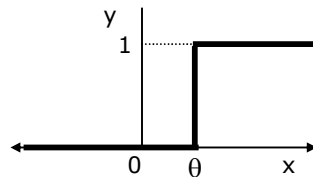
Gambar 8.7 Fungsi aktivasi: Undak Biner (*hard limit*).

### b. Fungsi Undak Biner (*Threshold*)

Fungsi undak biner dengan menggunakan nilai ambang sering juga disebut dengan nama fungsi nilai ambang (*threshold*) atau fungsi Heaviside (Gambar 8.8).

Fungsi undak biner (dengan nilai ambang  $\theta$ ) dirumuskan sebagai:

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



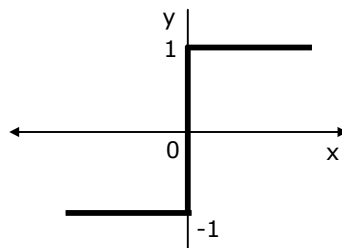
Gambar 8.8 Fungsi aktivasi: Undak Biner (*threshold*).

### c. Fungsi Bipolar (*Symetric Hard Limit*)

Fungsi bipolar sebenarnya hampir sama dengan fungsi undak biner, hanya saja output yang dihasilkan berupa 1, 0 atau -1 (Gambar 8.9).

Fungsi *Symetric Hard Limit* dirumuskan sebagai:

$$y = \begin{cases} 1, & \text{jika } x > 0 \\ 0, & \text{jika } x = 0 \\ -1, & \text{jika } x < 0 \end{cases}$$



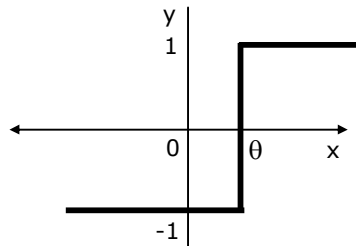
Gambar 8.9 Fungsi aktivasi: Bipolar (*symetric hard limit*).

**d. Fungsi Bipolar (dengan *threshold*)**

Fungsi bipolar sebenarnya hampir sama dengan fungsi undak biner dengan *threshold*, hanya saja output yang dihasilkan berupa 1, 0 atau -1 (Gambar 8.10).

Fungsi bipolar (dengan nilai ambang  $\theta$ ) dirumuskan sebagai:

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

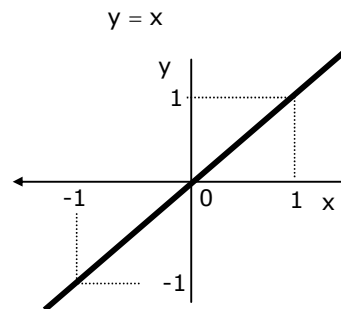


Gambar 8.10 Fungsi aktivasi: Bipolar (*threshold*).

**e. Fungsi Linear (identitas)**

Fungsi linear memiliki nilai output yang sama dengan nilai inputnya (Gambar 8.11).

Fungsi linear dirumuskan sebagai:



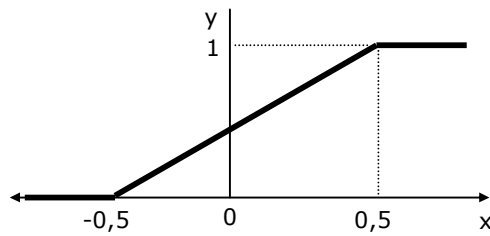
Gambar 8.11 Fungsi aktivasi: Linear (identitas).

**f. Fungsi Saturating Linear**

Fungsi ini akan bernilai 0 jika inputnya kurang dari  $-\frac{1}{2}$ , dan akan bernilai 1 jika inputnya lebih dari  $\frac{1}{2}$ . Sedangkan jika nilai input terletak antara  $-\frac{1}{2}$  dan  $\frac{1}{2}$ , maka outputnya akan bernilai sama dengan nilai input ditambah  $\frac{1}{2}$  (Gambar 8.12).

Fungsi *saturating linear* dirumuskan sebagai:

$$y = \begin{cases} 1; & \text{jika } x \geq 0,5 \\ x + 0,5; & \text{jika } -0,5 \leq x \leq 0,5 \\ 0; & \text{jika } x \leq -0,5 \end{cases}$$



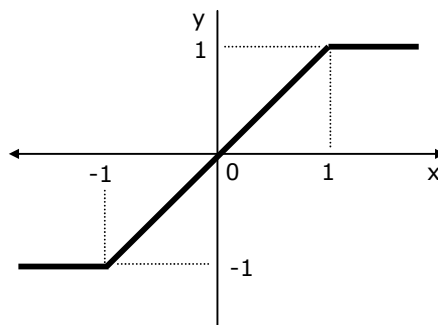
Gambar 8.12 Fungsi aktivasi: *Saturating Linear*

#### g. Fungsi Symetric Saturating Linear

Fungsi ini akan bernilai -1 jika inputnya kurang dari -1, dan akan bernilai 1 jika inputnya lebih dari 1. Sedangkan jika nilai input terletak antara -1 dan 1, maka outputnya akan bernilai sama dengan nilai inputnya (Gambar 8.13).

Fungsi *symetric saturating linear* dirumuskan sebagai:

$$y = \begin{cases} 1; & \text{jika } x \geq 1 \\ x; & \text{jika } -1 \leq x \leq 1 \\ -1; & \text{jika } x \leq -1 \end{cases}$$



Gambar 8.13 Fungsi aktivasi: *Symetric Saturating Linear*.

#### h. Fungsi Sigmoid Biner.

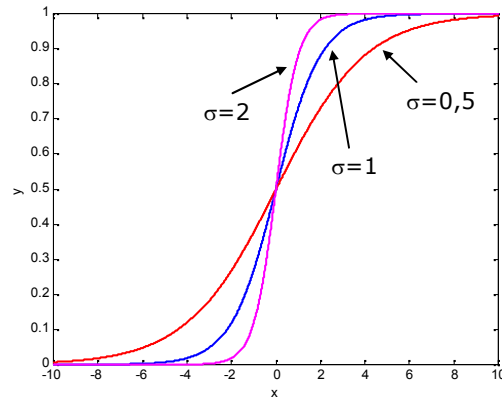
Fungsi ini digunakan untuk jaringan syaraf yang dilatih dengan menggunakan metode *backpropagation*. Fungsi sigmoid biner memiliki nilai pada range 0 sampai 1. Oleh karena itu, fungsi ini sering digunakan untuk jaringan syaraf yang membutuhkan nilai output yang terletak pada interval 0 sampai 1. Namun, fungsi ini bisa juga digunakan oleh jaringan syaraf yang nilai outputnya 0 atau 1 (Gambar 8.14).

Fungsi sigmoid biner dirumuskan sebagai:

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

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





Gambar 8.14 Fungsi aktivasi: *Sigmoid Biner*.

#### i. Fungsi Sigmoid Bipolar

Fungsi sigmoid bipolar hampir sama dengan fungsi sigmoid biner, hanya saja output dari fungsi ini memiliki range antara 1 sampai -1 (Gambar 8.15).

Fungsi sigmoid bipolar dirumuskan sebagai:

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

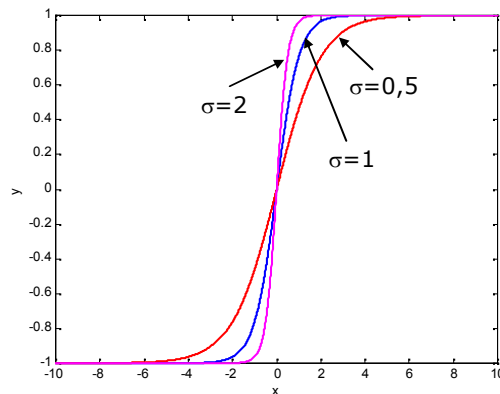
$$\text{dengan: } f'(x) = \frac{\sigma}{2} [1 + f(x)][1 - f(x)]$$

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

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

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

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



Gambar 8.15 Fungsi aktivasi: *Sigmoid Bipolar*.

## 8.6 PROSES PEMBELAJARAN

Pada otak manusia, informasi yang dilewatkan dari satu neuron ke neuron yang lainnya berbentuk rangsangan listrik melalui dendrit. Jika rangsangan tersebut diterima oleh suatu neuron, maka neuron tersebut akan membangkitkan output ke semua neuron yang berhubungan dengannya sampai informasi tersebut sampai ke tujuannya yaitu terjadinya suatu reaksi. Jika rangsangan yang diterima terlalu halus, maka output yang dibangkitkan oleh neuron tersebut tidak akan direspon. Tentu saja sangatlah sulit untuk memahami bagaimana otak manusia bisa belajar. Selama proses pembelajaran, terjadi perubahan yang cukup berarti pada bobot-bobot yang menghubungkan antar neuron. Apabila ada rangsangan yang sama dengan rangsangan yang telah diterima oleh neuron, maka neuron akan memberikan reaksi dengan cepat. Namun apabila kelak ada rangsangan yang berbeda dengan apa yang telah diterima oleh neuron, maka neuron akan segera beradaptasi untuk memberikan reaksi yang sesuai.

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

### a. Pembelajaran terawasi (*supervised learning*)

Metode pembelajaran pada jaringan syaraf disebut terawasi jika output yang diharapkan telah diketahui sebelumnya.

Contoh: andaikan kita memiliki jaringan syaraf yang akan digunakan untuk mengenali pasangan pola, misalkan pada operasi AND:

| Input | target |
|-------|--------|
| 0 0   | 0      |
| 0 1   | 0      |
| 1 0   | 0      |
| 1 1   | 1      |

Pada proses pembelajaran, satu pola input akan diberikan ke satu neuron pada lapisan input. Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan output. Lapisan output ini akan membangkitkan pola output yang nantinya akan dicocokkan dengan pola output targetnya. Apabila terjadi perbedaan antara pola output hasil pembelajaran dengan pola target, maka disini akan muncul error. Apabila nilai error ini masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi.

### **b. Pembelajaran tak terawasi (*unsupervised learning*)**

Pada metode pembelajaran yang tak terawasi ini tidak memerlukan target output. Pada metode ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai input yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk pengelompokan (klasifikasi) pola.

## **8.7 PEMBELAJARAN TERAWASI (SUPERVISED LEARNING)**

### **A. Hebb Rule**

Hebb rule adalah metode pembelajaran yang paling sederhana. Pada metode ini pembelajaran dilakukan dengan cara memperbaiki nilai bobot sedemikian rupa sehingga jika ada 2 neuron yang terhubung, dan keduanya pada kondisi 'hidup' (on) pada saat yang sama, maka bobot antara keduanya dinaikkan. Apabila data direpresentasikan secara bipolar, maka perbaikan bobotnya adalah:

$$w_i(\text{baru}) = w_i(\text{lama}) + x_i * y$$

dengan:

- $w_i$  : bobot data input ke-i;
- $x_i$  : input data ke-i.
- $y$  : output data.

Misalkan kita gunakan pasangan vektor input  $s$  dan vektor output sebagai pasangan vektor yang akan dilatih. Sedangkan vektor yang hendak digunakan untuk testing adalah vektor  $x$ .

Algoritma

0. Inisialisasi semua bobot:

$$w_{ij} = 0; \quad \text{dengan } i=1,2,\dots,n; \text{ dan } j=1,2,\dots,m.$$

1. Untuk setiap pasangan input-output ( $s$ - $t$ ), lakukan langkah-langkah sebagai berikut:

a. Set input dengan nilai sama dengan vektor input:

$$x_i = s_i; \quad (i=1,2,\dots,n)$$

b. Set output dengan nilai sama dengan vektor output:

$$y_j = t_j; \quad (j=1,2,\dots,m)$$

c. Perbaiki bobot:

$$w_{ij}(\text{baru}) = w_{ij}(\text{lama}) + x_i * y_j; \\ (i=1,2,\dots,n; \text{ dan } j=1,2,\dots,m)$$

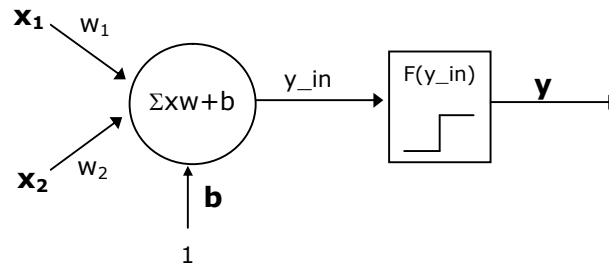
dengan catatan bahwa nilai bias selalu 1.

### **Contoh 8.1:**

Misalkan kita ingin membuat jaringan syaraf untuk melakukan pembelajaran terhadap fungsi OR dengan input dan target bipolar sebagai berikut:

| Input | Bias | Target |
|-------|------|--------|
| -1 -1 | 1    | -1     |
| -1 1  | 1    | 1      |
| 1 -1  | 1    | 1      |
| 1 1   | 1    | 1      |

Bobot awal dan bobot bias kita set=0.  
Arsitektur jaringan untuk contoh 8.1.



Gambar 8.16 Arsitektur jaringan contoh 8.1.

```
X =
-1 -1
-1 1
1 -1
1 1
T =
-1
1
1
1
```

```
Bobot awal =
w =
0
0
b = 0
```

Perubahan bobot:

```
Data ke-1
w1 = 0 + 1 = 1
w2 = 0 + 1 = 1
b = 0 - 1 = -1
```

```
Data ke-2
w1 = 1 - 1 = 0
w2 = 1 + 1 = 2
b = -1 + 1 = 0
```

```
Data ke-3
w1 = 0 + 1 = 1
w2 = 2 - 1 = 1
b = 0 + 1 = 1
```

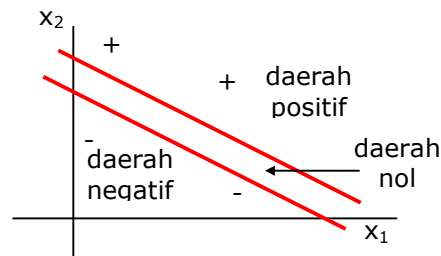
```
Data ke-4
w1 = 1 + 1 = 2
w2 = 1 + 1 = 2
b = 1 + 1 = 2
```

Kita bisa melakukan pengetesan terhadap salah satu data yang ada, misal kita ambil  $x = [-1 \ -1]$ .

$$Y = 2 + (-1 \cdot 2) + (-1 \cdot 2) = -2$$

## B. Perceptron

Perceptron juga termasuk salah satu bentuk jaringan syaraf yang sederhana. Perceptron biasanya digunakan untuk mengklasifikasikan suatu tipe pola tertentu yang sering dikenal dengan pemisahan secara linear. Pada dasarnya, perceptron pada jaringan syaraf dengan satu lapisan memiliki bobot yang bisa diatur dan suatu nilai ambang (*threshold*). Algoritma yang digunakan oleh aturan perceptron ini akan mengatur parameter-parameter bebasnya melalui proses pembelajaran. Nilai *threshold* ( $\theta$ ) pada fungsi aktivasi adalah non negatif. Fungsi aktivasi ini dibuat sedemikian rupa sehingga terjadi pembatasan antara daerah positif dan daerah negatif (Gambar 8.17).



Gambar 8.17 Pembatasan linear dengan perceptron.

Garis pemisah antara daerah positif dan daerah nol memiliki pertidaksamaan:

$$w_1x_1 + w_2x_2 + b > \theta$$

Sedangkan garis pemisah antara daerah negatif dengan daerah nol memiliki pertidaksamaan:

$$w_1x_1 + w_2x_2 + b < -\theta$$

Misalkan kita gunakan pasangan vektor input  $s$  dan vektor output sebagai pasangan vektor yang akan dilatih.

Algoritma:

0. Inisialisasi semua bobot dan bias:

(untuk sederhananya set semua bobot dan bobot bias sama dengan nol).

Set *learning rate*:  $\alpha$  ( $0 < \alpha \leq 1$ ).

(untuk sederhananya set sama dengan 1).

1. Selama kondisi berhenti bernilai *false*, lakukan langkah-langkah sebagai berikut:

(i). Untuk setiap pasangan pembelajaran  $s$ - $t$ , kerjakan:

a. Set input dengan nilai sama dengan vektor input:

$$x_i = s_i;$$

b. Hitung respon untuk unit output:

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

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

c. Perbaiki bobot dan bias jika terjadi *error*:

Jika  $y \neq t$  maka:

$$w_i(\text{baru}) = w_i(\text{lama}) + \alpha * t * x_i$$

$$b(\text{baru}) = b(\text{lama}) + \alpha * t$$

jika tidak, maka:

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

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

(ii). Tes kondisi berhenti: jika tidak terjadi perubahan bobot pada (i) maka kondisi berhenti TRUE, namun jika masih terjadi perubahan maka kondisi berhenti FALSE.

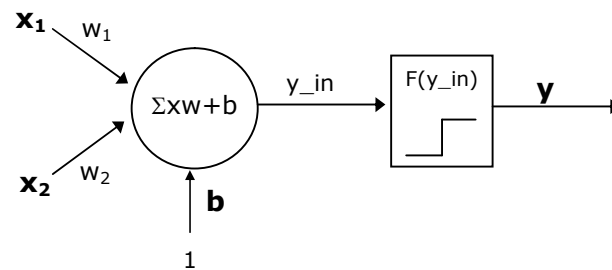
Algoritma di atas bisa digunakan baik untuk input biner maupun bipolar, dengan  $\theta$  tertentu, dan bias yang dapat diatur. Pada algoritma tersebut bobot-bobot yang diperbaiki hanyalah bobot-bobot yang berhubungan dengan input yang aktif ( $x_i \neq 0$ ) dan bobot-bobot yang tidak menghasilkan nilai  $y$  yang benar.

### Contoh 8.2:

Misalkan kita ingin membuat jaringan syaraf untuk melakukan pembelajaran terhadap fungsi AND dengan input biner dan target bipolar sebagai berikut:

| Input | Bias | Target |
|-------|------|--------|
| 1 1   | 1    | 1      |
| 1 0   | 1    | -1     |
| 0 1   | 1    | -1     |
| 0 0   | 1    | -1     |

Arsitektur jaringan untuk contoh 8.2 terlihat pada Gambar 8.18.



Gambar 8.18 Arsitektur jaringan contoh 8.2.

Bobot awal :  $w = [0, 0 \ 0, 0]$   
 Bobot bias awal :  $b = [0, 0]$   
 Learning rate (alfa): 0,8  
 Threshold (tetha) : 0,5

Epoh ke-1  
 -----  
 Data ke-1  
 $y_{in} = 0,0 + 0,0 + 0,0 = 0,0$   
 Hasil aktivasi = 0 ( $-0,5 < y_{in} < 0,5$ )  
 Target = 1  
 Bobot baru:  
 $w1 = 0,0 + 0,8 * 1,0 * 1,0 = 0,8$   
 $w2 = 0,0 + 0,8 * 1,0 * 1,0 = 0,8$   
 Bobot bias baru:  
 $b = 0,0 + 0,8 * 1,0 = 0,8$

Data ke-2  
 $y_{in} = 0,8 + 0,8 + 0,0 = 1,6$   
 Hasil aktivasi = 1 ( $y_{in} > 0,5$ )  
 Target = -1  
 Bobot baru:  
 $w1 = 0,8 + 0,8 * -1,0 * 1,0 = 0,0$   
 $w2 = 0,8 + 0,8 * -1,0 * 0,0 = 0,8$   
 Bobot bias baru:  
 $b = 0,8 + 0,8 * -1,0 = 0,0$

Data ke-3  
 $y_{in} = 0,0 + 0,0 + 0,8 = 0,8$   
 Hasil aktivasi = 1 ( $y_{in} > 0,5$ )  
 Target = -1  
 Bobot baru:  
 $w1 = 0,0 + 0,8 * -1,0 * 0,0 = 0,0$   
 $w2 = 0,8 + 0,8 * -1,0 * 1,0 = 0,0$   
 Bobot bias baru:  
 $b = 0,0 + 0,8 * -1,0 = -0,8$

Data ke-4  
 $y_{in} = -0,8 + 0,0 + 0,0 = -0,8$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Epoh ke-2  
 -----

Data ke-1  
 $y_{in} = -0,8 + 0,0 + 0,0 = -0,8$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = 1  
 Bobot baru:  
 $w1 = 0,0 + 0,8 * 1,0 * 1,0 = 0,8$   
 $w2 = 0,0 + 0,8 * 1,0 * 1,0 = 0,8$   
 Bobot bias baru:  
 $b = -0,8 + 0,8 * 1,0 = 0,0$

Data ke-2  
 $y_{in} = 0,0 + 0,8 + 0,0 = 0,8$   
 Hasil aktivasi = 1 ( $y_{in} > 0,5$ )  
 Target = -1  
 Bobot baru:  
 $w1 = 0,8 + 0,8 * -1,0 * 1,0 = 0,0$   
 $w2 = 0,8 + 0,8 * -1,0 * 0,0 = 0,8$   
 Bobot bias baru:  
 $b = 0,0 + 0,8 * -1,0 = -0,8$

Data ke-3  
 $y_{in} = -0,8 + 0,0 + 0,8 = 0,0$   
 Hasil aktivasi = 0 ( $-0,5 < y_{in} < 0,5$ )  
 Target = -1  
 Bobot baru:  
 $w1 = 0,0 + 0,8 * -1,0 * 0,0 = 0,0$   
 $w2 = 0,8 + 0,8 * -1,0 * 1,0 = 0,0$   
 Bobot bias baru:  
 $b = -0,8 + 0,8 * -1,0 = -1,6$

Data ke-4  
 $y_{in} = -1,6 + 0,0 + 0,0 = -1,6$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Epoh ke-3

-----

Data ke-1  
 $y_{in} = -1,6 + 0,0 + 0,0 = -1,6$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = 1  
 Bobot baru:  
 $w1 = 0,0 + 0,8 * 1,0 * 1,0 = 0,8$   
 $w2 = 0,0 + 0,8 * 1,0 * 1,0 = 0,8$   
 Bobot bias baru:  
 $b = -1,6 + 0,8 * 1,0 = -0,8$

Data ke-2  
 $y_{in} = -0,8 + 0,8 + 0,0 = 0,0$   
 Hasil aktivasi = 0 ( $-0,5 < y_{in} < 0,5$ )  
 Target = -1  
 Bobot baru:  
 $w1 = 0,8 + 0,8 * -1,0 * 1,0 = 0,0$   
 $w2 = 0,8 + 0,8 * -1,0 * 0,0 = 0,8$   
 Bobot bias baru:  
 $b = -0,8 + 0,8 * -1,0 = -1,6$

Data ke-3  
 $y_{in} = -1,6 + 0,0 + 0,8 = -0,8$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Data ke-4  
 $y_{in} = -1,6 + 0,0 + 0,0 = -1,6$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Epoh ke-4

-----

Data ke-1  
 $y_{in} = -1,6 + 0,0 + 0,8 = -0,8$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = 1  
 Bobot baru:  
 $w1 = 0,0 + 0,8 * 1,0 * 1,0 = 0,8$   
 $w2 = 0,8 + 0,8 * 1,0 * 1,0 = 1,6$   
 Bobot bias baru:  
 $b = -1,6 + 0,8 * 1,0 = -0,8$



Data ke-2  
 $y_{in} = -0,8 + 0,8 + 0,0 = 0,0$   
 Hasil aktivasi = 0 ( $-0,5 < y_{in} < 0,5$ )  
 Target = -1  
 Bobot baru:  
 $w1 = 0,8 + 0,8 * -1,0 * 1,0 = 0,0$   
 $w2 = 1,6 + 0,8 * -1,0 * 0,0 = 1,6$   
 Bobot bias baru:  
 $b = -0,8 + 0,8 * -1,0 = -1,6$

Data ke-3  
 $y_{in} = -1,6 + 0,0 + 1,6 = 0,0$   
 Hasil aktivasi = 0 ( $-0,5 < y_{in} < 0,5$ )  
 Target = -1  
 Bobot baru:  
 $w1 = 0,0 + 0,8 * -1,0 * 0,0 = 0,0$   
 $w2 = 1,6 + 0,8 * -1,0 * 1,0 = 0,8$   
 Bobot bias baru:  
 $b = -1,6 + 0,8 * -1,0 = -2,4$

Data ke-4  
 $y_{in} = -2,4 + 0,0 + 0,0 = -2,4$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Epoh ke-5

-----  
 Data ke-1  
 $y_{in} = -2,4 + 0,0 + 0,8 = -1,6$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = 1  
 Bobot baru:  
 $w1 = 0,0 + 0,8 * 1,0 * 1,0 = 0,8$   
 $w2 = 0,8 + 0,8 * 1,0 * 1,0 = 1,6$   
 Bobot bias baru:  
 $b = -2,4 + 0,8 * 1,0 = -1,6$

Data ke-2  
 $y_{in} = -1,6 + 0,8 + 0,0 = -0,8$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Data ke-3  
 $y_{in} = -1,6 + 0,0 + 1,6 = -0,0$   
 Hasil aktivasi = 0 ( $-0,5 < y_{in} < 0,5$ )  
 Target = -1  
 Bobot baru:  
 $w1 = 0,8 + 0,8 * -1,0 * 0,0 = 0,8$   
 $w2 = 1,6 + 0,8 * -1,0 * 1,0 = 0,8$   
 Bobot bias baru:  
 $b = -1,6 + 0,8 * -1,0 = -2,4$

Data ke-4  
 $y_{in} = -2,4 + 0,0 + 0,0 = -2,4$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Epoh ke-6

-----

```

Data ke-1
y_in = -2,4 + 0,8 + 0,8 = -0,8
Hasil aktivasi = -1 (y_in < -0,5)
Target          = 1
Bobot baru:
w1 = 0,8 + 0,8 * 1,0 * 1,0 = 1,6
w2 = 0,8 + 0,8 * 1,0 * 1,0 = 1,6
Bobot bias baru:
b = -2,4 + 0,8 * 1,0 = -1,6

Data ke-2
y_in = -1,6 + 1,6 + 0,0 = -0,0
Hasil aktivasi = 0 (-0,5 < y_in < 0,5)
Target          = -1
Bobot baru:
w1 = 1,6 + 0,8 * -1,0 * 1,0 = 0,8
w2 = 1,6 + 0,8 * -1,0 * 0,0 = 1,6
Bobot bias baru:
b = -1,6 + 0,8 * -1,0 = -2,4

Data ke-3
y_in = -2,4 + 0,0 + 1,6 = -0,8
Hasil aktivasi = -1 (y_in < -0,5)
Target          = -1

Data ke-4
y_in = -2,4 + 0,0 + 0,0 = -2,4
Hasil aktivasi = -1 (y_in < -0,5)
Target          = -1

Epoh ke-7
-----
Data ke-1
y_in = -2,4 + 0,8 + 1,6 = -0,0
Hasil aktivasi = 0 (-0,5 < y_in < 0,5)
Target          = 1
Bobot baru:
w1 = 0,8 + 0,8 * 1,0 * 1,0 = 1,6
w2 = 1,6 + 0,8 * 1,0 * 1,0 = 2,4
Bobot bias baru:
b = -2,4 + 0,8 * 1,0 = -1,6

Data ke-2
y_in = -1,6 + 1,6 + 0,0 = -0,0
Hasil aktivasi = 0 (-0,5 < y_in < 0,5)
Target          = -1
Bobot baru:
w1 = 1,6 + 0,8 * -1,0 * 1,0 = 0,8
w2 = 2,4 + 0,8 * -1,0 * 0,0 = 2,4
Bobot bias baru:
b = -1,6 + 0,8 * -1,0 = -2,4

Data ke-3
y_in = -2,4 + 0,0 + 2,4 = 0,0
Hasil aktivasi = 0 (-0,5 < y_in < 0,5)
Target          = -1
Bobot baru:
w1 = 0,8 + 0,8 * -1,0 * 0,0 = 0,8
w2 = 2,4 + 0,8 * -1,0 * 1,0 = 1,6
Bobot bias baru:

```

```

b = -2,4 + 0,8 * -1,0 = -3,2

Data ke-4
y_in = -3,2 + 0,0 + 0,0 = -3,2
Hasil aktivasi = -1 (y_in < -0,5)
Target          = -1

Epooh ke-8
-----
Data ke-1
y_in = -3,2 + 0,8 + 1,6 = -0,8
Hasil aktivasi = -1 (y_in < -0,5)
Target          = 1
Bobot baru:
w1 = 0,8 + 0,8 * 1,0 * 1,0 = 1,6
w2 = 1,6 + 0,8 * 1,0 * 1,0 = 2,4
Bobot bias baru:
b = -3,2 + 0,8 * 1,0 = -2,4

Data ke-2
y_in = -2,4 + 1,6 + 0,0 = -0,8
Hasil aktivasi = -1 (y_in < -0,5)
Target          = -1

Data ke-3
y_in = -2,4 + 0,0 + 2,4 = 0,0
Hasil aktivasi = 0 (-0,5 < y_in < 0,5)
Target          = -1
Bobot baru:
w1 = 1,6 + 0,8 * -1,0 * 0,0 = 1,6
w2 = 2,4 + 0,8 * -1,0 * 1,0 = 1,6
Bobot bias baru:
b = -2,4 + 0,8 * -1,0 = -3,2

Data ke-4
y_in = -3,2 + 0,0 + 0,0 = -3,2
Hasil aktivasi = -1 (y_in < -0,5)
Target          = -1

Epooh ke-9
-----
Data ke-1
y_in = -3,2 + 1,6 + 1,6 = 0,0
Hasil aktivasi = 0 (-0,5 < y_in < 0,5)
Target          = 1
Bobot baru:
w1 = 1,6 + 0,8 * 1,0 * 1,0 = 2,4
w2 = 1,6 + 0,8 * 1,0 * 1,0 = 2,4
Bobot bias baru:
b = -3,2 + 0,8 * 1,0 = -2,4

Data ke-2
y_in = -2,4 + 2,4 + 0,0 = 0,0
Hasil aktivasi = 0 (-0,5 < y_in < 0,5)
Target          = -1
Bobot baru:
w1 = 2,4 + 0,8 * -1,0 * 1,0 = 1,6
w2 = 2,4 + 0,8 * -1,0 * 0,0 = 2,4
Bobot bias baru:
b = -2,4 + 0,8 * -1,0 = -3,2

```

Data ke-3  
 $y_{in} = -3,2 + 0,0 + 2,4 = -0,8$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Data ke-4  
 $y_{in} = -3,2 + 0,0 + 0,0 = -3,2$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Epohe ke-10

-----

Data ke-1  
 $y_{in} = -3,2 + 1,6 + 2,4 = 0,8$   
 Hasil aktivasi = 1 ( $y_{in} > 0,5$ )  
 Target = 1

Data ke-2  
 $y_{in} = -3,2 + 1,6 + 0,0 = -1,6$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Data ke-3  
 $y_{in} = -3,2 + 0,0 + 2,4 = -0,8$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Data ke-4  
 $y_{in} = -3,2 + 0,0 + 0,0 = -3,2$   
 Hasil aktivasi = -1 ( $y_{in} < -0,5$ )  
 Target = -1

Pada epohe ke-10 ini sudah tidak terjadi perubahan bobot, sehingga proses pembelajaran dihentikan. Hasil akhir diperoleh:

Nilai bobot,  $w_1 = 1,6$ ; dan  $w_2 = 2,4$ .  
 Bobot bias,  $b = -3,2$ .

Dengan demikian garis yang membatasi daerah positif dengan daerah nol memenuhi pertidaksamaan:

$$1,6 x_1 + 2,4 x_2 - 3,2 > 0,5$$

Sedangkan garis yang membatasi daerah negatif dengan daerah nol memenuhi pertidaksamaan:

$$1,6 x_1 + 2,4 x_2 - 3,2 < -0,5$$

### C. Delta Rule

Pada delta rule akan mengubah bobot yang menghubungkan antara jaringan input ke unit output ( $y_{in}$ ) dengan nilai target ( $t$ ). Hal ini dilakukan untuk meminimalkan error selama pelatihan pola. Delta rule untuk memperbaiki bobot ke- $i$  (untuk setiap pola) adalah:

$$\Delta w_i = \alpha(t - y_{in}) * x_i;$$

dengan:

$x$  = vektor input.

$y_{in}$  = input jaringan ke unit output  $Y$ .

$$y_{in} = \sum_{i=1}^n x_i * w_i$$

$t$  = target (output).

Nilai  $w$  baru diperoleh dari nilai  $w$  lama ditambah dengan  $\Delta w$ ,

$$w_i = w_i + \Delta w_i$$

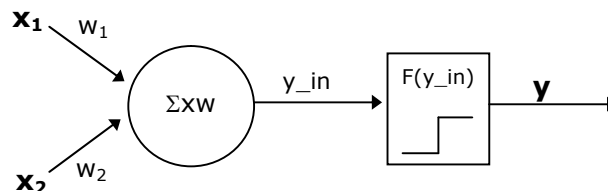
### Contoh 8.3:

Akan dibentuk jaringan syaraf untuk operasi OR (Tabel 8.1).

Tabel 8.1. Operasi OR

| $x_1$ | $x_2$ | Target (t)<br>$x_1$ OR $x_2$ |
|-------|-------|------------------------------|
| 0     | 0     | 0                            |
| 0     | 1     | 1                            |
| 1     | 0     | 1                            |
| 1     | 1     | 1                            |

Ada 2 input ( $x_1$  dan  $x_2$ ) dengan 1 output (target  $t$ ). Arsitektur jaringan terdiri-dari 1 lapisan input dan satu lapisan output (Gambar 8.19). Hanya ada 1 neuron pada lapisan output.



Gambar 8.19 Arsitektur jaringan pada Contoh 8.3.

Karena target yang diharapkan berbentuk biner, maka fungsi aktivasi yang digunakan adalah fungsi undak biner, dengan  $\theta = 0,5$ . Learning rate yang digunakan adalah  $\alpha = 0,2$ . Bobot awal yang dipilih  $w_1 = 0,1$  dan  $w_2 = 0,3$ . Disini nilai error ( $\delta$ ) diperoleh dari nilai  $t-y$ .

Pada Epoch pertama:

#### ▪ Data ke-1 :

- o  $x_{11} = 0$ ;  $x_{12} = 0$ ;  $t_1 = 0$ .
- o  $w_1 = 0,1$  dan  $w_2 = 0,3$ .
- o  $a_{11} = x_{11}w_1 + x_{12}w_2 = 0*0,1 + 0*0,3 = 0$ .
- o  $y_{11} = 0$ , karena  $a_{11} = 0 \leq 0,5$ .
- o  $\delta_{11} = t_1 - y_{11} = 0 - 0 = 0$ .
- o  $w_1 = w_1 + \alpha * x_{11} * \delta_{11} = 0,1 + 0,2*0*0 = 0,1$ .
- o  $w_2 = w_2 + \alpha * x_{12} * \delta_{11} = 0,3 + 0,2*0*0 = 0,3$ .

#### ▪ Data ke-2 :

- o  $x_{21} = 0$ ;  $x_{22} = 1$ ;  $t_2 = 1$ .

- o  $w_1 = 0,1$  dan  $w_2 = 0,3$ .
- o  $a_{12} = x_{21}w_1 + x_{22}w_2 = 0*0,1 + 1*0,3 = 0,3$ .
- o  $y_{12} = 0$ , karena  $a_{12} = 0,3 \leq 0,5$ .
- o  $\delta_{12} = t_2 - y_{12} = 1 - 0 = 1$ .
- o  $w_1 = w_1 + \alpha * x_{21} * \delta_{12} = 0,1 + 0,2*0*1 = 0,1$ .
- o  $w_2 = w_2 + \alpha * x_{22} * \delta_{12} = 0,3 + 0,2*1*1 = 0,5$ .

▪ Data ke-3 :

- o  $x_{31} = 1; x_{32} = 0; t_3 = 1$ .
- o  $w_1 = 0,1$  dan  $w_2 = 0,5$ .
- o  $a_{13} = x_{31}w_1 + x_{32}w_2 = 1*0,1 + 0*0,5 = 0,1$ .
- o  $y_{13} = 0$ , karena  $a_{13} = 0,1 \leq 0,5$ .
- o  $\delta_{13} = t_3 - y_{13} = 1 - 0 = 1$ .
- o  $w_1 = w_1 + \alpha * x_{31} * \delta_{13} = 0,1 + 0,2*1*1 = 0,3$ .
- o  $w_2 = w_2 + \alpha * x_{32} * \delta_{13} = 0,5 + 0,2*0*1 = 0,5$ .

▪ Data ke-4 :

- o  $x_{41} = 1; x_{42} = 1; t_4 = 1$ .
- o  $w_1 = 0,3$  dan  $w_2 = 0,5$ .
- o  $a_{14} = x_{41}w_1 + x_{42}w_2 = 1*0,3 + 1*0,5 = 0,8$ .
- o  $y_{14} = 1$ , karena  $a_{14} = 0,8 > 0,5$ .
- o  $\delta_{14} = t_4 - y_{14} = 1 - 1 = 0$ .
- o  $w_1 = w_1 + \alpha * x_{41} * \delta_{14} = 0,3 + 0,2*1*0 = 0,3$ .
- o  $w_2 = w_2 + \alpha * x_{42} * \delta_{14} = 0,5 + 0,2*0*0 = 0,5$ .

Pada Epoch kedua:

▪ Data ke-1 :

- o  $x_{11} = 0; x_{12} = 0; t_1 = 0$ .
- o  $w_1 = 0,3$  dan  $w_2 = 0,5$ .
- o  $a_{21} = x_{11}w_1 + x_{12}w_2 = 0*0,3 + 0*0,5 = 0$ .
- o  $y_{21} = 0$ , karena  $a_{21} = 0 \leq 0,5$ .
- o  $\delta_{21} = t_1 - y_{21} = 0 - 0 = 0$ .
- o  $w_1 = w_1 + \alpha * x_{11} * \delta_{21} = 0,3 + 0,2*0*0 = 0,3$ .
- o  $w_2 = w_2 + \alpha * x_{12} * \delta_{21} = 0,5 + 0,2*0*0 = 0,5$ .

Demikian seterusnya. Hasil akhir, dengan Err=0 tercapai pada Epoch ke-4. Hasil selengkapnya terlihat pada Tabel 8.2.

Tabel 8.2. Hasil pembelajaran sampai epoch ke-4.

| Epoch | $x_1$ | $x_2$ | $t$ | Bobot Awal |       | $a$ | $y$ | $\delta$ | Bobot Akhir |       |
|-------|-------|-------|-----|------------|-------|-----|-----|----------|-------------|-------|
|       |       |       |     | $w_1$      | $w_2$ |     |     |          | $w_1$       | $w_2$ |
| 1     | 0     | 0     | 0   | 0,1        | 0,3   | 0   | 0   | 0,0      | 0,1         | 0,3   |
|       | 0     | 1     | 1   | 0,1        | 0,3   | 0,3 | 0   | 1,0      | 0,1         | 0,5   |
|       | 1     | 0     | 1   | 0,1        | 0,5   | 0,1 | 0   | 1,0      | 0,3         | 0,5   |
|       | 1     | 1     | 1   | 0,3        | 0,5   | 0,8 | 1   | 0,0      | 0,3         | 0,5   |
| 2     | 0     | 0     | 0   | 0,3        | 0,5   | 0   | 0   | 0,0      | 0,3         | 0,5   |
|       | 0     | 1     | 1   | 0,3        | 0,5   | 0,5 | 0   | 1,0      | 0,3         | 0,7   |
|       | 1     | 0     | 1   | 0,3        | 0,7   | 0,3 | 0   | 1,0      | 0,5         | 0,7   |
|       | 1     | 1     | 1   | 0,5        | 0,7   | 1,2 | 1   | 0,0      | 0,5         | 0,7   |
| 3     | 0     | 0     | 0   | 0,5        | 0,7   | 0   | 0   | 0,0      | 0,5         | 0,7   |
|       | 0     | 1     | 1   | 0,5        | 0,7   | 0,7 | 1   | 0,0      | 0,5         | 0,7   |
|       | 1     | 0     | 1   | 0,5        | 0,7   | 0,5 | 0   | 1,0      | 0,7         | 0,7   |
|       | 1     | 1     | 1   | 0,7        | 0,7   | 1,4 | 1   | 0,0      | 0,7         | 0,7   |
| 4     | 0     | 0     | 0   | 0,7        | 0,7   | 0   | 0   | 0,0      | 0,7         | 0,7   |
|       | 0     | 1     | 1   | 0,7        | 0,7   | 0,7 | 1   | 0,0      | 0,7         | 0,7   |
|       | 1     | 0     | 1   | 0,7        | 0,7   | 0,7 | 1   | 0,0      | 0,7         | 0,7   |

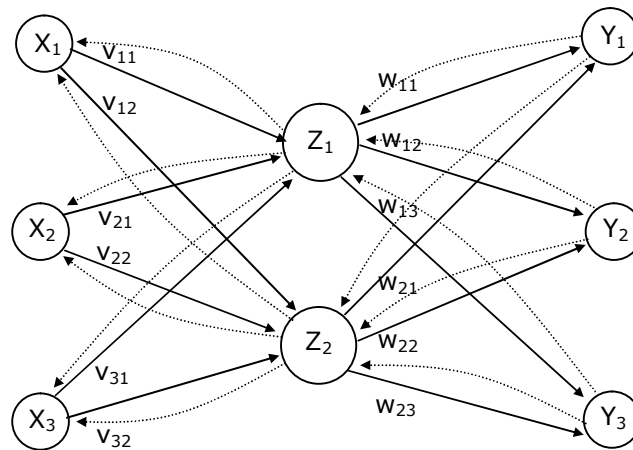
|  |   |   |   |     |     |     |   |     |     |     |
|--|---|---|---|-----|-----|-----|---|-----|-----|-----|
|  | 1 | 1 | 1 | 0,7 | 0,7 | 1,4 | 1 | 0,0 | 0,7 | 0,7 |
|--|---|---|---|-----|-----|-----|---|-----|-----|-----|

#### D. Backpropagation

*Backpropagation* merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyinya. Algoritma *backpropagation* menggunakan error output untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan error ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan menggunakan fungsi aktivasi sigmoid, yaitu:

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

Arsitektur jaringan *backpropagation* seperti terlihat pada Gambar 8.20.



Gambar 8.20 Arsitektur jaringan *backpropagation*.

Algoritma *backpropagation*:

- ⌚ Inisialisasi bobot (ambil bobot awal dengan nilai random yang cukup kecil).
- ⌚ Kerjakan langkah-langkah berikut selama kondisi berhenti bernilai FALSE:
  1. Untuk tiap-tiap pasangan elemen yang akan dilakukan pembelajaran, kerjakan:

*Feedforward*:

- a. Tiap-tiap unit input ( $X_i$ ,  $i=1,2,3,\dots,n$ ) menerima sinyal  $x_i$  dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada di atasnya (lapisan tersembunyi).
- b. Tiap-tiap unit tersembunyi ( $Z_j$ ,  $j=1,2,3,\dots,p$ ) menjumlahkan sinyal-sinyal input terbobot:

$$z_{in_j} = v_{0j} + \sum_{i=1}^n x_i v_{ij}$$

gunakan fungsi aktivasi untuk menghitung sinyal outputnya:

$$z_j = f(z\_in_j)$$

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

- c. Tiap-tiap unit output ( $Y_k$ ,  $k=1,2,3,\dots,m$ ) menjumlahkan sinyal-sinyal input terbobot.

$$y\_in_k = w_{0k} + \sum_{i=1}^p z_i w_{jk}$$

gunakan fungsi aktivasi untuk menghitung sinyal outputnya:

$$y_k = f(y\_in_k)$$

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

#### *Backpropagation*

- d. Tiap-tiap unit output ( $Y_k$ ,  $k=1,2,3,\dots,m$ ) menerima target pola yang berhubungan dengan pola input pembelajaran, hitung informasi errornya:

$$\delta_k = (t_k - y_k) f'(y\_in_k)$$

kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai  $w_{jk}$ ):

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

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

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

kirimkan  $\delta_k$  ini ke unit-unit yang ada di lapisan bawahnya.

- e. Tiap-tiap unit tersembunyi ( $Z_j$ ,  $j=1,2,3,\dots,p$ ) menjumlahkan delta inputnya (dari unit-unit yang berada pada lapisan di atasnya):

$$\delta\_in_j = \sum_{k=1}^m \delta_k w_{jk}$$

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

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

kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai  $v_{ij}$ ):

$$\Delta v_{jk} = \alpha \delta_j x_i$$

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



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

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

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

Tiap-tiap unit tersembunyi ( $Z_j$ ,  $j=1,2,3,\dots,p$ ) memperbaiki bias dan bobotnya ( $i=0,1,2,\dots,n$ ):

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

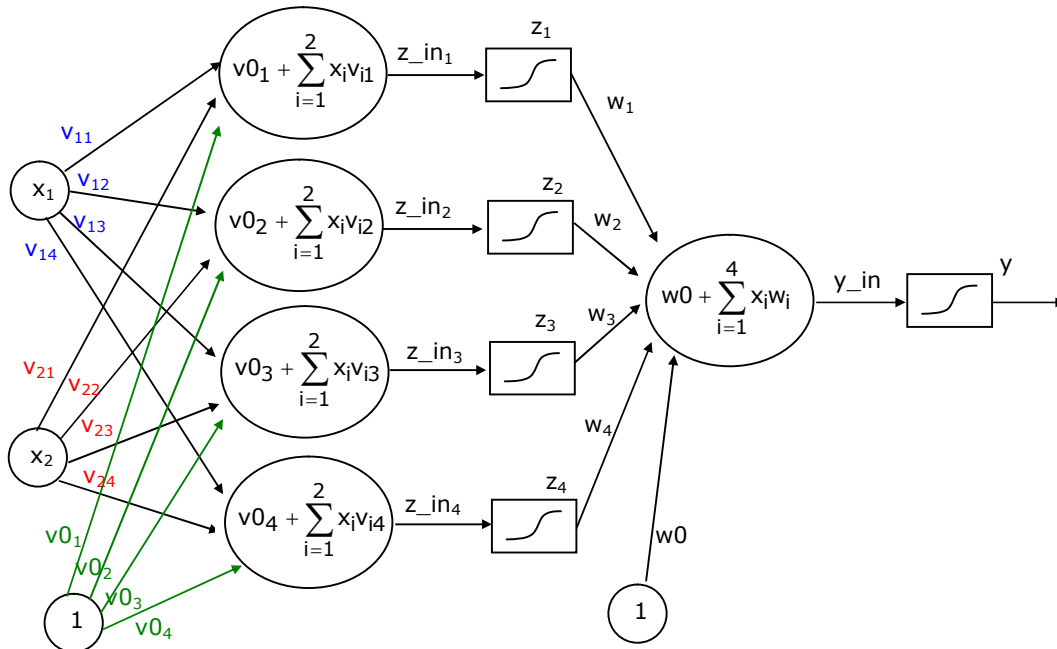
2. Tes kondisi berhenti,

#### Contoh 8.4:

Akan dibentuk jaringan syaraf untuk operasi XOR. Ada 2 input ( $X_1$  dan  $X_2$ ) dengan 1 output (target  $T$ ). Arsitektur jaringan (Gambar 8.21) terdiri-dari:

- ✱ 1 lapisan input,
- ✱ 1 lapisan tersembunyi dengan 4 neuron, fungsi aktivasi yang digunakan pada setiap neuron pada lapisan ini adalah fungsi sigmoid ( $z = \frac{1}{1 + e^{-z_{in}}}$ ).
- ✱ 1 lapisan output dengan 1 neuron, fungsi aktivasi yang digunakan pada neuron di lapisan ini adalah fungsi sigmoid ( $y = \frac{1}{1 + e^{-y_{in}}}$ ).

Bobot awal yang menghubungkan neuron-neuron pada lapisan input dan lapisan tersembunyi ( $v_{11}$ ,  $v_{12}$ ,  $v_{21}$ ,  $v_{22}$ ) dan bobot bias  $v_{01}$  dan  $v_{02}$  dipilih secara acak. Demikian pula bobot awal yang menghubungkan neuron-neuron pada lapisan tersembunyi dan lapisan output ( $w_1$ ,  $w_2$ ) dan bobot bias  $w_0$  juga dipilih secara acak.



Gambar 8.21 Arsitektur jaringan syaraf untuk *backpropagation*: Operasi XOR.

Pada inisialisasi ditetapkan:

```
X =  
  0   0  
  0   1  
  1   0  
  1   1
```

```
T =  
  0  
  1  
  1  
  0
```

Jumlah neuron pada input layer = 4  
Jumlah neuron pada hidden layer = 2  
Jumlah neuron pada output layer = 1  
Learning rate ( $\alpha$ ) = 0,02  
Maksimum Epoch = 500  
Target Error = 0,01

Bobot awal (ditentukan secara acak):

\* Bobot Awal input ke hidden

```
v =  
  0,9562   0,7762   0,1623   0,2886  
  0,1962   0,6133   0,0311   0,9711
```

\* Bobot Awal bias ke hidden

```
v0 =  
  0,7496   0,3796   0,7256   0,1628
```

\* Bobot Awal hidden ke output

```
w =  
  0,2280  
  0,9585  
  0,6799  
  0,0550
```

\* Bobot Awal bias ke output

```
w0 =  
  0,9505
```

Pembelajaran:

⚙ Epoch ke-1:

Data ke = 1

-----  
Operasi pada Hidden Layer --->

> Penjumlahan terbobot:

```
z_in1 = v01 + v11*x11 + v21*x12  
       = 0,7496 + 0,9562*0 + 0,1962*0  
       = 0,7496
```

```
z_in2 = v02 + v12*x11 + v22*x12  
       = 0,3796 + 0,7762*0 + 0,6133*0  
       = 0,3796
```

$$\begin{aligned}
 z_{in_3} &= v_{0_3} + v_{1_3} * x_{11} + v_{2_3} * x_{12} \\
 &= 0,7256 + 0,1623 * 0 + 0,0311 * 0 \\
 &= 0,7256
 \end{aligned}$$

$$\begin{aligned}
 z_{in_4} &= v_{0_4} + v_{1_4} * x_{11} + v_{2_4} * x_{12} \\
 &= 0,1628 + 0,2886 * 0 + 0,9711 * 0 \\
 &= 0,1628
 \end{aligned}$$

> Pengaktifan:

$$z_1 = \frac{1}{1 + e^{-0,7496}} = 0,6791$$

$$z_2 = \frac{1}{1 + e^{-0,3796}} = 0,5938$$

$$z_3 = \frac{1}{1 + e^{-0,7256}} = 0,6738$$

$$z_4 = \frac{2}{1 + e^{-0,1628}} - 1 = 0,5406$$

Operasi pada Output Layer --->

> Perkalian:

$$\begin{aligned}
 y_{in} &= w_0 + w_1 * z_1 + w_2 * z_2 + w_3 * z_3 + w_4 * z_4 \\
 &= 0,9505 + 0,2280 * 0,6791 + 0,9585 * 0,5938 + 0,6799 * 0,6738 + \\
 &\quad 0,0550 * 0,5406 \\
 &= 2,1623
 \end{aligned}$$

> Pengaktifan:

$$y = \frac{1}{1 + e^{-2,1623}} = 0,8968$$

$$\text{Error} = 0 - 0,8968 = -0,8968$$

$$\text{Jumlah Kuadrat Error} = (-0,8968)^2 = 0,80428$$

$$\delta = (T_1 - y) * \left( \frac{1}{1 + e^{-y_{in}}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-y_{in}}} \right) \right]$$

$$\delta = (0 - 0,8968) * \left( \frac{1}{1 + e^{-2,1623}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-2,1623}} \right) \right] = -0,08299$$

$$\Delta w_1 = \alpha * \delta * z_1$$

$$\Delta w_1 = 1 * (-0,08299) * 0,6791 = -0,05636$$

$$\Delta w_2 = \alpha * \delta * z_2$$

$$\Delta w_2 = 1 * (-0,08299) * 0,5936 = -0,04928$$

$$\Delta w_3 = \alpha * \delta * z_3$$

$$\Delta w_3 = 1 * (-0,08299) * 0,6738 = -0,05592$$

$$\Delta w_4 = \alpha * \delta * z_4$$

$$\Delta w_4 = 1 * (-0,08299) * 0,5406 = -0,04486$$

$$\Delta w_0 = \alpha * \delta$$

$$\Delta w_0 = 1 * (-0,08299) = -0,08299$$

$$\delta_{in1} = \delta * w_1 = -0,08299 * 0,2280 = -0,01893$$

$$\delta_{in2} = \delta * w_2 = -0,08299 * 0,9585 = -0,07955$$

$$\delta_{in3} = \delta * w_3 = -0,08299 * 0,6799 = -0,05642$$

$$\delta_{in4} = \delta * w_4 = -0,08299 * 0,0550 = -0,00456$$

$$\delta_1 = \delta_{in1} * \left( \frac{1}{1 + e^{-z_{in1}}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-z_{in1}}} \right) \right]$$

$$\delta_1 = -0,01893 * \left( \frac{1}{1 + e^{-0,6791}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-0,6791}} \right) \right] = -0,00412$$

$$\delta_2 = \delta_{in2} * \left( \frac{1}{1 + e^{-z_{in2}}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-z_{in2}}} \right) \right]$$

$$\delta_2 = -0,07955 * \left( \frac{1}{1 + e^{-0,5938}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-0,5938}} \right) \right] = -0,01919$$

$$\delta_3 = \delta_{in3} * \left( \frac{1}{1 + e^{-z_{in3}}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-z_{in3}}} \right) \right]$$

$$\delta_3 = -0,05642 * \left( \frac{1}{1 + e^{-0,6738}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-0,6738}} \right) \right] = -0,01240$$

$$\delta_4 = \delta_{in4} * \left( \frac{1}{1 + e^{-z_{in4}}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-z_{in4}}} \right) \right]$$

$$\delta_4 = -0,00456 * \left( \frac{1}{1 + e^{-0,5406}} \right) * \left[ 1 - \left( \frac{1}{1 + e^{-0,5406}} \right) \right] = -0,00113$$

$$\Delta v_{11} = \alpha * \delta_1 * x_{11} = 1 * (-0,00412) * 0 = 0$$

$$\text{demikian juga } \Delta v_{12} = \Delta v_{13} = \Delta v_{14} = \Delta v_{21} = \Delta v_{22} = \Delta v_{23} = \Delta v_{24}, = 0.$$

$$\Delta v_{01} = \alpha * \delta_1 = 1 * (-0,00412) = -0,00412$$

$$\Delta v_{02} = \alpha * \delta_2 = 1 * (-0,01919) = -0,01919$$

$$\Delta v_{03} = \alpha * \delta_3 = 1 * (-0,01240) = -0,01240$$

$$\Delta v_{04} = \alpha * \delta_4 = 1 * (-0,00113) = -0,00113$$

$$v_{11} = v_{11} + \Delta v_{11} = 0,9562 + 0 = 0,9562$$

$$v_{12} = v_{12} + \Delta v_{12} = 0,7762 + 0 = 0,7762$$

$$\begin{aligned}
v_{13} &= v_{13} + \Delta v_{13} = 0,1623 + 0 = 0,1623 \\
v_{14} &= v_{14} + \Delta v_{14} = 0,2886 + 0 = 0,2886 \\
v_{21} &= v_{21} + \Delta v_{21} = 0,1962 + 0 = 0,1962 \\
v_{22} &= v_{22} + \Delta v_{22} = 0,6133 + 0 = 0,6133 \\
v_{23} &= v_{23} + \Delta v_{23} = 0,0311 + 0 = 0,0311 \\
v_{24} &= v_{24} + \Delta v_{24} = 0,9711 + 0 = 0,9711 \\
v_{01} &= v_{01} + \Delta v_{01} = 0,7496 - 0,00412 = 0,7455 \\
v_{02} &= v_{02} + \Delta v_{02} = 0,3796 - 0,01919 = 0,3604 \\
v_{03} &= v_{03} + \Delta v_{03} = 0,7256 - 0,01240 = 0,7132 \\
v_{04} &= v_{04} + \Delta v_{04} = 0,1628 - 0,00113 = 0,1617
\end{aligned}$$

$$\begin{aligned}
w_1 &= w_1 + \Delta w_1 = 0,2280 - 0,05636 = 0,1717 \\
w_2 &= w_2 + \Delta w_2 = 0,9585 - 0,04928 = 0,9092 \\
w_3 &= w_3 + \Delta w_3 = 0,6799 - 0,05592 = 0,6239 \\
w_4 &= w_4 + \Delta w_4 = 0,0550 - 0,04486 = 0,0101 \\
w_0 &= w_0 + \Delta w_0 = 0,9505 - 0,08299 = 0,8675
\end{aligned}$$

Pada data kedua, juga dilakukan operasi-operasi yang sama dengan menggunakan bobot-bobot akhir hasil pengolahan data pertama ini sebagai bobot-bobot awalnya. Proses ini dilakukan secara berulang sampai pada maksimum epoh (1500) atau kuadrat error < target error (0,01).

Berikut adalah bobot akhir setelah epoh ke-898:

\* Bobot Akhir input ke hidden

$$\begin{aligned}
v &= \\
&\begin{matrix} 5,8716 & 3,6067 & 3,4877 & -0,0704 \\ -4,8532 & 2,8028 & -5,1943 & 0,7636 \end{matrix}
\end{aligned}$$

\* Bobot Akhir bias ke hidden

$$\begin{aligned}
v_0 &= \\
&\begin{matrix} 2,4618 & -0,3884 & -1,4258 & -0,6994 \end{matrix}
\end{aligned}$$

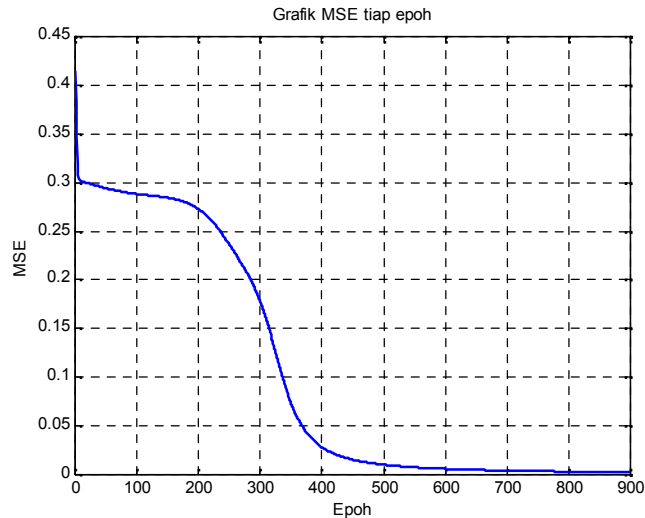
\* Bobot Akhir hidden ke output

$$\begin{aligned}
w &= \\
&\begin{matrix} -7,0997 \\ 3,5782 \\ 6,9212 \\ -0,7503 \end{matrix}
\end{aligned}$$

\* Bobot Akhir bias ke output

$$\begin{aligned}
w_0 &= \\
&0,6571
\end{aligned}$$

Gambar 8.22 menunjukkan grafik penurunan *Mean Square Error* (MSE) pada setiap epoh, mulai dari epoh pertama sampai epoh ke-898.



Gambar 8.22 Grafik penurunan *Mean Square Error* (MSE) sampai epoch ke-898.

Sekarang kita akan melakukan pengujian terhadap data:  $x=[0 \ 1]$

✱ Operasi pada hidden layer:

$$\begin{aligned} z_{in_1} &= v_{01} + (v_{11} * x_1) + (v_{21} * x_2) \\ &= 2,4618 + (5,8716 * 0) + (-4,8532 * 1) \\ &= -2,3914 \end{aligned}$$

$$z_1 = f(-2,3914) = \frac{1}{1 + e^{2,3914}} = 0,0838$$

$$\begin{aligned} z_{in_2} &= v_{02} + (v_{12} * x_1) + (v_{22} * x_2) \\ &= -0,3884 + (3,6067 * 0) + (2,8028 * 1) \\ &= 2,4144 \end{aligned}$$

$$z_2 = f(2,4144) = \frac{1}{1 + e^{-2,4144}} = 0,9179$$

$$\begin{aligned} z_{in_3} &= v_{03} + (v_{13} * x_1) + (v_{23} * x_2) \\ &= -1,4258 + (3,4877 * 0) + (-5,1943 * 1) \\ &= -6,6201 \end{aligned}$$

$$z_3 = f(-6,6201) = \frac{1}{1 + e^{6,6201}} = 0,0013$$

$$\begin{aligned} z_{in_4} &= v_{04} + (v_{14} * x_1) + (v_{24} * x_2) \\ &= -0,6994 + (-0,0704 * 0) + (0,7636 * 1) \\ &= 0,0642 \end{aligned}$$

$$z_4 = f(0,0642) = \frac{1}{1 + e^{-0,0642}} = 0,5160$$

✱ Operasi pada output layer:

$$\begin{aligned} y_{in} &= w_0 + z_1 * w_1 + z_2 * w_2 + z_3 * w_3 + z_4 * w_4 \\ &= 0,6571 + (0,0838 * -7,0997) + (0,9179 * 3,5782) + \end{aligned}$$

$$(0,0013*6,9212) + (0,5160*-0,7503) \\ = 2,9684$$

$$z = f(2,9684) = \frac{1}{1 + e^{-2,9684}} = 0,9511$$

Misalkan kita ambil *threshold* = 0,5; artinya jika nilai  $y \geq 0,5$  maka output yang diberikan adalah 1, namun jika nilai  $y < 0,5$  maka output yang diberikan adalah 0. Dengan demikian output dari  $x=[0 \ 1]$  adalah 1 (karena  $0,9511 > 0,5$ ). Sesuai dengan target yang diharapkan.

## E. Heteroassociative Memory

Jaringan syaraf *associative memory* adalah jaringan yang bobot-bobotnya ditentukan sedemikian rupa sehingga jaringan tersebut dapat menyimpan kumpulan pengelompokan pola. Masing-masing kelompok merupakan pasangan vektor ( $s(p), t(p)$ ) dengan  $p=1,2,\dots,P$ . Tiap-tiap vektor  $s(p)$  memiliki  $n$  komponen, dan tiap-tiap  $t(p)$  memiliki  $m$  komponen. Bobot-bobot tersebut dapat ditentukan dengan menggunakan Hebb rule atau delta rule. Jaringan ini nanti akhirnya akan mendapatkan vektor output yang sesuai dengan vektor inputnya ( $x$ ) yang merupakan salah satu vektor  $s(p)$  atau merupakan vektor lain di luar  $s(p)$ . Algoritma pembelajaran yang biasa digunakan oleh jaringan ini adalah Hebb rule dan delta rule.

Algoritma:

0. Inisialisasi bobot dengan menggunakan Hebb rule atau delta rule.

1. Untuk setiap vektor input, kerjakan:

⌚ Set input dengan nilai sama dengan vektor input:

⌚ Hitung input jaringan ke unit output:

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

⌚ Tentukan aktivasi dari setiap unit output:

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

(untuk target bipolar)

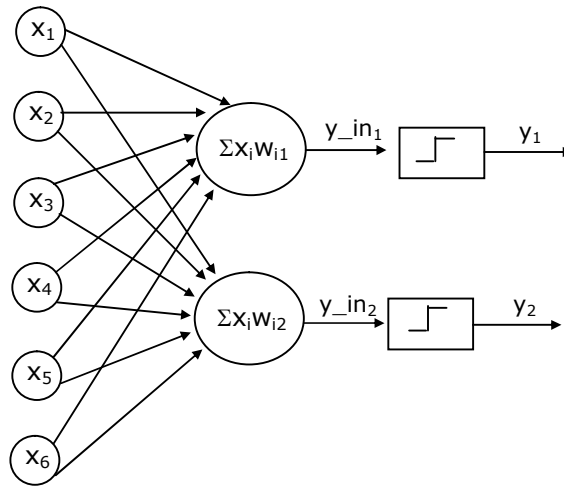
### Contoh 8.5:

Misalkan kita memiliki matriks  $P$  sebagai input yang hendak dilakukan pembelajaran, dengan target  $T$ . Fungsi aktivasi yang digunakan adalah fungsi bipolar.

| Input (s)       | Target (t) |
|-----------------|------------|
| -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   | -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 | 1  | 1 | 1 | -1 |

Arsitektur jaringan seperti terlihat pada Gambar 8.23.



Gambar 8.23 Arsitektur jaringan contoh 8.5.

Bobot awal (w)

|   |   |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |

Proses pembelajaran:

Perubahan bobot

Data ke-1

|     |   |      |   |       |   |       |
|-----|---|------|---|-------|---|-------|
| w11 | = | 0,00 | + | 1,00  | = | 1,00  |
| w12 | = | 0,00 | + | 1,00  | = | -1,00 |
| w21 | = | 0,00 | + | 1,00  | = | 1,00  |
| w22 | = | 0,00 | + | 1,00  | = | -1,00 |
| w31 | = | 0,00 | + | -1,00 | = | -1,00 |
| w32 | = | 0,00 | + | -1,00 | = | 1,00  |
| w41 | = | 0,00 | + | 1,00  | = | 1,00  |
| w42 | = | 0,00 | + | 1,00  | = | -1,00 |
| w51 | = | 0,00 | + | 1,00  | = | 1,00  |
| w52 | = | 0,00 | + | 1,00  | = | -1,00 |
| w61 | = | 0,00 | + | -1,00 | = | -1,00 |
| w62 | = | 0,00 | + | -1,00 | = | 1,00  |

Data ke-2

|     |   |       |   |       |   |       |
|-----|---|-------|---|-------|---|-------|
| w11 | = | 1,00  | + | 1,00  | = | 2,00  |
| w12 | = | -1,00 | + | 1,00  | = | -2,00 |
| w21 | = | 1,00  | + | 1,00  | = | 2,00  |
| w22 | = | -1,00 | + | 1,00  | = | -2,00 |
| w31 | = | -1,00 | + | -1,00 | = | -2,00 |
| w32 | = | 1,00  | + | -1,00 | = | 2,00  |
| w41 | = | 1,00  | + | 1,00  | = | 2,00  |
| w42 | = | -1,00 | + | 1,00  | = | -2,00 |
| w51 | = | 1,00  | + | -1,00 | = | 0,00  |
| w52 | = | -1,00 | + | -1,00 | = | 0,00  |



$w_{61} = -1,00 + 1,00 = 0,00$   
 $w_{62} = 1,00 + 1,00 = 0,00$

Data ke-3

$w_{11} = 2,00 + 1,00 = 3,00$   
 $w_{12} = -2,00 + 1,00 = -3,00$   
 $w_{21} = 2,00 + -1,00 = 1,00$   
 $w_{22} = -2,00 + -1,00 = -1,00$   
 $w_{31} = -2,00 + 1,00 = -1,00$   
 $w_{32} = 2,00 + 1,00 = 1,00$   
 $w_{41} = 2,00 + 1,00 = 3,00$   
 $w_{42} = -2,00 + 1,00 = -3,00$   
 $w_{51} = 0,00 + 1,00 = 1,00$   
 $w_{52} = 0,00 + 1,00 = -1,00$   
 $w_{61} = 0,00 + -1,00 = -1,00$   
 $w_{62} = 0,00 + -1,00 = 1,00$

Data ke-4

$w_{11} = 3,00 + -1,00 = 2,00$   
 $w_{12} = -3,00 + -1,00 = -2,00$   
 $w_{21} = 1,00 + 1,00 = 2,00$   
 $w_{22} = -1,00 + 1,00 = -2,00$   
 $w_{31} = -1,00 + -1,00 = -2,00$   
 $w_{32} = 1,00 + -1,00 = 2,00$   
 $w_{41} = 3,00 + 1,00 = 4,00$   
 $w_{42} = -3,00 + 1,00 = -4,00$   
 $w_{51} = 1,00 + -1,00 = 0,00$   
 $w_{52} = -1,00 + -1,00 = 0,00$   
 $w_{61} = -1,00 + -1,00 = -2,00$   
 $w_{62} = 1,00 + -1,00 = 2,00$

Data ke-5

$w_{11} = 2,00 + 1,00 = 1,00$   
 $w_{12} = -2,00 + 1,00 = -1,00$   
 $w_{21} = 2,00 + 1,00 = 1,00$   
 $w_{22} = -2,00 + 1,00 = -1,00$   
 $w_{31} = -2,00 + -1,00 = -1,00$   
 $w_{32} = 2,00 + -1,00 = 1,00$   
 $w_{41} = 4,00 + -1,00 = 5,00$   
 $w_{42} = -4,00 + -1,00 = -5,00$   
 $w_{51} = 0,00 + 1,00 = -1,00$   
 $w_{52} = 0,00 + 1,00 = 1,00$   
 $w_{61} = -2,00 + 1,00 = -3,00$   
 $w_{62} = 2,00 + 1,00 = 3,00$

Data ke-6

$w_{11} = 1,00 + 1,00 = 0,00$   
 $w_{12} = -1,00 + 1,00 = 0,00$   
 $w_{21} = 1,00 + -1,00 = 2,00$   
 $w_{22} = -1,00 + -1,00 = -2,00$   
 $w_{31} = -1,00 + 1,00 = -2,00$   
 $w_{32} = 1,00 + 1,00 = 2,00$   
 $w_{41} = 5,00 + -1,00 = 6,00$   
 $w_{42} = -5,00 + -1,00 = -6,00$   
 $w_{51} = -1,00 + 1,00 = -2,00$   
 $w_{52} = 1,00 + 1,00 = 2,00$   
 $w_{61} = -3,00 + 1,00 = -4,00$   
 $w_{62} = 3,00 + 1,00 = 4,00$

Data ke-7

$w_{11} = 0,00 + -1,00 = 1,00$   
 $w_{12} = 0,00 + -1,00 = -1,00$   
 $w_{21} = 2,00 + 1,00 = 1,00$   
 $w_{22} = -2,00 + 1,00 = -1,00$   
 $w_{31} = -2,00 + 1,00 = -3,00$   
 $w_{32} = 2,00 + 1,00 = 3,00$   
 $w_{41} = 6,00 + -1,00 = 7,00$   
 $w_{42} = -6,00 + -1,00 = -7,00$   
 $w_{51} = -2,00 + 1,00 = -3,00$   
 $w_{52} = 2,00 + 1,00 = 3,00$   
 $w_{61} = -4,00 + -1,00 = -3,00$   
 $w_{62} = 4,00 + -1,00 = 3,00$

Data ke-8

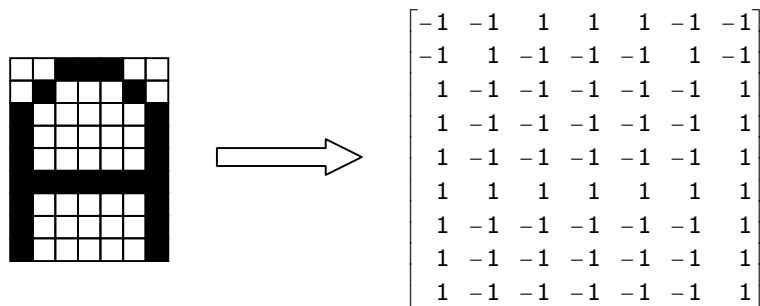
$w_{11} = 1,00 + 1,00 = 0,00$   
 $w_{12} = -1,00 + 1,00 = 0,00$   
 $w_{21} = 1,00 + -1,00 = 2,00$   
 $w_{22} = -1,00 + -1,00 = -2,00$   
 $w_{31} = -3,00 + -1,00 = -2,00$   
 $w_{32} = 3,00 + -1,00 = 2,00$   
 $w_{41} = 7,00 + -1,00 = 8,00$   
 $w_{42} = -7,00 + -1,00 = -8,00$   
 $w_{51} = -3,00 + -1,00 = -2,00$   
 $w_{52} = 3,00 + -1,00 = 2,00$   
 $w_{61} = -3,00 + -1,00 = -2,00$   
 $w_{62} = 3,00 + -1,00 = 2,00$

Testing:

- ⊕ Kita gunakan vektor input pertama:  $x = [-1 -1 1 -1 -1 1]$ 
  - \* Penjumlahan terbobot ke-1 ( $y_{in_1}$ ) = -12 -----> (-1)
  - \* Penjumlahan terbobot ke-2 ( $y_{in_2}$ ) = 12 -----> (1)
- ⊕ Vektor di luar vektor input:  $x = [-1 -1 1 1 -1 -1]$ 
  - \* Penjumlahan terbobot ke-1 ( $y_{in_1}$ ) = 8 -----> (1)
  - \* Penjumlahan terbobot ke-2 ( $y_{in_2}$ ) = -8 -----> (-1)

### Contoh 8.6:

Misalkan  $x$  adalah vektor dengan 63 komponen yang merepresentasikan pola karakter 2 dimensi (9x7) (Gambar 8.24). Sedangkan  $y$  adalah vektor dengan 15 komponen yang juga merepresentasikan pola karakter 2 dimensi (5x3) (Gambar 8.25).



Gambar 8.24 Huruf A, sebagai matriks 9x7.



```

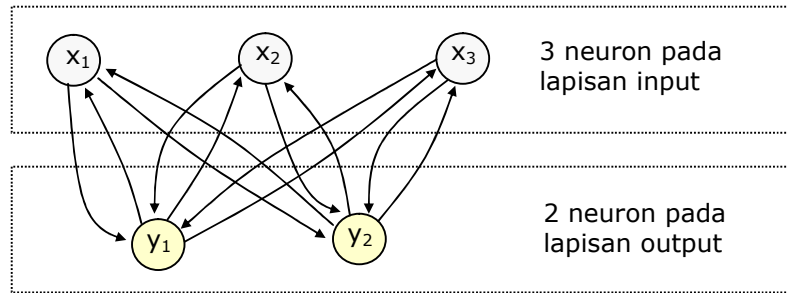
-1  3 -3  1 -1  1  1  1 -1  1 -1  1  1  1 -1
 1 -3  3 -1  1 -1 -1  1 -1  1 -1 -1 -1  1  1
 1  1 -1  3 -3  3  3  3  1  3 -3  3  3 -1  1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
 1  1 -1  3 -3  3  3  3  1  3 -3  3  3 -1  1
 1  1 -1  3 -3  3  3  3  1  3 -3  3  3 -1  1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
 1  1 -1 -1  1 -1 -1 -1 -3 -1  1 -1 -1  3 -3
-1 -1  1  1 -1  1  1  1  3  1 -1  1  1 -3  3
 1  1 -1  3 -3  3  3  3  1  3 -3  3  3 -1  1
 3 -1  1  1 -1  1  1  1 -1  1 -1  1  1  1 -1
 3 -1  1  1 -1  1  1  1 -1  1 -1  1  1  1 -1
 3 -1  1  1 -1  1  1  1 -1  1 -1  1  1  1 -1
 3 -1  1  1 -1  1  1  1 -1  1 -1  1  1  1 -1
 1 -3  3 -1  1 -1 -1 -1  1 -1  1 -1 -1 -1  1
-1 -1  1  1 -1  1  1  1  3  1 -1  1  1 -3  3
 1  1 -1  3 -3  3  3  3  1  3 -3  3  3 -1  1
-3  1 -1 -1  1 -1 -1 -1  1 -1  1 -1 -1 -1  1
-3  1 -1 -1  1 -1 -1 -1  1 -1  1 -1 -1 -1  1
-3  1 -1 -1  1 -1 -1 -1  1 -1  1 -1 -1 -1  1
-3  1 -1 -1  1 -1 -1 -1  1 -1  1 -1 -1 -1  1
-1  3 -3  1 -1  1  1  1 -1  1 -1  1  1  1 -1
-1 -1  1  1 -1  1  1  1  3  1 -1  1  1 -3  3
 1  1 -1  3 -3  3  3  3  1  3 -3  3  3 -1  1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
 1  1 -1  3 -3  3  3  3  1  3 -3  3  3 -1  1
 1  1 -1  3 -3  3  3  3  1  3 -3  3  3 -1  1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
 1  1 -1 -1  1 -1 -1 -1 -3 -1  1 -1 -1  3 -3
-1 -1  1  1 -1  1  1  1  3  1 -1  1  1 -3  3
 1  1 -1  3 -3  3  3  3  1  3 -3  3  3 -1  1
 1  1 -1 -1  1 -1 -1 -1 -3 -1  1 -1 -1  3 -3
 1  1 -1 -1  1 -1 -1 -1 -3 -1  1 -1 -1  3 -3
 1  1 -1 -1  1 -1 -1 -1 -3 -1  1 -1 -1  3 -3
 1  1 -1 -1  1 -1 -1 -1 -3 -1  1 -1 -1  3 -3
-1 -1  1 -3  3 -3 -3 -3 -1 -3  3 -3 -3  1 -1
-1 -1  1  1 -1  1  1  1  3  1 -1  1  1 -3  3

```

## F. Bidirectional Associative Memory (BAM)

*Bidirectional Associative Memory* (BAM) adalah model jaringan syaraf yang memiliki 2 lapisan dan terhubung penuh dari satu lapisan ke lapisan yang lainnya. Pada jaringan ini dimungkinkan adanya hubungan timbal balik antara lapisan input dan lapisan output. Namun demikian, bobot yang menghubungkan antara satu neuron (A) di satu lapisan dengan neuron (B) di lapisan lainnya akan sama dengan bobot yang menghubungkan neuron (B) ke neuron (A). Bisa

dikatakan bahwa, matriks bobot yang menghubungkan neuron-neuron pada lapisan output ke lapisan input sama dengan transpose matriks bobot neuron-neuron yang menghubungkan lapisan input ke lapisan output. Arsitektur jaringan untuk 3 neuron pada lapisan input dan 2 neuron pada lapisan output seperti terlihat pada Gambar 8.27.



Gambar 8.27 Arsitektur jaringan *Bidirectional Associative Memory*.

(i) BAM Diskret.

Pada BAM diskret ada 2 kemungkinan tipe data, yaitu biner dan bipolar. Matriks bobot awal dibuat sedemikian rupa sehingga dapat menyimpan pasangan vektor input dan vektor output  $s(p)$ - $t(p)$ , dengan  $p=1,2,3,\dots,P$ .

\* Untuk vektor input biner, matriks bobot ditentukan sebagai:

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

Sedangkan fungsi aktivasi yang digunakan adalah:

▪ Untuk lapisan output:

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

▪ Untuk lapisan input:

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

\* Sedangkan untuk vektor input bipolar, matriks bobot ditentukan sebagai:

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

Sedangkan fungsi aktivasi yang digunakan adalah:

▪ Untuk lapisan output:

$$y_j = \begin{cases} 1; & \text{jika } y\_in_j > \theta \\ y_j; & \text{jika } y\_in_j = \theta \\ -1; & \text{jika } y\_in_j < \theta \end{cases}$$

- Untuk lapisan input:

$$x_i = \begin{cases} 1; & \text{jika } x_{in_i} > \theta \\ x_i; & \text{jika } x_{in_i} = \theta \\ -1; & \text{jika } x_{in_i} < \theta \end{cases}$$

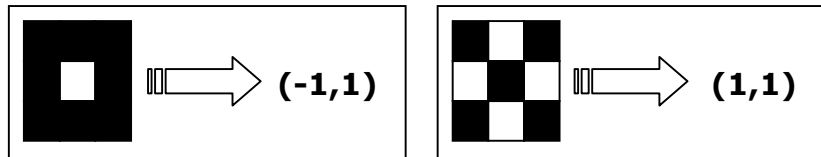
Dengan catatan bahwa input input hasil olahan pada jaringan ( $x_{in_i}$  atau  $y_{in_j}$ ) sama dengan nilai *thresholdnya* ( $\theta$ ), maka fungsi aktivasi akan menghasilkan nilai sama dengan nilai sebelumnya.

(ii) BAM kontinu

BAM kontinu akan mentransformasikan input secara lebih halus dan kontinu ke kawasan output dengan nilai yang terletak pada range  $[0,1]$ . Fungsi aktivasi yang digunakan adalah fungsi sigmoid.

#### Contoh 8.7:

Misalkan kita memiliki 2 matriks 3x3 yang mewakili bilangan huruf O dan X. Tiap matriks berhubungan dengan kode bipolar seperti terlihat pada Gambar 8.28.



Gambar 8.28 Matriks 3x3 untuk BAM.

Kita bisa membawa tiap-tiap matriks menjadi satu bentuk vektor dengan elemen-elemennya berupa bilangan biner -1 atau 1, sebagai berikut:

O : 1 1 1 1 -1 1 1 1 1 ---> -1 1  
X : 1 -1 1 -1 1 -1 1 -1 1 ---> 1 1

Matriks bobot:

$$W(\text{untuk } p=1) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$W(\text{untuk } p=2) = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \end{bmatrix}$$

$$W = w_{(\text{untuk } p=1)} + w_{(\text{untuk } p=2)}$$

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

Matriks bobot tersebut menghubungkan antara neuron-neuron di lapisan input ke neuron-neuron yang ada di lapisan output. Sedangkan matriks bobot yang menghubungkan antara neuron-neuron di lapisan output ke neuron-neuron yang ada di lapisan input adalah  $W^T$ .

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

Kita bisa menguji bobot tersebut. Misalkan kita cobakan vektor input pertama yang mewakili Huruf O, maka output yang diperoleh adalah:

$$y_{in1} = x_1 * W = \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 2 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 2 & 0 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} -10 & 8 \end{bmatrix}$$

Karena  $(y_{in1}(1)=-10 < 0)$ , maka  $y_1(1)=-1$  dan  $(y_{in1}(2)=8 > 0)$ , maka  $y_1(2)=1$ , maka nilai  $y_1 = [-1 \ 1]$ , sama dengan target yang diharapkan.

Untuk vektor input kedua yang mewakili Huruf X, maka output yang diperoleh adalah:

$$y_{in1} = x_1 * W = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 2 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 2 & 0 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 10 & 8 \end{bmatrix}$$

Karena  $(y_{in2}(1)=10 > 0)$ , maka  $y_2(1)=1$  dan  $(y_{in2}(2)=8 > 0)$ , maka  $y_2(2)=1$ , maka nilai  $y_2 = [1 \ 1]$ , sama dengan target yang diharapkan.

Sekarang apabila dibalik, y digunakan sebagai input untuk mendapatkan x. Misalkan kita cobakan vektor input pertama  $y_1 = [-1 \ 1]$ , maka output yang diperoleh adalah:

$$\begin{aligned} x\_in_1 &= y_1 * W^T = [-1 \ 1] * \begin{bmatrix} 0 & -2 & 0 & -2 & 2 & -2 & 0 & -2 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 2 \end{bmatrix} \\ &= [2 \ 2 \ 2 \ 2 \ -2 \ 2 \ 2 \ 2 \ 2] \Rightarrow [1 \ 1 \ 1 \ 1 \ -1 \ 1 \ 1 \ 1 \ 1] \end{aligned}$$

Outputnya sama dengan target, yaitu Huruf O.

Untuk vektor input kedua yaitu  $y_2 = [1 \ 1]$ , maka output yang diperoleh adalah:

$$\begin{aligned} x\_in_2 &= y_2 * W^T = [1 \ 1] * \begin{bmatrix} 0 & -2 & 0 & -2 & 2 & -2 & 0 & -2 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 2 \end{bmatrix} \\ &= [2 \ -2 \ 2 \ -2 \ 2 \ -2 \ 2 \ -2 \ 2] \Rightarrow [1 \ -1 \ 1 \ -1 \ 1 \ -1 \ 1 \ -1 \ 1] \end{aligned}$$

Outputnya sama dengan target, yaitu Huruf X.

Algoritma

0. Inisialisasi bobot (untuk menyimpan sekumpulan P vektor).

Inisialisasi semua aktivasi sama dengan 0.

1. Untuk tiap-tiap input, kerjakan:

a (1). Berikan input pola x ke lapisan X (kita set aktivasi lapisan X sebagai pola input)

a (2). Berikan input pola y ke lapisan Y (salah satu dari vektor input tersebut biasanya diset sebagai vektor nol)

b. Kerjakan jika aktivasi-aktivasi tersebut belum konvergen:

(i). Perbaiki setiap unit aktivasi di lapisan Y:

Hitung:

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

Hitung:

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

Berikan informasi tersebut ke lapisan X.

(ii). Perbaiki setiap unit aktivasi di lapisan X:

Hitung:

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

Hitung:

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

Berikan informasi tersebut ke lapisan Y.

(iii). Tes kekonvergenan. Jika vektor x dan y telah mencapai keadaan stabil, maka iterasi berhenti, jika tidak demikian lanjutkan iterasi.



## G. Learning Vector Quantization (LVQ)

*Learning Vector Quantization* (LVQ) adalah suatu metode untuk melakukan pembelajaran pada lapisan kompetitif yang terawasi. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor input. Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor-vektor input. Jika 2 vektor input mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor input tersebut ke dalam kelas yang sama.

Algoritma

1. Tetapkan:  $W$ ,  $MaxIter$ ,  $Eps$ ,  $\alpha$ .
2. Masukkan:
  - o input  $x(m,n)$ ;
  - o  $m$  = jumlah input,
  - o  $n$  = jumlah data
  - o  $target(1,n)$ .
3. Tetapkan kondisi awal:
  - o  $epoch=0$ ;
  - o  $err=1$ .
4. Kerjakan jika: ( $epoch < MaxIter$ ) atau ( $\alpha > eps$ )
  - a.  $epoch = epoch + 1$ ;
  - b. Kerjakan untuk  $i=1$  sampai  $n$ 
    - i. Tentukan  $J$  sedemikian hingga  $\|x-w_j\|$  minimum
    - ii. Perbaiki  $w_j$  dengan ketentuan:
      - o Jika  $T = C_j$  maka:
$$w_j(\text{baru}) = w_j(\text{lama}) + \alpha (x - w_j(\text{lama}))$$
      - o Jika  $T \neq C_j$  maka:
$$w_j(\text{baru}) = w_j(\text{lama}) - \alpha (x - w_j(\text{lama}))$$
  - c. Kurangi nilai  $\alpha$ .

### Contoh 8.8:

Misalkan diketahui 10 input vektor dalam 2 kelas sebagai berikut:

|     |                    |   |
|-----|--------------------|---|
| 1.  | (1, 0, 0, 0, 1, 0) | 1 |
| 2.  | (0, 1, 1, 1, 1, 0) | 2 |
| 3.  | (0, 0, 1, 0, 0, 1) | 1 |
| 4.  | (0, 0, 1, 0, 1, 0) | 1 |
| 5.  | (0, 1, 0, 0, 0, 1) | 1 |
| 6.  | (1, 0, 1, 0, 1, 1) | 1 |
| 7.  | (0, 0, 1, 1, 0, 0) | 2 |
| 8.  | (0, 1, 0, 1, 0, 0) | 2 |
| 9.  | (1, 0, 0, 1, 0, 1) | 2 |
| 10. | (0, 1, 1, 1, 1, 1) | 2 |

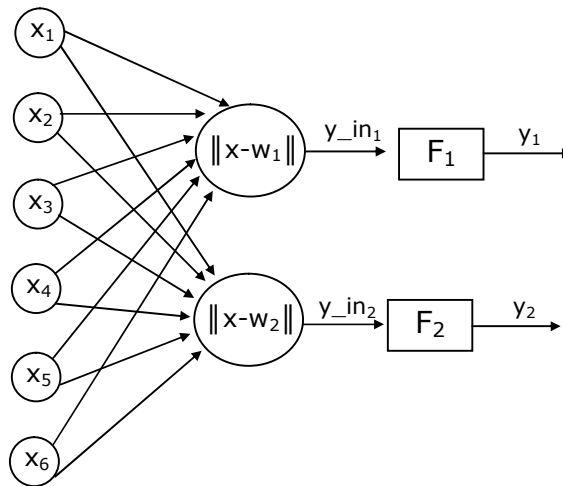
Dua input pertama akan dijadikan sebagai inisialisasi bobot:

|    | Vektor (w)         | Kelas |
|----|--------------------|-------|
| 1. | (1, 0, 0, 0, 1, 0) | 1     |
| 2. | (0, 1, 1, 1, 1, 0) | 2     |

Sedangkan 4 input sisanya:

|    | Vektor (x)         | Kelas |
|----|--------------------|-------|
| 1. | (0, 0, 1, 0, 0, 1) | 1     |
| 2. | (0, 0, 1, 0, 1, 0) | 1     |
| 3. | (0, 1, 0, 0, 0, 1) | 1     |
| 4. | (1, 0, 1, 0, 1, 1) | 1     |
| 5. | (0, 0, 1, 1, 0, 0) | 2     |
| 6. | (0, 1, 0, 1, 0, 0) | 2     |
| 7. | (1, 0, 0, 1, 0, 1) | 2     |
| 8. | (0, 1, 1, 1, 1, 1) | 2     |

akan digunakan sebagai data yang akan dilatih. Arsitektur jaringan syaraf pada contoh 8.8 ini seperti terlihat pada Gambar 8.29.



Gambar 8.29 Arsitektur jaringan untuk contoh 8.8.

Sebagai nilai awal dipilih learning rate ( $\alpha=0,05$ ), dengan pengurangan sebesar  $0,1*\alpha$ ; dan maksimum epoh (MaxEpoh=10).

Epoh ke-1:

⊕ Data ke-1: (0, 0, 1, 0, 0, 1)

Jarak pada:

$$\begin{aligned} \text{* bobot ke-1} &= \\ &= \sqrt{(0-1)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2 + (0-1)^2 + (1-0)^2} = 2,0. \end{aligned}$$

$$\begin{aligned} \text{* bobot ke-2} &= \\ &= \sqrt{(0-1)^2 + (0-1)^2 + (1-1)^2 + (0-1)^2 + (0-1)^2 + (1-0)^2} = 2,0. \end{aligned}$$

Jarak terkecil pada bobot ke-1

Target data ke-1= 1

Bobot ke- 1 baru:

$$w_{11} = w_{11} + \alpha * (x_{11} - w_{11}) = 1 + 0,05 * (0-1) = 0,9500;$$

$$w_{12} = w_{12} + \alpha * (x_{12} - w_{12}) = 0 + 0,05 * (0-0) = 0,0000;$$

$$\begin{aligned}
w_{13} &= w_{13} + \alpha^*(x_{13} - w_{13}) = 0 + 0,05*(1-0) = 0,0500; \\
w_{14} &= w_{14} + \alpha^*(x_{14} - w_{14}) = 0 + 0,05*(0-0) = 0,0000; \\
w_{15} &= w_{15} + \alpha^*(x_{15} - w_{15}) = 1 + 0,05*(0-1) = 0,9500; \\
w_{16} &= w_{16} + \alpha^*(x_{16} - w_{16}) = 0 + 0,05*(1-0) = 0,0500;
\end{aligned}$$

Jadi

$$w_1 = (0,9500 \quad 0,0000 \quad 0,0500 \quad 0,0000 \quad 0,9500 \quad 0,0500)$$

④ **Data ke-2:** (0, 0, 1, 0, 1, 0)

Jarak pada:

$$\begin{aligned}
&* \text{ bobot ke-1} = \\
&= \sqrt{(0-0,95)^2 + (0-0)^2 + (1-0,05)^2 + (0-0)^2 + (1-0,95)^2 + (0-0,05)^2} \\
&= 1,3454. \\
&* \text{ bobot ke-2} = \\
&= \sqrt{(0-0)^2 + (0-1)^2 + (1-1)^2 + (0-1)^2 + (1-1)^2 + (0-0)^2} \\
&= 1,4142.
\end{aligned}$$

Jarak terkecil pada bobot ke-1

Target data ke-2= 1

Bobot ke- 1 baru:

$$\begin{aligned}
w_{11} &= w_{11} + \alpha^*(x_{21} - w_{11}) = 0,95 + 0,05*(0-0,95) = 0,9025; \\
w_{12} &= w_{12} + \alpha^*(x_{22} - w_{12}) = 0,00 + 0,05*(0-0,00) = 0,0000; \\
w_{13} &= w_{13} + \alpha^*(x_{23} - w_{13}) = 0,05 + 0,05*(1-0,05) = 0,0975; \\
w_{14} &= w_{14} + \alpha^*(x_{24} - w_{14}) = 0,00 + 0,05*(0-0,00) = 0,0000; \\
w_{15} &= w_{15} + \alpha^*(x_{25} - w_{15}) = 0,95 + 0,05*(1-0,95) = 0,9525; \\
w_{16} &= w_{16} + \alpha^*(x_{26} - w_{16}) = 0,05 + 0,05*(0-0,05) = 0,0475;
\end{aligned}$$

Jadi

$$w_1 = (0,9025 \quad 0,0000 \quad 0,0975 \quad 0,0000 \quad 0,9525 \quad 0,0475)$$

④ **Data ke-3:** (0, 1, 0, 0, 0, 1)

Jarak pada:

$$\begin{aligned}
&* \text{ bobot ke-1} = \\
&= \sqrt{(0-0,9025)^2 + (1-0)^2 + (0-0,0975)^2 + (0-0)^2 + (0-0,9525)^2 + (1-0,0475)^2} \\
&= 1,9075 \\
&* \text{ bobot ke-2} = \\
&= \sqrt{(0-0)^2 + (1-1)^2 + (0-1)^2 + (0-1)^2 + (0-1)^2 + (1-0)^2} \\
&= 2,000.
\end{aligned}$$

Jarak terkecil pada bobot ke-1

Target data ke-3= 1

Bobot ke- 1 baru:

$$\begin{aligned}
w_{11} &= w_{11} + \alpha^*(x_{31} - w_{11}) = 0,9025 + 0,05*(0-0,9025) = 0,8574; \\
w_{12} &= w_{12} + \alpha^*(x_{32} - w_{12}) = 0,0000 + 0,05*(0-0,0000) = 0,0500; \\
w_{13} &= w_{13} + \alpha^*(x_{33} - w_{13}) = 0,0975 + 0,05*(1-0,0975) = 0,0926; \\
w_{14} &= w_{14} + \alpha^*(x_{34} - w_{14}) = 0,0000 + 0,05*(0-0,0000) = 0,0000; \\
w_{15} &= w_{15} + \alpha^*(x_{35} - w_{15}) = 0,9525 + 0,05*(1-0,9525) = 0,9049; \\
w_{16} &= w_{16} + \alpha^*(x_{36} - w_{16}) = 0,0475 + 0,05*(0-0,0475) = 0,0951;
\end{aligned}$$

Jadi:

$$w_1 = (0,8574 \quad 0,0500 \quad 0,0926 \quad 0,0000 \quad 0,9049 \quad 0,0951)$$

④ **Data ke-4:** (1, 0, 1, 0, 1, 1)

Jarak pada:

\* bobot ke-1 =  

$$= \sqrt{(1-0,8574)^2 + (0-0,0500)^2 + (1-0,0926)^2 + (0-0)^2 + (1-0,9049)^2 + (1-0,0951)^2}$$

$$= 1,2938$$
\* bobot ke-2 =  

$$= \sqrt{(1-0)^2 + (0-1)^2 + (1-1)^2 + (0-1)^2 + (1-1)^2 + (1-0)^2}$$

$$= 2,000.$$
Jarak terkecil pada bobot ke-1  
Target data ke-4= 1  
Bobot ke- 1 baru:  

$$w_{11} = w_{11} + \alpha^*(x_{41} - w_{11}) = 0,8574 + 0,05*(1-0,8574) = 0,8645;$$

$$w_{12} = w_{12} + \alpha^*(x_{42} - w_{12}) = 0,0500 + 0,05*(0-0,0500) = 0,0475;$$

$$w_{13} = w_{13} + \alpha^*(x_{43} - w_{13}) = 0,0926 + 0,05*(1-0,0926) = 0,1380;$$

$$w_{14} = w_{14} + \alpha^*(x_{44} - w_{14}) = 0,0000 + 0,05*(0-0,0000) = 0,0000;$$

$$w_{15} = w_{15} + \alpha^*(x_{45} - w_{15}) = 0,9049 + 0,05*(1-0,9049) = 0,9096;$$

$$w_{16} = w_{16} + \alpha^*(x_{46} - w_{16}) = 0,0951 + 0,05*(1-0,0951) = 0,1404;$$
Jadi:  

$$w_1 = (0,8645 \quad 0,0475 \quad 0,1380 \quad 0,0000 \quad 0,9096 \quad 0,1404)$$

④ Data ke-5: (0, 0, 1, 1, 0, 0)

Jarak pada:  
\* bobot ke-1 =  

$$= \sqrt{(0-0,8645)^2 + (0-0,0475)^2 + (1-0,1380)^2 + (1-0)^2 + (0-0,9096)^2 + (0-0,1404)^2}$$

$$= 1,8275$$
\* bobot ke-2 =  

$$= \sqrt{(0-0)^2 + (0-1)^2 + (1-1)^2 + (1-1)^2 + (0-1)^2 + (0-0)^2}$$

$$= 1,4142.$$
Jarak terkecil pada bobot ke-2  
Target data ke-5= 2  
Bobot ke- 2 baru:  

$$w_{21} = w_{21} + \alpha^*(x_{51} - w_{21}) = 0 + 0,05*(1-0) = 0,0000;$$

$$w_{22} = w_{22} + \alpha^*(x_{52} - w_{22}) = 1 + 0,05*(0-1) = 0,9500;$$

$$w_{23} = w_{23} + \alpha^*(x_{53} - w_{23}) = 1 + 0,05*(1-1) = 1,0000;$$

$$w_{24} = w_{24} + \alpha^*(x_{54} - w_{24}) = 1 + 0,05*(0-1) = 1,0000;$$

$$w_{25} = w_{25} + \alpha^*(x_{55} - w_{25}) = 1 + 0,05*(1-1) = 0,9500;$$

$$w_{26} = w_{26} + \alpha^*(x_{56} - w_{26}) = 0 + 0,05*(1-0) = 0,0000;$$
Jadi:  

$$w_2 = (0,0000 \quad 0,9500 \quad 1,0000 \quad 1,0000 \quad 0,9500 \quad 0,0000)$$

④ Data ke-6: (0, 1, 0, 1, 0, 0)

Jarak pada:  
\* bobot ke-1 =  

$$= \sqrt{(0-0,8645)^2 + (1-0,0475)^2 + (0-0,1380)^2 + (1-0)^2 + (0-0,9096)^2 + (0-0,1404)^2}$$

$$= 1,8764$$
\* bobot ke-2 =  

$$= \sqrt{(0-0)^2 + (1-0,9500)^2 + (0-1)^2 + (1-1)^2 + (0-0,9500)^2 + (0-0)^2}$$

$$= 1,3802.$$
Jarak terkecil pada bobot ke-2  
Target data ke-6= 2  
Bobot ke- 2 baru:  

$$w_{21} = w_{21} + \alpha^*(x_{61} - w_{21}) = 0,0000 + 0,05*(0-0,0000) = 0,0000;$$

$$\begin{aligned}
w_{22} &= w_{22} + \alpha^*(x_{62} - w_{22}) = 0,9500 + 0,05*(1-0,9500) = 0,9525; \\
w_{23} &= w_{23} + \alpha^*(x_{63} - w_{23}) = 1,0000 + 0,05*(0-1,0000) = 0,9500; \\
w_{24} &= w_{24} + \alpha^*(x_{64} - w_{24}) = 1,0000 + 0,05*(1-1,0000) = 1,0000; \\
w_{25} &= w_{25} + \alpha^*(x_{65} - w_{25}) = 0,9500 + 0,05*(0-0,9500) = 0,9025; \\
w_{26} &= w_{26} + \alpha^*(x_{66} - w_{26}) = 0,0000 + 0,05*(0-0,0000) = 0,0000;
\end{aligned}$$

Jadi:

$$w_2 = (0,0000 \quad 0,9525 \quad 0,9500 \quad 1,0000 \quad 0,9025 \quad 0,0000)$$

④ Data ke-7: (1, 0, 0, 1, 0, 1)

Jarak pada:

\* bobot ke-1 =

$$\begin{aligned}
&= \sqrt{(1-0,8645)^2 + (0-0,0475)^2 + (0-0,1380)^2 + (1-0)^2 + (0-0,9096)^2 + (1-0,1404)^2} \\
&= 1,6143
\end{aligned}$$

\* bobot ke-2 =

$$\begin{aligned}
&= \sqrt{(1-0)^2 + (0-0,9525)^2 + (0-0,9500)^2 + (1-1)^2 + (0-0,9025)^2 + (1-0)^2} \\
&= 2,1504
\end{aligned}$$

Jarak terkecil pada bobot ke-1

Target data ke-7= 2

Bobot ke- 1 baru:

$$\begin{aligned}
w_{11} &= w_{11} - \alpha^*(x_{71} - w_{11}) = 0,8645 - 0,0266*(1-0,8645) = 0,8577; \\
w_{12} &= w_{12} - \alpha^*(x_{72} - w_{12}) = 0,0475 - 0,0266*(0-0,0475) = 0,0499; \\
w_{13} &= w_{13} - \alpha^*(x_{73} - w_{13}) = 0,1380 - 0,0266*(0-0,1380) = 0,1449; \\
w_{14} &= w_{14} - \alpha^*(x_{74} - w_{14}) = 0,0000 - 0,0266*(1-0,0000) = -0,0500; \\
w_{15} &= w_{15} - \alpha^*(x_{75} - w_{15}) = 0,9096 - 0,0266*(0-0,9096) = 0,9551; \\
w_{16} &= w_{16} - \alpha^*(x_{76} - w_{16}) = 0,1404 - 0,0266*(1-0,1404) = 0,0974;
\end{aligned}$$

Jadi:

$$w_1 = (0,8577 \quad 0,0499 \quad 0,1449 \quad -0,0500 \quad 0,9511 \quad 0,0974)$$

④ Data ke-8: (0, 1, 1, 1, 1, 1)

Jarak pada:

\* bobot ke-1 =

$$\begin{aligned}
&= \sqrt{(0-0,8577)^2 + (1-0,0499)^2 + (1-0,1449)^2 + (1+0,0500)^2 + (1-0,9551)^2 + (1-0,0974)^2} \\
&= 2,0710
\end{aligned}$$

\* bobot ke-2 =

$$\begin{aligned}
&= \sqrt{(0-0)^2 + (1-0,9525)^2 + (1-0,9500)^2 + (1-1)^2 + (1-0,9025)^2 + (1-0)^2} \\
&= 1,0028
\end{aligned}$$

Jarak terkecil pada bobot ke-2

Target data ke-8= 2

Bobot ke- 2 baru:

$$\begin{aligned}
w_{21} &= w_{21} + \alpha^*(x_{81} - w_{21}) = 0,0000 + 0,05*(0-0,0000) = 0,0000; \\
w_{22} &= w_{22} + \alpha^*(x_{82} - w_{22}) = 0,9525 + 0,05*(1-0,9525) = 0,9549; \\
w_{23} &= w_{23} + \alpha^*(x_{83} - w_{23}) = 0,9500 + 0,05*(0-0,9500) = 0,9525; \\
w_{24} &= w_{24} + \alpha^*(x_{84} - w_{24}) = 1,0000 + 0,05*(1-1,0000) = 1,0000; \\
w_{25} &= w_{25} + \alpha^*(x_{85} - w_{25}) = 0,9025 + 0,05*(0-0,9025) = 0,9074; \\
w_{26} &= w_{26} + \alpha^*(x_{86} - w_{26}) = 0,0000 + 0,05*(0-0,0000) = 0,0500;
\end{aligned}$$

Jadi:

$$w_2 = (0,0000 \quad 0,9549 \quad 0,9525 \quad 1,0000 \quad 0,9074 \quad 0,0500)$$

$$\alpha = \alpha - 0,1*\alpha = 0,05 - 0,1*0,05 = 0,045$$

Proses tersebut diteruskan untuk epoh ke-2 sampai ke-10, untuk setiap data dengan menggunakan cara yang sama. Setelah mencapai epoh yang ke-10 diperoleh bobot akhir:

$$w_1 = (0,3727 \quad 0,2161 \quad 0,6347 \quad -0,2164 \quad 0,7981 \quad 0,4254)$$

$$w_2 = (0,0000 \quad 0,7969 \quad 0,7900 \quad 1,0000 \quad 0,5869 \quad 0,2171)$$

Apabila kita ingin mensimulasikan input: (0, 1, 0, 1, 1, 0), maka kita cari terlebih dahulu jarak input tersebut dengan kedua bobot. Nomor dari bobot dengan jarak yang terpendek akan menjadi kelasnya.

Jarak pada:

\* bobot ke-1 =

$$= \sqrt{(0-0,3727)^2 + (1-0,2161)^2 + (0-0,6347)^2 + (1+0,2164)^2 + (1-0,7981)^2 + (0-0,4254)^2}$$

$$= 1,6904$$

\* bobot ke-2 =

$$= \sqrt{(0-0)^2 + (1-0,7969)^2 + (0-0,7900)^2 + (1-1)^2 + (1-0,5869)^2 + (0-0,2171)^2}$$

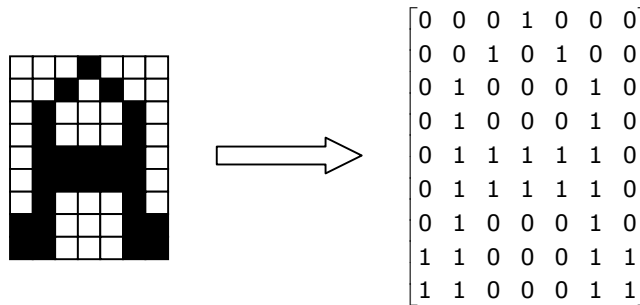
$$= 0,9398$$

Jarak terkecil pada bobot ke-2

Sehingga input tersebut termasuk dalam kelas 2.

### Contoh 8.9:

Akan dicoba untuk mengenali huruf A, B, atau H yang direpresentasikan dengan menggunakan kode 0 dan 1 pada matriks berukuran 9x7 seperti terlihat pada Gambar 8.30.

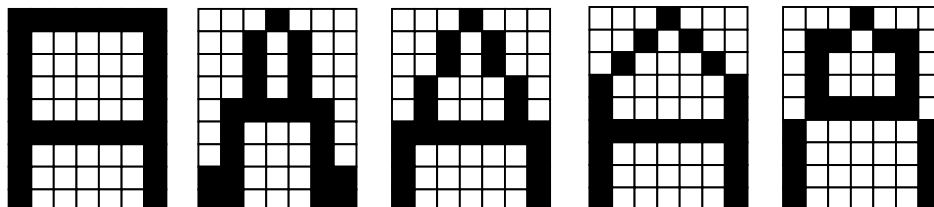


Gambar 8.30 Pemetaan huruf pada matriks 9x7.

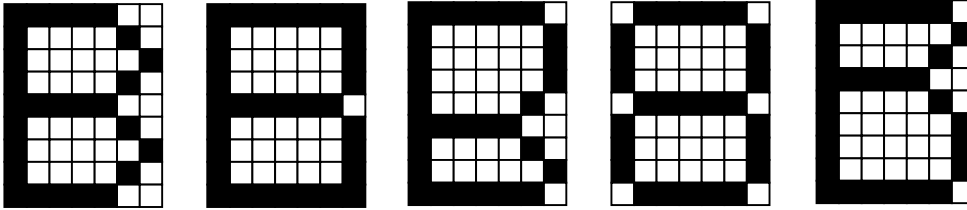
Pada Gambar 8.30 tersebut, kode 1 menunjukkan suatu kotak berwarna hitam, sedangkan kode 0 menunjukkan suatu kotak berwarna putih. Untuk mempermudah dalam mengimplementasikannya, matriks 9x7 tersebut dibawa ke bentuk vektor 63 kolom. Pada Gambar 8.30 tersebut vektor yang bersesuaian adalah:

000100000101000100010010001001111100111110010001011000111100011

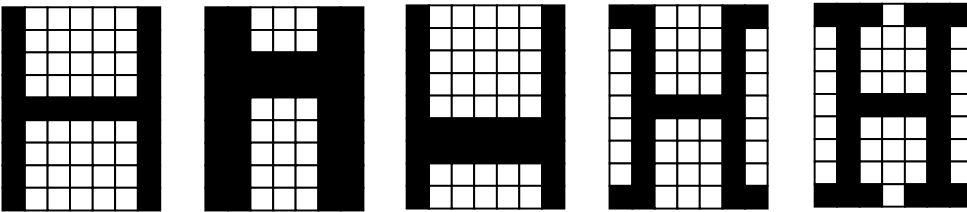
Misalkan ada 15 data yang akan dilatih, yaitu 5 data A, 5 data B, dan 5 data H, sebagai berikut (Gambar 8.31, 8.32 dan 8.33):



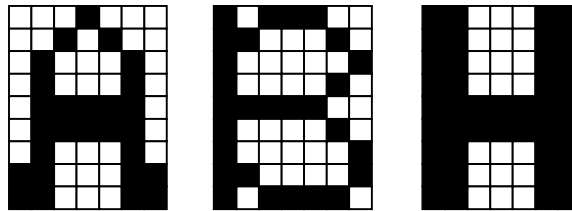
Gambar 8.31 Pola huruf 'A' pada matriks 9x7.



Gambar 8.32 Pola huruf 'B' pada matriks 9x7.



Gambar 8.33 Pola huruf 'H' pada matriks 9x7.



Gambar 8.34 Pola huruf yang digunakan sebagai bobot .

Matrik yang akan dilatih adalah transformasi Gambar 8.31, 8.32, dan 8.33 yaitu:

```
P=[111111110000011000001100000110000011111111100000110000011000001;
000100000101000010100001010001111100100010010001011000111100011;
000100000101000010100010001001000101111111100000110000011000001;
000100000101000100010100000110000011111111100000110000011000001;
000100001101100100010010001001111101000001100000110000011000001;
111110010000101000001100001011111001000010100000110000101111100;
1111111100000110000011000001111110100000110000011000001111111;
11111101000001100000110000011000010111110010000101000001111110;
11111101000001100000110000010111110100000110000011000001011110;
111111010000011000001011111001000010100000110000011000001111110;
10000011000001100000110000011111111000001100000110000011000001;
1100011110001111111111111111000111100011110001111000111100011;
100000110000011000001100000110000011111111111111110000011000001;
110001101000100100010010001001111100100010010001001000101100011;
11101110100010010001001000100111110010001001000100100010110111];
```

dengan vektor output (target):

```
T=[1 1 1 1 1 2 2 2 2 2 3 3 3 3 3];
```

Angka 1 menunjukkan kelas A, angka 2 menunjukkan kelas B, dan angka 3 menunjukkan kelas H.

Sedangkan untuk matriks bobot diperoleh dari transformasi dari Gambar 8.34.

```
w=[000100000101000100010010001001111100111110010001011000111100011;
101110011000101000001100001010111001000010100000111000011011110;
1100011110001111000111100011111111111111110001111000111100011];
```

Sebagai nilai awal dipilih learning rate ( $\alpha=0,05$ ), dengan pengurangan sebesar  $0,1*\alpha$ ; dan maksimum epoh (MaxEpoh=30).

Epoh ke-1:

④ Data ke-1:

$P_1=(111111110000011000001100000110000011111111100000110000011000001)$ ;

Jarak pada:

\* Bobot ke-1 = 5,9161

\* Bobot ke-2 = 4,7958

\* Bobot ke-3 = 4,4721

Jarak terkecil pada bobot ke-3

Target data ke-1= 1

Bobot ke- 3 baru ( $w_3$ ):

|        |        |         |         |         |        |        |        |     |
|--------|--------|---------|---------|---------|--------|--------|--------|-----|
| 1,0000 | 1,0000 | -0,0500 | -0,0500 | -0,0500 | 1,0000 | 1,0000 | 1,0000 | ... |
| 1,0500 | 0,0000 | 0,0000  | 0,0000  | 1,0500  | 1,0000 | 1,0000 | 1,0500 | ... |
| 0,0000 | 0,0000 | 0,0000  | 1,0500  | 1,0000  | 1,0000 | 1,0500 | 0,0000 | ... |
| 0,0000 | 0,0000 | 1,0500  | 1,0000  | 1,0000  | 1,0500 | 1,0500 | 1,0500 | ... |
| 1,0500 | 1,0500 | 1,0000  | 1,0000  | 1,0000  | 1,0000 | 1,0000 | 1,0000 | ... |
| 1,0000 | 1,0000 | 1,0000  | 1,0500  | 0,0000  | 0,0000 | 0,0000 | 1,0500 | ... |
| 1,0000 | 1,0000 | 1,0500  | 0,0000  | 0,0000  | 0,0000 | 1,0500 | 1,0000 | ... |
| 1,0000 | 1,0500 | 0,0000  | 0,0000  | 0,0000  | 1,0500 | 1,0000 |        |     |

$\alpha = 0,0450$

④ Data ke-2:

$P_2=(000100000101000010100001010001111100100010010001011000111100011)$ ;

Jarak pada

\* Bobot ke-1 = 3,3166

\* Bobot ke-2 = 5,5678

\* Bobot ke-3 = 5,7228

Jarak terkecil pada bobot ke-1

Target data ke-2= 1

Bobot ke-1 baru ( $w_1$ ):

|        |        |        |        |        |        |        |        |     |
|--------|--------|--------|--------|--------|--------|--------|--------|-----|
| 0,0000 | 0,0000 | 0,0000 | 1,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 | ... |
| 0,0000 | 1,0000 | 0,0000 | 1,0000 | 0,0000 | 0,0000 | 0,0000 | 0,9550 | ... |
| 0,0450 | 0,0000 | 0,0450 | 0,9550 | 0,0000 | 0,0000 | 0,9550 | 0,0450 | ... |
| 0,0000 | 0,0450 | 0,9550 | 0,0000 | 0,0000 | 1,0000 | 1,0000 | 1,0000 | ... |
| 1,0000 | 1,0000 | 0,0000 | 0,0000 | 1,0000 | 0,9550 | 0,9550 | 0,9550 | ... |
| 1,0000 | 0,0000 | 0,0000 | 1,0000 | 0,0000 | 0,0000 | 0,0000 | 1,0000 | ... |
| 0,0000 | 1,0000 | 1,0000 | 0,0000 | 0,0000 | 0,0000 | 1,0000 | 1,0000 | ... |
| 1,0000 | 1,0000 | 0,0000 | 0,0000 | 0,0000 | 1,0000 | 1,0000 |        |     |

$\alpha = 0,0405$

Proses tersebut diteruskan untuk data ke-3 sampai data ke-15, kemudian dilanjutkan epoh ke-2 sampai ke-30, untuk setiap data dengan menggunakan cara yang sama. Setelah mencapai epoh yang ke-30 diperoleh bobot akhir:



$W_1 =$

|         |         |         |        |         |         |         |        |     |
|---------|---------|---------|--------|---------|---------|---------|--------|-----|
| -0,0297 | -0,0297 | -0,0140 | 1,0297 | -0,0140 | -0,0297 | -0,0297 | 0,0000 | ... |
| 0,0113  | 1,0297  | 0,0000  | 1,0297 | 0,0113  | 0,0000  | 0,0000  | 0,9010 | ... |
| 0,0990  | 0,0000  | 0,0990  | 0,9010 | 0,0000  | 0,0443  | 0,9044  | 0,0513 | ... |
| 0,0000  | 0,0513  | 0,9044  | 0,0443 | 0,0443  | 0,9557  | 0,9080  | 0,9080 | ... |
| 0,9080  | 0,9557  | 0,0443  | 0,1330 | 0,9590  | 0,9375  | 0,9375  | 0,9375 | ... |
| 0,9590  | 0,1330  | 0,1330  | 0,8670 | 0,0000  | 0,0000  | 0,0000  | 0,8670 | ... |
| 0,1330  | 1,0297  | 0,8670  | 0,0000 | 0,0000  | 0,0000  | 0,8670  | 1,0297 | ... |
| 1,0000  | 0,8670  | -0,0140 | 0,0000 | -0,0140 | 0,8670  | 1,0000  |        |     |

$W_2 =$

|        |        |         |        |        |        |         |        |     |
|--------|--------|---------|--------|--------|--------|---------|--------|-----|
| 1,0000 | 0,1358 | 1,0217  | 1,0217 | 1,0217 | 0,1014 | -0,0005 | 1,0000 | ... |
| 0,8858 | 0,0000 | 0,0000  | 0,0000 | 0,9203 | 0,0797 | 1,0000  | 0,0000 | ... |
| 0,0000 | 0,0000 | 0,0000  | 0,0244 | 0,9756 | 1,0000 | 0,0244  | 0,0244 | ... |
| 0,0244 | 0,0244 | 0,9203  | 0,0553 | 0,9733 | 0,0712 | 0,9570  | 0,9570 | ... |
| 0,9570 | 0,0903 | -0,0322 | 1,0000 | 0,0185 | 0,0185 | 0,0185  | 0,0185 | ... |
| 0,9097 | 0,0506 | 1,0000  | 0,0000 | 0,0000 | 0,0000 | 0,0000  | 0,0291 | ... |
| 0,9709 | 1,0000 | 0,8858  | 0,0000 | 0,0000 | 0,0000 | 0,0345  | 0,9655 | ... |
| 0,9733 | 0,1464 | 1,0322  | 1,0322 | 1,0322 | 0,9978 | -0,0005 |        |     |

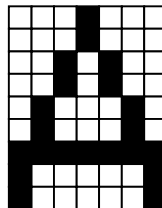
$W_3 =$

|        |        |         |         |         |        |        |        |     |
|--------|--------|---------|---------|---------|--------|--------|--------|-----|
| 1,0000 | 0,9823 | -0,0504 | -0,0504 | -0,0504 | 0,9823 | 1,0000 | 1,0000 | ... |
| 1,0327 | 0,0000 | 0,0000  | 0,0000  | 1,0327  | 1,0000 | 1,0000 | 1,0327 | ... |
| 0,0194 | 0,0194 | 0,0194  | 1,0327  | 1,0000  | 1,0000 | 1,0327 | 0,0194 | ... |
| 0,0194 | 0,0194 | 1,0327  | 1,0000  | 1,0000  | 1,0327 | 1,0132 | 1,0132 | ... |
| 1,0132 | 1,0327 | 1,0000  | 1,0000  | 1,0000  | 0,9806 | 0,9806 | 0,9806 | ... |
| 1,0000 | 1,0000 | 1,0000  | 1,0504  | 0,0177  | 0,0177 | 0,0177 | 1,0504 | ... |
| 1,0000 | 1,0000 | 1,0327  | 0,0000  | 0,0000  | 0,0000 | 1,0327 | 1,0000 | ... |
| 1,0000 | 1,0327 | 0,0000  | 0,0000  | 0,0000  | 1,0327 | 1,0000 |        |     |

Apabila kita ingin melakukan testing input:

$X = (00010000001000001010000101000100010010001011111110000011000001);$

yang bersesuaian dengan Gambar 8.35.



Gambar 8.35 Pola huruf yang digunakan sebagai tes .

maka kita cari terlebih dahulu jarak input tersebut dengan kedua bobot. Nomor dari bobot dengan jarak yang terpendek akan menjadi kelasnya.

Jarak pada:

\* bobot ke-1 = 0,0693

\* bobot ke-2 = 1,7655

\* bobot ke-3 = 2,2446

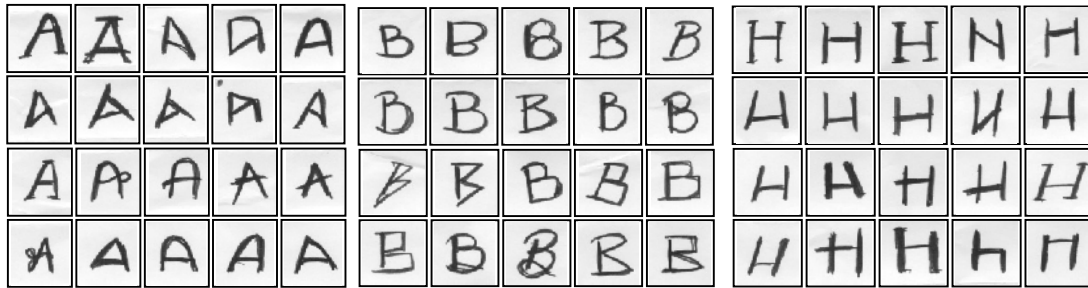
Jarak terkecil pada bobot ke-1

Sehingga input tersebut termasuk dalam kelas A.

#### Contoh 8.10:

Kita memiliki data tentang beberapa pola karakter A, B dan H dengan menggunakan tulisan tangan seperti terlihat pada Gambar 8.36. Kita akan

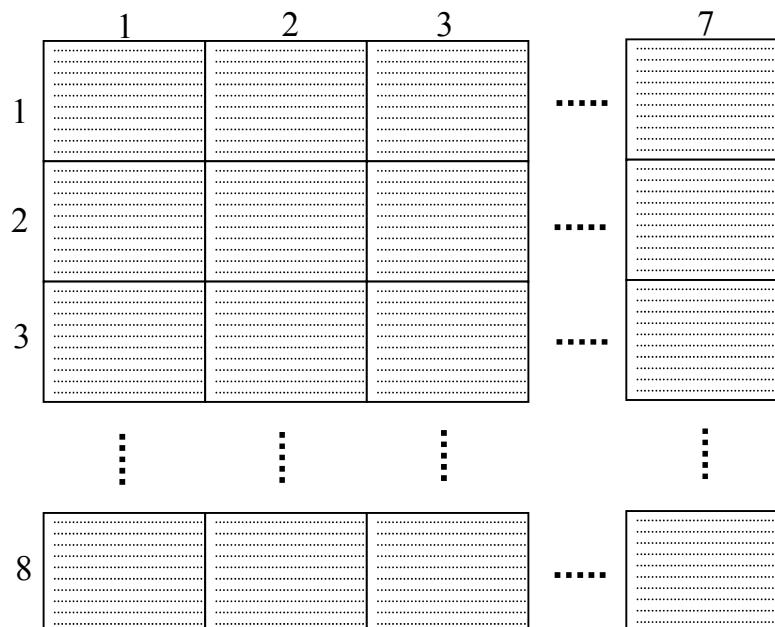
menggunakan Metode LVQ untuk melatih pola-pola yang ada, sehingga nantinya kalo ada pola baru, kita dapat menentukan pola tersebut termasuk dalam kelas mana.



Gambar 8.36 Pola huruf A, B, H dengan tulisan tangan.

Sebelum kita melatih pola-pola yang telah ada, terlebih dahulu akan kita lakukan preprosesing sebagai berikut:

- \* Penyederhanaan pola citra karakter dengan melakukan deteksi tepi. Deteksi tepi ini dilakukan untuk menghilangkan tepi-tepi citra yang tidak berisi garis-garis penyusun citra pola karakter. Deteksi tepi dilakukan dengan menggunakan operator Sobel. Selama proses deteksi tepi ini dilakukan penelusuran gambar secara vertikal dan horisontal sambil melihat apakah terjadi perubahan warna secara mendadak yang melebihi suatu harga antara 2 titik yang berdekatan.
- \* Setelah disederhanakan, setiap citra pola karakter dibagi menjadi matriks berukuran  $m \times n$  (m baris, n kolom). Pada contoh ini, matriks yang terjadi berukuran  $8 \times 7$ . Setiap elemen matriks berisi  $10 \times 10$  piksel. Lihat Gambar 8.37. Setiap kotak (elemen) kita lakukan penelusuran. Jika ditemukan nilai 1 (ada piksel) dalam jumlah tertentu (dalam contoh ini  $> 10$  piksel), maka kita beri nilai 1 pada kotak tersebut. Namun jika tidak demikian, kita beri nilai 0 untuk kotak tersebut. Dari sini kita peroleh matriks berukuran  $8 \times 7$ , dengan tiap elemen bernilai 1 atau 0. Kemudian, matriks tersebut kita bawa ke bentuk vektor dengan ukuran  $mn$  (56 elemen). Karena setiap pola berupa vektor berukuran 56 elemen, maka apabila kita punya 60 pola, maka sekarang kita memiliki matriks berukuran  $60 \times 56$ .



Gambar 8.37 Matriks 8x7 dengan tiap elemen berisi 10x10 piksel.

Dari matriks berukuran 60x56 ini kita gunakan 3 baris sebagai bobot. Ketiga baris tersebut adalah elemen matriks pada baris ke-1 (pola huruf A), baris ke-21 (pola huruf B), dan baris ke 41 (pola huruf H). Ukuran matriks bobot 56x3. Sehingga untuk pembelajaran hanya kita lakukan pada elemen-elemen sisanya, yaitu matriks berukuran 57x56. Vektor target berukuran 57, dengan nilai 1 yang mewakili huruf A, nilai 2 mewakili huruf B, dan nilai 3 mewakili huruf H.


Matriks bobot setelah epoh ke-10 adalah:

W =

|        |        |        |
|--------|--------|--------|
| 0,000  | 0,000  | -0,066 |
| 0,000  | 0,000  | 0,000  |
| 0,000  | 0,000  | 0,000  |
| 0,000  | -0,054 | 0,000  |
| 0,000  | 0,000  | 0,000  |
| 0,000  | 0,000  | 0,000  |
| 0,000  | 0,000  | 0,000  |
| 0,000  | 0,000  | 0,000  |
| -0,016 | 0,214  | 0,191  |
| 0,016  | 0,974  | -0,029 |
| 0,932  | 0,805  | -0,166 |
| 0,933  | 0,495  | 0,028  |
| 0,000  | 0,019  | 0,285  |
| 0,000  | 0,000  | 0,000  |
| 0,000  | 0,000  | 0,000  |
| -0,105 | 0,904  | 1,152  |
| 0,108  | 0,947  | 0,431  |
| 0,914  | 0,640  | -0,293 |
| 0,910  | 0,524  | 0,939  |
| 0,884  | 0,228  | 0,733  |
| 0,000  | 0,000  | 0,000  |
| 0,000  | 0,000  | 0,000  |
| -0,088 | 0,098  | 1,075  |
| 0,960  | 1,009  | 0,702  |
| 0,853  | 1,163  | 0,508  |
| 1,000  | 0,957  | 0,952  |
| 1,070  | 0,067  | 0,762  |
| 0,000  | 0,000  | 0,000  |
| 0,000  | 0,000  | 0,000  |
| 0,157  | 0,085  | 1,142  |
| 0,947  | 0,926  | 0,958  |
| 0,149  | 0,031  | 0,830  |
| 0,272  | 0,619  | 1,013  |
| 1,025  | 1,031  | 0,613  |
| 0,760  | 0,000  | 0,000  |
| 0,000  | 0,000  | 0,069  |
| 0,866  | 0,585  | 1,038  |
| 0,990  | 1,108  | 0,490  |
| -0,075 | 0,952  | 0,103  |
| 0,147  | 0,983  | 1,028  |
| 1,080  | 0,171  | 0,334  |
| 0,760  | 0,000  | 0,000  |
| 0,000  | -0,056 | 0,000  |
| 0,730  | 0,304  | 0,931  |


|        |        |       |
|--------|--------|-------|
| -0,040 | 0,608  | 0,178 |
| -0,040 | 0,828  | 0,032 |
| -0,024 | 0,422  | 0,716 |
| -0,017 | -0,058 | 0,198 |
| 0,000  | 0,000  | 0,000 |
| 0,000  | 0,000  | 0,000 |
| 0,000  | 0,000  | 0,000 |
| 0,000  | 0,000  | 0,000 |
| 0,000  | 0,000  | 0,000 |
| 0,000  | 0,000  | 0,000 |
| 0,000  | 0,000  | 0,000 |
| 0,000  | 0,000  | 0,000 |
| 0,000  | 0,000  | 0,000 |

Sekarang kita akan lakukan testing terhadap pola karakter yang ikut dilatih, yaitu input data ke-2 masing-masing untuk huruf A, B dan H.

\* Untuk huruf A: 


- o Jarak terhadap bobot ke-1 = 3,0681
- o Jarak terhadap bobot ke-2 = 3,0736
- o Jarak terhadap bobot ke-3 = 3,4500

Jarak paling pendek adalah jarak dengan bobot ke-1, dengan demikian pola karakter tersebut dikenali sebagai huruf 'A'.

\* Untuk huruf B: 

- o Jarak terhadap bobot ke-1 = 3,5087
- o Jarak terhadap bobot ke-2 = 2,4079
- o Jarak terhadap bobot ke-3 = 3,5897

Jarak paling pendek adalah jarak dengan bobot ke-2, dengan demikian pola karakter tersebut dikenali sebagai huruf 'B'.


\* Untuk huruf H: 








- o Jarak terhadap bobot ke-1 = 3,3045
- o Jarak terhadap bobot ke-2 = 3,9236
- o Jarak terhadap bobot ke-3 = 2,1927

Jarak paling pendek adalah jarak dengan bobot ke-3, dengan demikian pola karakter tersebut dikenali sebagai huruf 'H'.

Kemudian kita akan melakukan testing terhadap pola karakter yang belum dikenali (tidak ikut dilatih). Pola tersebut ada yang berupa karakter yang sama sekali tidak ikut dilatih, dan ada pula yang merupakan modifikasi dari karakter yang ikut dilatih dengan menghilangkan beberapa bagiannya. Hasilnya terlihat pada Tabel 8.3:

Tabel 8.3 Hasil testing terhadap pola yang belum dikenali.

| Pola  | Jarak terhadap bobot ke- |        |        | Jarak terdekat dengan bobot ke- | Dikenali sebagai huruf |
|---|--------------------------|--------|--------|---------------------------------|------------------------|
|   | 1                        | 2      | 3      |                                 |                        |
|  | 3,6943                   | 3,3908 | 2,8775 | 3                               | H                      |

|   |        |        |        |   |   |
|---|--------|--------|--------|---|---|
|    | 3,1688 | 3,3245 | 2,9300 | 3 | H |
|    | 3,6657 | 2,8964 | 3,6568 | 2 | B |
|    | 3,2154 | 3,3497 | 3,7467 | 1 | A |
|    | 4,1239 | 2,6657 | 2,8438 | 2 | B |
|    | 3,5281 | 3,8501 | 2,5597 | 3 | H |
|    | 2,8987 | 3,3070 | 3,5529 | 1 | A |
|  | 4,1124 | 2,5066 | 3,0339 | 2 | B |

Dari hasil pada tabel terlihat bahwa, dari 8 pola tersebut hanya 1 pola yang tidak tepat dari dugaan, yaitu pola ke-2. Pada pola tersebut dugaan kita adalah mendekati huruf A, namun setelah dites ternyata pola tersebut lebih mendekati huruf H.

## 8.8 UNSUPERVISED LEARNING (JARINGAN KOHONEN)

Jaringan kohonen ini pertama kali diperkenalkan oleh Prof. Teuvo Kohonen pada tahun 1982. Pada jaringan ini, suatu lapisan yang berisi neuron-neuron akan menyusun dirinya sendiri berdasarkan input nilai tertentu dalam suatu kelompok yang dikenal dengan istilah cluster. Selama proses penyusunan diri, cluster yang memiliki vektor bobot paling cocok dengan pola input (memiliki jarak yang paling dekat) akan terpilih sebagai pemenang. Neuron yang menjadi pemenang beserta neuron-neuron tetangganya akan memperbaiki bobot-bobotnya.

Algoritma

0. Inisialisasi bobot: wij.  
Set parameter-parameter tetangga.  
Set parameter *learning rate*.
1. Kerjakan jika kondisi berhenti bernilai FALSE:
  - a. Untuk setiap vektor input x, kerjakan:

- \* Untuk setiap j, hitung:

$$D(J) = \sum_i (w_{ij} - x_i)^2$$

- \* Tentukan J, sedemikian hingga D(J) minimum.
- \* Untuk setiap unit j dengan spesifikasi tetangga tertentu, dan untuk setiap i:

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

- Perbaiki *learning rate*.
- Kurangi radius ke-tetangga-an pada waktu-waktu tertentu.
- Tes kondisi berhenti.

### Contoh 8.11:

Misalkan kita ingin mengcluster data-data berikut:

| $X_1$ | $X_2$ |
|-------|-------|
| 0,10  | 0,10  |
| 0,20  | 0,20  |
| 0,30  | 0,10  |
| 0,50  | 0,30  |
| 0,40  | 0,40  |
| 0,20  | 0,40  |

menjadi 2 cluster. Bobot awal yang akan kita gunakan adalah matriks berukuran 2x2 dengan tiap-tiap elemen bernilai 0,5. Learning rate ( $\alpha=0,6$ ) dengan tiap kenaikan epoh akan diset  $0,5 \times (\alpha)$ . Maksimum epoh ditetapkan sebesar 10.

alfa ( $\alpha$ ) = 0,60

MaxEpoh = 10

Pengurangan alfa = 0,50

Matriks yang akan dicluster, X:

|      |      |
|------|------|
| 0,10 | 0,10 |
| 0,20 | 0,20 |
| 0,30 | 0,10 |
| 0,50 | 0,30 |
| 0,40 | 0,40 |
| 0,20 | 0,40 |

Bobot awal:

|      |      |
|------|------|
| 0,50 | 0,50 |
| 0,50 | 0,50 |

Epoh ke- 1

-----  
Data ke- 1

Jarak pada:

$$* \text{ bobot ke-1} = (0,5 - 0,1)^2 + (0,5 - 0,1)^2 = 0,32$$

$$* \text{ bobot ke-2} = (0,5 - 0,1)^2 + (0,5 - 0,1)^2 = 0,32$$

Jarak terkecil pada bobot ke-1

Bobot ke- 1 baru:

$$w_{11} = 0,5 + 0,6(0,1 - 0,5) = 0,26$$

$$w_{12} = 0,5 + 0,6(0,1 - 0,5) = 0,26$$

Data ke- 2

Jarak pada:

$$* \text{ bobot ke-1} = (0,26 - 0,2)^2 + (0,26 - 0,2)^2 = 0,0072$$

$\ast \text{ bobot ke-2} = (0,50 - 0,2)^2 + (0,50 - 0,2)^2 = 0,1800$   
 Jarak terkecil pada bobot ke-1  
 Bobot ke- 1 baru:  
 $w_{11} = 0,26 + 0,6(0,2 - 0,26) = 0,224$   
 $w_{12} = 0,26 + 0,6(0,2 - 0,26) = 0,224$

Data ke- 3  
 Jarak pada:  
 $\ast \text{ bobot ke-1} = (0,224 - 0,3)^2 + (0,224 - 0,1)^2 = 0,0212$   
 $\ast \text{ bobot ke-2} = (0,5 - 0,3)^2 + (0,5 - 0,1)^2 = 0,2000$   
 Jarak terkecil pada bobot ke-1  
 Bobot ke- 1 baru:  
 $w_{11} = 0,224 + 0,6(0,3 - 0,224) = 0,2696$   
 $w_{12} = 0,224 + 0,6(0,1 - 0,224) = 0,1496$

Data ke- 4  
 Jarak pada:  
 $\ast \text{ bobot ke-1} = (0,2696 - 0,5)^2 + (0,1496 - 0,3)^2 = 0,0757$   
 $\ast \text{ bobot ke-2} = (0,5 - 0,5)^2 + (0,5 - 0,3)^2 = 0,0400$   
 Jarak terkecil pada bobot ke-2  
 Bobot ke- 2 baru:  
 $w_{21} = 0,5 + 0,6(0,5 - 0,5) = 0,5000$   
 $w_{22} = 0,5 + 0,6(0,3 - 0,5) = 0,3800$

Data ke- 5  
 Jarak pada:  
 $\ast \text{ bobot ke-1} = (0,2696 - 0,4)^2 + (0,1496 - 0,4)^2 = 0,0797$   
 $\ast \text{ bobot ke-2} = (0,5 - 0,4)^2 + (0,38 - 0,4)^2 = 0,0104$   
 Jarak terkecil pada bobot ke-2  
 Bobot ke- 2 baru:  
 $w_{21} = 0,5 + 0,6(0,4 - 0,5) = 0,4400$   
 $w_{22} = 0,38 + 0,6(0,4 - 0,38) = 0,3920$

Data ke- 6  
 Jarak pada:  
 $\ast \text{ bobot ke-1} = (0,2696 - 0,2)^2 + (0,1496 - 0,4)^2 = 0,0675$   
 $\ast \text{ bobot ke-2} = (0,44 - 0,2)^2 + (0,392 - 0,4)^2 = 0,0577$   
 Jarak terkecil pada bobot ke-2  
 Bobot ke- 2 baru:  
 $w_{21} = 0,44 + 0,6(0,2 - 0,44) = 0,2960$   
 $w_{22} = 0,392 + 0,6(0,4 - 0,392) = 0,3968$

$\alpha = 0,5 \ast 0,6 = 0,3$

Nilai bobot akhir setelah epoch ke-10 adalah:

$w =$   
 $\begin{matrix} 0,2190 & 0,3424 \\ 0,1351 & 0,3754 \end{matrix}$

Misalkan kita akan melakukan testing terhadap data ketiga: [0,3 0,1] termasuk dalam cluster mana, maka kita cari terlebih dahulu jarak data tersebut pada bobot setiap cluster:

Jarak pada:  
 $\ast \text{ bobot ke-1} = (0,2190 - 0,3)^2 + (0,1351 - 0,1)^2 = 0,0078$   
 $\ast \text{ bobot ke-2} = (0,3424 - 0,3)^2 + (0,3754 - 0,1)^2 = 0,0776$

Ternyata jarak yang lebih pendek adalah jarak terhadap bobot ke-1, maka data tersebut termasuk dalam cluster pertama.

Kita juga bisa melakukan testing terhadap data yang tidak ikut dilatih, misalkan: [0,4 0,5] termasuk dalam cluster mana, maka seperti biasa kita cari terlebih dahulu jarak data tersebut pada bobot setiap cluster:

Jarak pada:

$$* \text{ bobot ke-1} = (0,2190 - 0,4)^2 + (0,1351 - 0,5)^2 = 0,1997$$

$$* \text{ bobot ke-2} = (0,3424 - 0,4)^2 + (0,3754 - 0,5)^2 = 0,0404$$

Ternyata jarak yang lebih pendek adalah jarak terhadap bobot ke-2, maka data tersebut termasuk dalam cluster kedua.