Artificial Neural Network

- Pelatihan ANN di RC-OPPINET ITB
- Sri Kusumadewi, Artificial Intelligence (Teknik dan Aplikasinya).
- Perkuliah ANN Elektro ITB

ANN

- Model komputasi yang terinspirasi dari nuerological model brain/otak.
- Otak manusia bekerja dengan cara yang berbeda dibandingkan komputer digital.
  - highly complex, nonlinear, and parallel computing
  - many times faster than d-computer in
    - pattern recognition, perception, motor control
  - has great structure and ability to build up its own rules by experience
    - dramatic development within 2 years after birth
    - continues to develop afterward
      - Language Learning Device before 13 years old
- Plasticity : ability to adapt to its environment

Jaringan Syaraf Biologi

- Struktur otak manusia sangat komplek
- Otak manusia terdiri dari sel-sel syaraf (neuron) dan penghubung (sinapsis)
- Neuron bekerja berdasarkan impuls/sinyal yang diberikan padanya, dan diteruskan ke neuron lainnya.

Kemampuan otak

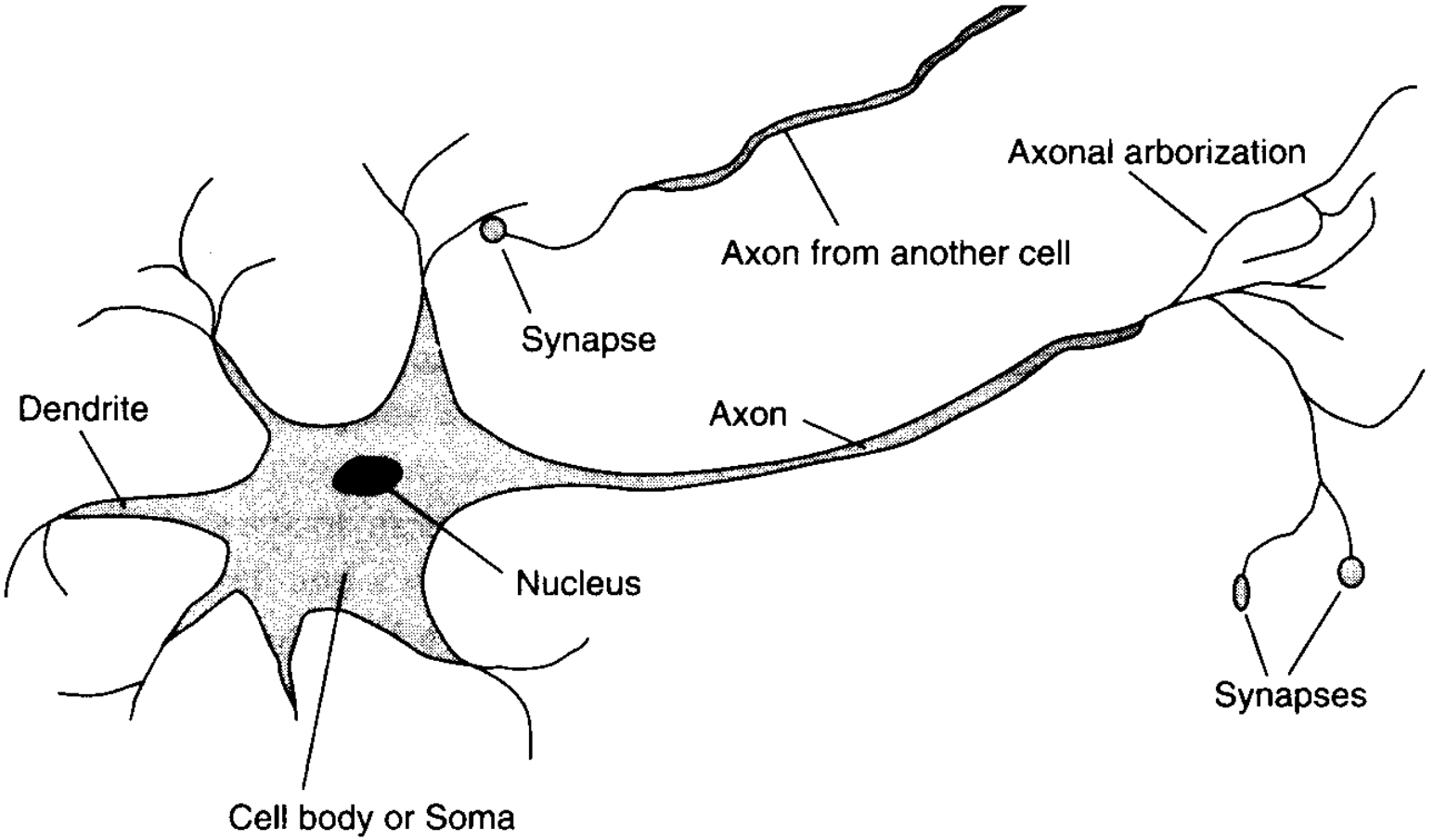
- Diantaranya :
  - Mengenali pola
  - Melakukan perhitungan
  - Mengontrol organ-organ tubuh

  semuanya dilakukan dengan kecepatan tinggi dibandingkan komputer digital.

  Contoh:

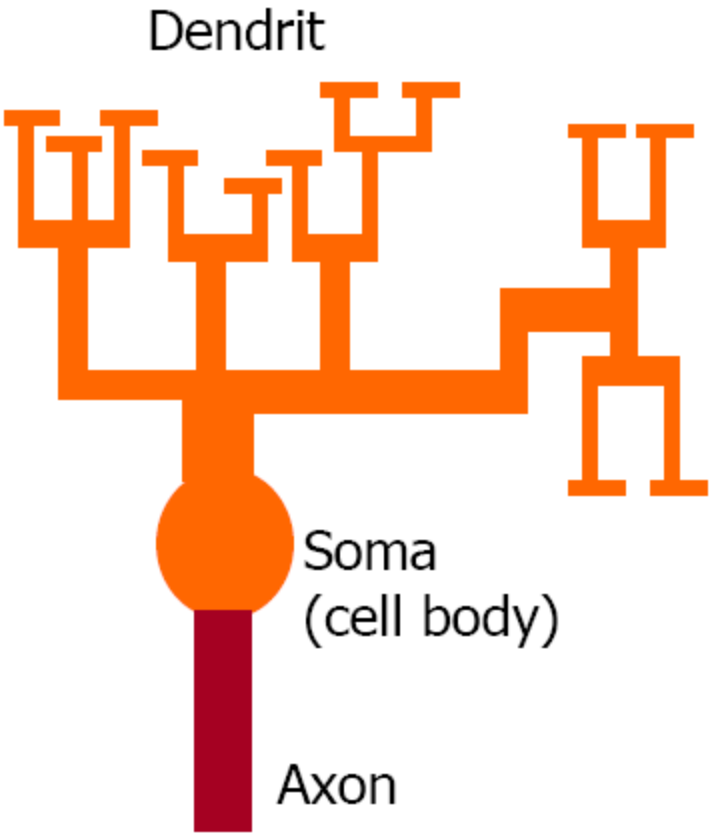
  Pengenalan wajah seseorang yang sedikit berubah (memakai topi, jenggot tambahan).

Biological Neuron



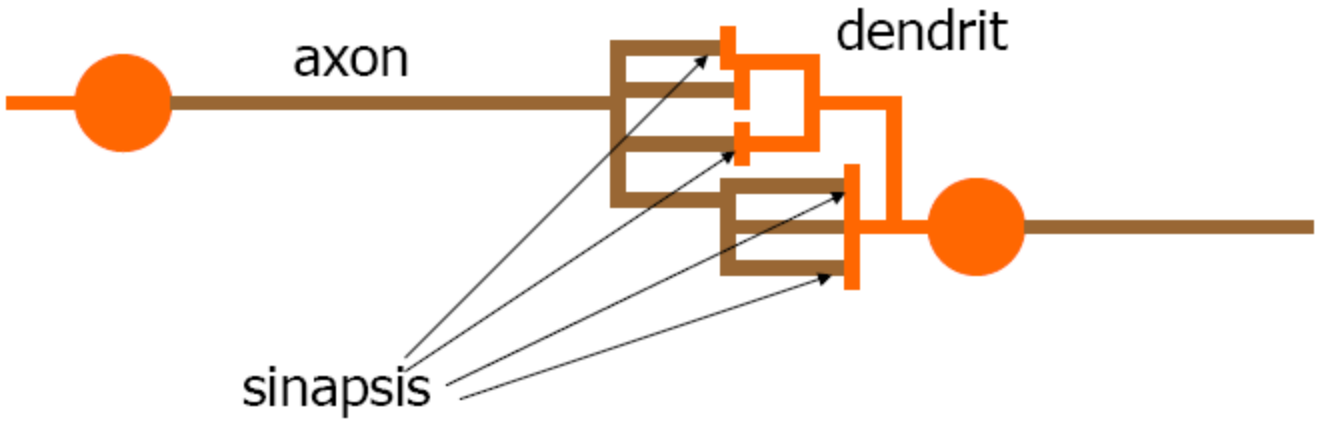
Excerpted from *Artificial Intelligence: Modern Approach* by S. Russel and P. Norvig

Komponen Penting Neuron



- Dendrit
  - Menerima sinyal kimia listrik dari neuron lain
- Soma
  - Menjumlahkan semua sinyal yang masuk
  - Jika hasil penjumlahan sinyal cukup kuat atau melebihi threshold, sinyal akan diteruskan ke sel lain melalui axon
- Axon
  - Mengirimkan sinyal ke neuron lain

Koneksi Antar Neuron



- Pengiriman sinyal / informasi terjadi pada sinapsis
- Sinapsis memperkuat atau memperlemah sinyal yang hendak dikirimkan

Jaringan Syaraf Tiruan

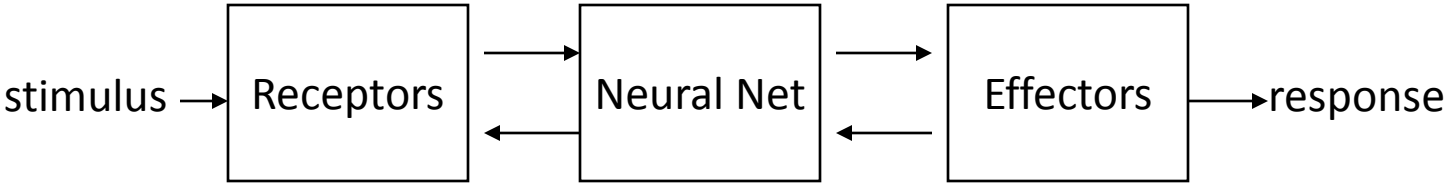
- Sistem Pemrosesan informasi dengan karakteristik menyerupai jaringan syaraf biologi.
- Dibentuk sebagai generalisasi model matematika dari jaringan syaraf biologi.
- Asumsi pada model:
  - Pemrosesan informasi terjadi pada neuron.
  - Sinyal dikirimkan antar neuron melalui penghubung.
  - Penghubung antar neuron memiliki bobot yang akan memperkuat atau memperlemahkan sinyal.
  - Untuk menentukan output, setiap neuron menggunakan fungsi aktivasi, yang menentukan sinyal diteruskan ke neuron lain atau tidak.

Definisi Neural Network

- Machine designed to model the way in which brain performs tasks
  - implemented by electronic devices and/or software (simulation)
  - Learning is the major emphasis of NN
- Massively parallel distributed processor
  - massive interconnection of simple processing units
  - simple processing units store experience and make it available to use
  - knowledge is acquired from environment thru learning process
- Learning Machine
  - modify synaptic weights to obtain design objective
  - modify own topology – neurons die and new one can grow
- Connectionist network – connectionism

Human Brain

- Human nervous system



  - 10^{11} neurons in human cortex
  - 60 x 10^{12} synaptic connections
  - 10^{4} synapses per neuron
  - 10^{-3} sec cycle time (computer : 10^{-9} sec)
  - energetic efficiency : 10^{-16} joules operation per second

    (computer : 10^{-6} joules)

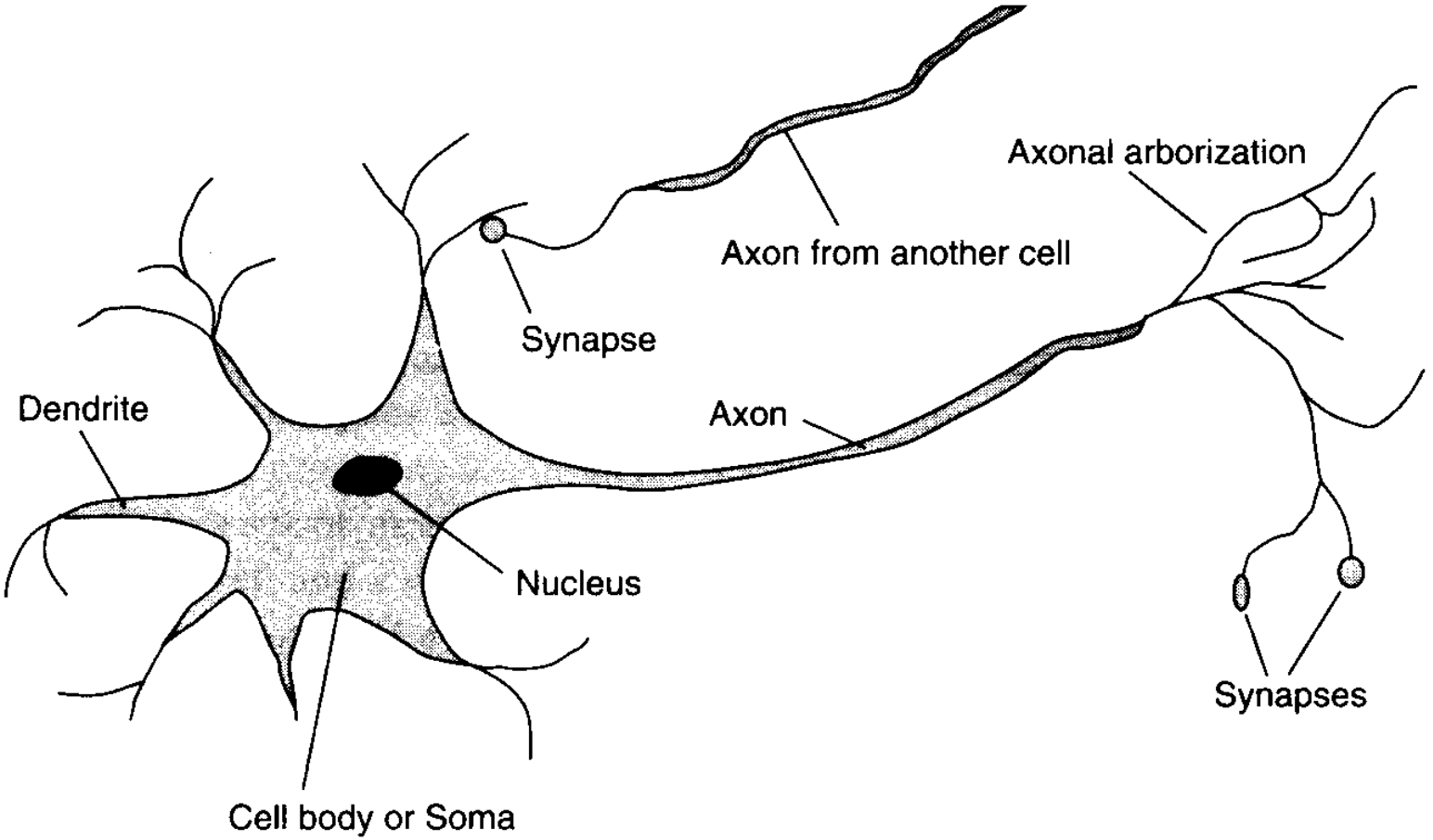
Human Brain

- Neuron structure
  - nucleus, cell body, axon, dendrite, synapses
- Neurotransmission
  - Neuron's output is encoded as a series of voltage pulses
    - called action potentials or spikes
  - by means of electrical impulse effected by chemical transmitter
  - Period of latent summation
    - generate impulse if total potential of membrane reaches a level : firing
  - excitatory or inhibitory

Models of Neuron

- Neuron is information processing unit
- A set of synapses or connecting links
  - characterized by weight or strength
- An adder
  - summing the input signals weighted by synapses
  - a linear combiner
- An activation function
  - also called squashing function
    - squash (limits) the output to some finite values

Biological Neuron



Excerpted from *Artificial Intelligence: Modern Approach* by S. Russel and P. Norvig

Sejarah Jaringan Syaraf Tiruan

- JST sederhana pertama kali diperkenalkan oleh **McCulloh dan Pitts tahun 1943**.
  - Menyimpulkan bahwa kombinasi beberapa neuron sederhana menjadi sebuah sistem neuron akan meningkatkan kemampuan komputasinya.
  - Bobot dalam jaringan diatur untuk melakukan fungsi logika sederhana
  - Fungsi aktivasi yang digunakan adalah fungsi threshold.
- **Tahun 1958, Rosenblatt** memperkenalkan model jaringan perceptron
  - Terdapat metode pelatihan untuk mengoptimalkan hasil iterasi.

Sejarah JST (2)

- **Widrow dan Hoff (1960)** mengembangkan perceptron dengan memperkenalkan aturan pelatihan jaringan (ADALINE = adative Linear Neuron).
  - Disebut aturan delta (kuadrat rata-rata terkecil)
  - Mengubah bobot perceptron apabila keluaran yang dihasilkan tidak sesuai dengan target yang diinginkan.
- Apa yang dilakukan peneliti sebelumnya hanya menggunakan single layer.
- **Rumelhart (1986)** mengembangkan perceptron menjadi backpropagation.
  - Jaringan di proses menjadi beberapa layer.

Aplikasi JST

- Pengenalan pola (pattern recognition)
  - Mengenal huruf, angka, suara, tanda tangan.
  - Menyerupai otak manusia yang masih mampu mengenali orang sudah beberapa lama tidak dijumpai (wajah/bentuk tubuh sedikit berubah)
- Pengolahan signal
  - Menekan noise dalam saluran telepon
- Peramalan (forecasting)
  - Meramal kejadian masa datang berdasarkan pola kejadian yang ada di masa lampau.
  - JST mampu mengingat dan membuat generalisasi dari apa yang sudah ada sebelumnya.

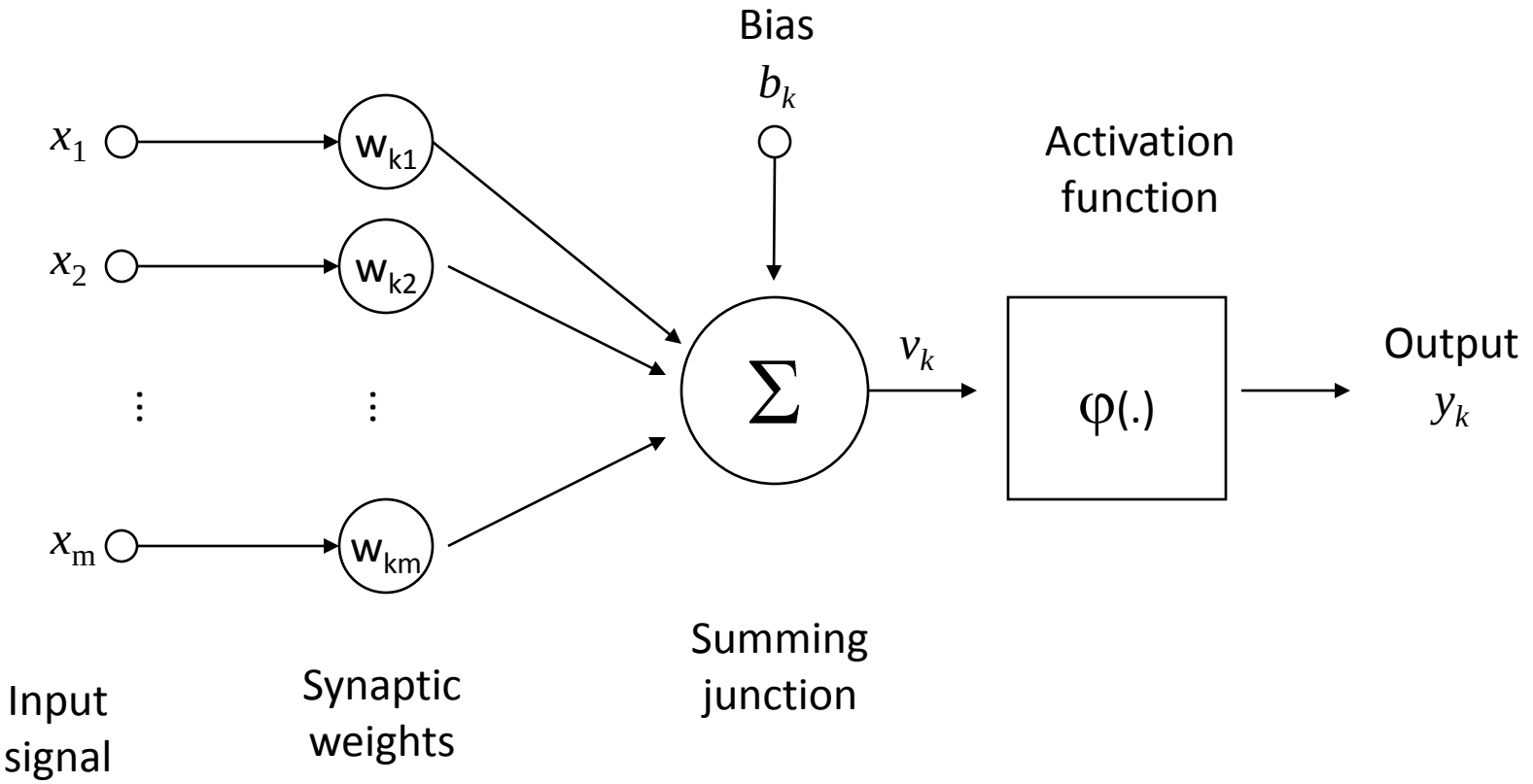
Kelebihan dan Kekurangan JST

- Kelebihan :
  - Banyak hal/aplikasi yang dapat dilakukan oleh JST.
- Kekurangan:
  - Ketidak akuratan hasil yang diperoleh.
  - Bekerja berdasarkan pola yang terbentuk pada inputnya.
  - Membutuhkan data pelatihan yang banyak.

Komponen ANN

- **Input:** data masukan beserta bobot-bobotnya.
- **Summation(Σ)**: menjumlahkan semua hasil perkalian nilai dengan bobotnya.
- **Fungsi aktifasi**: sebagai penentuan nilai ambang, jika terpenuhi maka signal diteruskan.
- **Output:** hasil dari komputasi.

Nonlinear model of a neuron

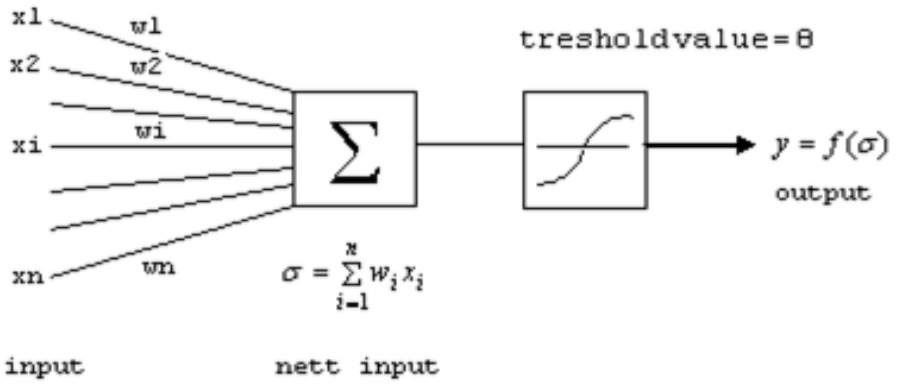


$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k$$

$$y_k = \varphi(v_k)$$

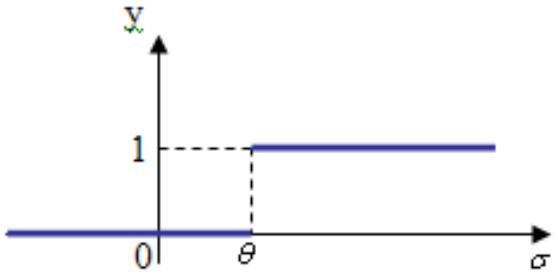
Ilustrasi Processing Unit ANN

SKEMATIK



Contoh :

$$y = f(\sigma) = \begin{cases} 1, \sigma > \theta \\ 0, \sigma \le \theta \end{cases}$$



Processing Unit ANN (cont'd)

- Input: data masukan dari processing unit sebelumnya atau dari luar.
- Bobot(weight): derajat pengaruh nilai input pada processing unit.
- Summation: weighted sum dari nilai input.

$$\sum w_{ij} x_{ij}$$

- Nilai ambang(threshold value): jika weighted sum melebihi nilai threshold maka signal akan ditransmisikan.
- Ouput: signal yang keluar lewat axon.

Proses Pembelajaran ANN

- Pembelajaran terawasi: jika output yang diharapkan telah diketahui sebelumnya.
- Pembelajaran tak terawasi: tidak memerlukan/tidak dapat ditentukan target output.

Pembelajaran Terawasi(supervised learning)

- **McCulloch dan Pitts (penemu)**
- **Hebb Rule.**
- **Perceptron.**
- **Delta rule. (ADALINE = Adative Linear Neuron)**
- **Backpropagation.**
- Heteroassociative Memory.
- Bidirection Associative Memory (BAM).
- Learning Vector Quantization (LVQ).

Pembelajaran tak terawasi (Unsupervised Learning)

- **Jaringan Kohonen**

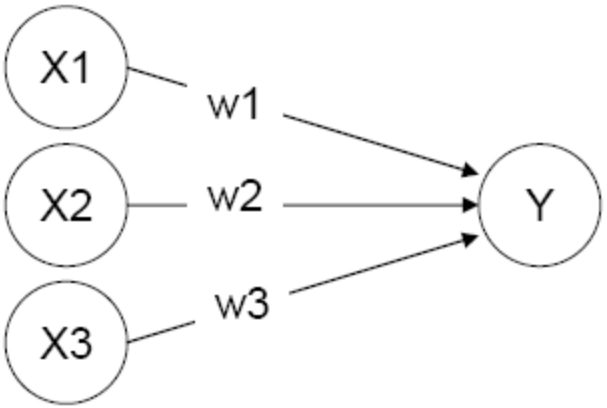
0. McCulloch dan Pitts

- Model JST pertama pada tahun 1943
- Menyimpulkan bahwa kombinasi beberapa neuron sederhana menjadi suatu sistem neuron dapat meningkatkan kemampuan komputasi.
- Digunakan untuk melakukan fungsi logika sederhana.
- Banyak prinsipnya yang masih digunakan pada JST saat ini.

Karakteristik McCulloch-Pitts

Karakteristik model neuron McCulloch-Pitts :

1. Fungsi aktivasi neuron dalam biner
   - Neuron meneruskan sinyal : aktivasi bernilai 1
   - Neuron tidak meneruskan sinyal : aktivasi bernilai 0
2. Setiap neuron pada jaringan dihubungkan dengan jalur terarah yang memiliki bobot tertentu
   - Jika bobot jalur bernilai positif, jalur diperkuat
   - Jika bobot jalur bernilai negatif, jalur diperlemah



Fungsi aktivasi unit Y :

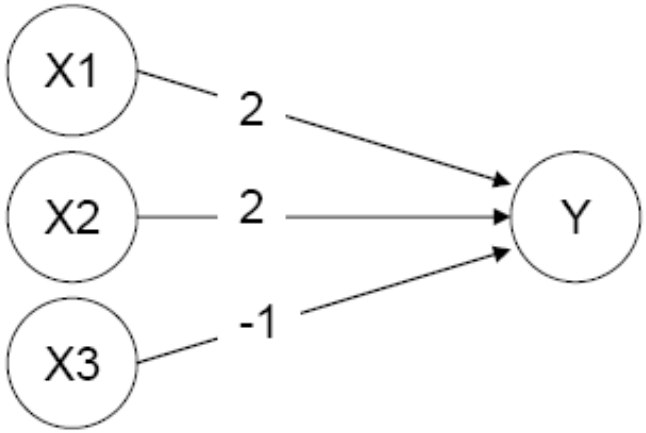
f(net)=1, if net >= θ, else 0

Dimana :

- net adalah total sinyal input = X1w1+X2w2+X3w3
- θ adalah nilai threshold Y

Karakteristik McCulloch-Pitts

3. Seluruh jalur bernilai positif yang terhubung ke neuron tertentu, memiliki bobot sama
   - Bobot koneksi yang berbeda dapat terhubung ke neuron lain
   - Setiap neuron memiliki nilai threshold tetap
4. Jika net input ke neuron lebih besar dari threshold, neuron akan meneruskan sinyal (fire)
   - Threshold di-set sedemikian sehingga setiap masukan yang memperlemah akan mencegah neuron meneruskan sinyal



- Ditentukan θ = 4
- net = 2.X1 + 2.X2 + (-1).X3
- f(net)=1, if net >= θ, else 0

Jaringan dapat meneruskan sinyal (*fire*) jika masukan X1 dan X2 bernilai 1 dan X3 bernilai nol

A. Hebb Rule

- Metode pembelajaran paling sederhana.
- Untuk menghitung nilai logika.
- Pembelajaran dengan cara memperbaiki nilai bobot yaitu bobot dinaikkan jika terdapat 2 neuron yang terhubung dan keduanya pada kondisi "on" pada saat yang sama.

$$w_i(baru) = w_i(lama) + x_i * y$$

w_i= bobot pada input ke-i.

x_i = input data ke-i

y = output data

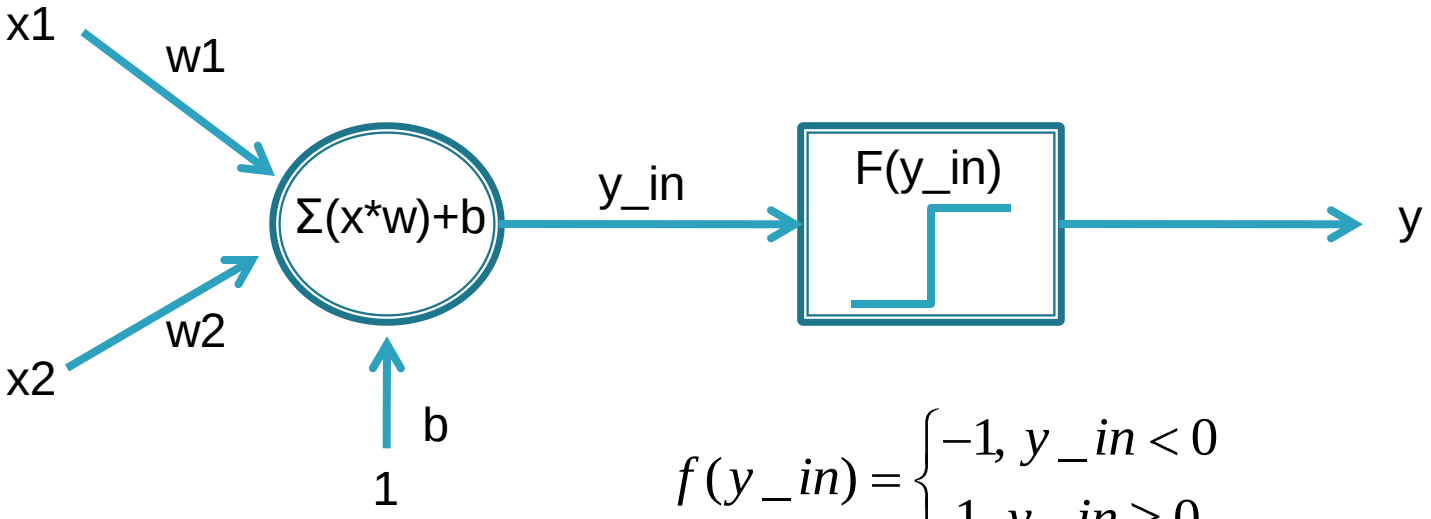
Contoh kasus

- Misalkan kita ingin membangun jaringan dengan kriteria sbb:

Input		Bias	Target/Output
x1	X2		
-1	-1	1	-1
-1	1	1	1
1	-1	1	1
1	1	1	1

Bobot awal dan bobot bias = 0

Arsitektur Jaringan



$$f(y\_in) = \begin{cases} -1, & y\_in < 0 \\ 1, & y\_in \geq 0 \end{cases}$$

- Langkah:
  - Hitung perubahan bobot untuk tiap data input.
  - Lakukan iterasi sebanyak data input.
  - Lakukan testing/pengujian.

Komputasi Hebb Rule

Perubahan bobot: $w1 = w_{awal} + x1 * y1$

Data ke-1

w1=0 + (-1)(-1) = 1

w2=0 + (-1)(-1)=1

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

Data ke-2

w1= 1 + (-1)(1) = 0

w2= 1 + (1)(1) = 2

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

Data ke-3

w1= 0 + (1)(1) = 1

w2= 2 + (-1)(1) = 1

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

Data ke -4

w1 = 1 + (1)(1) = 2

w2 = 1 + (1)(1) = 2

b = 1 + (1)(1) = 2

Pengujian:
Misal ambil data x=[-1,-1]

Maka

$$Y = 1 * b + x1 * w1 + x2 * w2$$

$$Y = 2 + (-1 * 2) + (-1 * 2) = -2$$

Berdasarkan fungsi aktifasi maka

$$f(y\_in) = \begin{cases} -1, & y\_in < 0 \\ 1, & y\_in \geq 0 \end{cases}$$

Target yang dihasilkan = -1 (cocok).

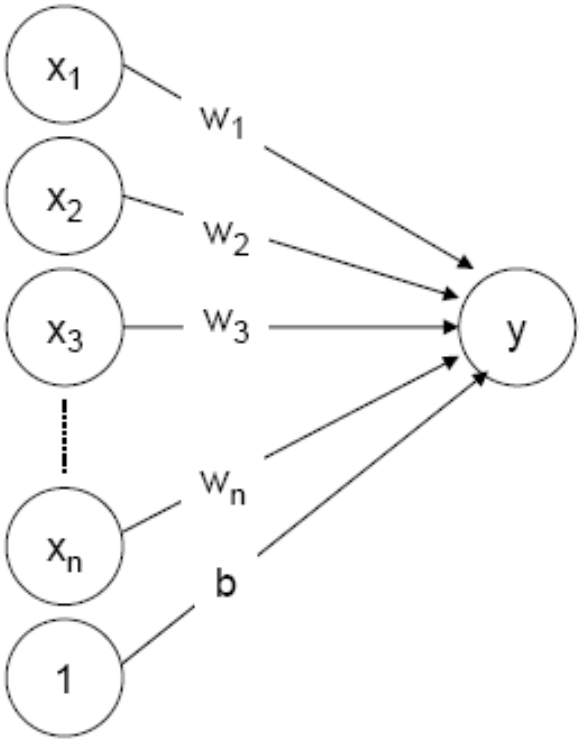
B Perceptron

- Jaringan perceptron menyerupai arsitektur jaringan Hebb.
- Diperkenalkan oleh Rosenblatt (1962) dan Minsky-Papert(1969).
- Model dengan aplikasi dan pelatihan yang terbaik pada masa tersebut.

Arsitektur Jaringan

- Jaringan satu layer
  - Beberapa neuron masukan dihubungkan langsung dengan sebuah neuron keluaran
  - Ditambah satu buah bias
- Fungsi aktivasi memiliki nilai -1, 0 dan 1

$$f(net) = \begin{cases} 1, net > \theta \\ 0, -\theta \le net \le \theta \\ -1, net < -\theta \end{cases}$$



Pelatihan Perceptron

Algoritma pelatihan

- Inisialisasi semua bobot, bias dan learning rate

  $w_i = 0$ (i=1, …, n), b=0, learning rate = nilai antara 0 s/d 1

- Selama ada elemen masukan s yang keluarannya tidak sama dengan target t, lakukan
  - Set aktivasi unit masukan : $x_i = s_i$ (i=1, …, n)
  - Hitung respon unit keluaran net dan y = f(net)
  - Bila y ≠ t, perbaiki bobot dan bias menurut persamaan :

    w_i(baru)=w_i(lama)+delta w (i=1, …, n),

    dimana delta w = $\alpha t\, x_i$

    b(baru) = b(lama) + α t

Pelatihan Perceptron

- Iterasi dilakukan terus menerus hingga seluruh keluaran sama dengan target yang ditentukan
  - Jaringan sudah memahami pola
  - Pada Hebb, iterasi berhenti setelah semua pola dimasukkan
- Perubahan bobot hanya dilakukan bila keluaran jaringan tidak sama dengan target
  - Yaitu bila $y = f(net) \neq t$
- Kecepatan iterasi ditentukan oleh laju pemahanan (learning rate, α)
  - Umumnya, $0 < \alpha < 1$
  - Semakin besar α, semakin sedikit iterasi yang diperlukan
  - Bila α terlalu besar dapat merusak pola yang sudah benar dan mengakibatkan pemahaman menjadi lama
- Satu siklus pelatihan yang melibatkan semua pola disebut epoch
  - Pada Hebb, pelatihan hanya dilakukan dalam satu epoch saja

Kelebihan Perceptron

- Seluruh pola yang dimasukkan dibandingkan dengan target yang sesungguhnya
  - Jika terdapat perbedaan, maka bobot akan dimodifikasi
  - Bobot tidak selalu dimodifikasi pada setiap iterasi
- Modifikasi bobot menggunakan laju pemahaman yang nilainya dapat diatur
  - Modifikasi bobot tidak hanya ditentukan oleh perkalian antara target dan masukan saja
- Pelatihan dilakukan secara terus menerus hingga jaringan dapat mengerti pola yang ditentukan
  - Teorema konvergensi perceptron menyatakan bahwa apabila ada bobot yang tepat, maka proses pelatihan akan konvergen ke bobot yang tepat tersebut

Kasus 1: Fungsi Logika

- Kasus :
  - Buat perceptron yang dapat menyatakan fungsi logika AND
- Representasi masukan/keluaran yang digunakan :
  - A. Masukan dan keluaran bipolar (-1 atau 1)
  - B. Masukan biner (0 atau 1) dan keluaran bipolar (-1 atau 1)
- Inisialisasi :
  - Bobot dan bias awal $w_i = 0$, $b = 0$
  - Learning rate $\alpha = 1$ (penyederhanaan)
  - Threshold $\theta = 0$

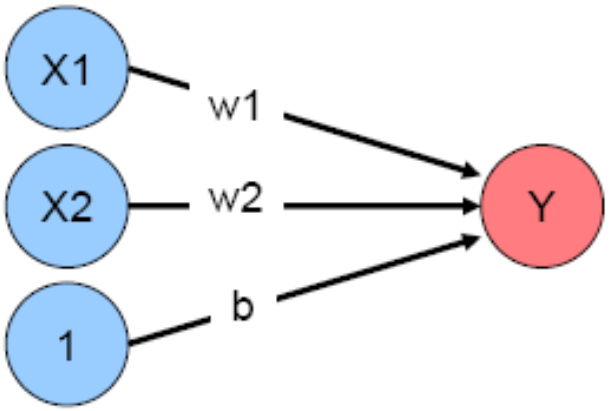
A. Representasi Bipolar

Tabel masukan dan target fungsi logika AND :

Masukan			Target
x1	x2	bias	t
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Masukan bipolar dan target bipolar

Arsitektur jaringan Perceptron :



Fungsi aktivasi untuk θ = 0 :

$$y = f(net) = \begin{cases} 1 & , net > 0 \\ 0 & , net = 0 \\ -1 & , net < 0 \end{cases}$$

Epoch Pertama (Bipolar)

Masukan			Target	Keluaran		Perubahan Bobot			Bobot Baru		
x1	x2	bias	t	net	f(net)	dw1	dw2	db	w1	w2	b
									0	0	0
1	1	1	1	0	0	1	1	1	1	1	1
1	-1	1	-1	1	1	-1	1	-1	0	2	0
-1	1	1	-1	2	1	1	-1	-1	1	1	-1
-1	-1	1	-1	-3	-1	0	0	0	1	1	-1

$$net = \sum_{i=1}^{2} x_i w_i + b$$

$$y = f(net) = \begin{cases} 1, net > 0 \\ 0, net = 0 \\ -1, net < 0 \end{cases}$$

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

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

- Epoch pertama terdiri dari empat iterasi
- Pada iterasi pertama-ketiga, keluaran y = f(net) tidak sama dengan target → bobot diubah.
- Pada iterasi keempat, nilai f(net) = -1 yang sama dengan target → bobot tidak diubah.
- Pada epoch pertama, belum seluruh f(net) sama dengan target → iterasi dilanjutkan pada epoch kedua

Epoch Kedua (Bipolar)

Masukan			Target	Keluaran		Perubahan Bobot			Bobot Baru		
x1	x2	bias	t	net	f(net)	dw1	dw2	db	w1	w2	b
									1	1	-1
1	1	1	1	1	1	0	0	0	1	1	-1
1	-1	1	-1	-1	-1	0	0	0	1	1	-1
-1	1	1	-1	-1	-1	0	0	0	1	1	-1
-1	-1	1	-1	-3	-1	0	0	0	1	1	-1

$$net = \sum_{i=1}^{2} x_i w_i + b$$

$$y = f(net) = \begin{cases} 1, net > 0 \\ 0, net = 0 \\ -1, net < 0 \end{cases}$$

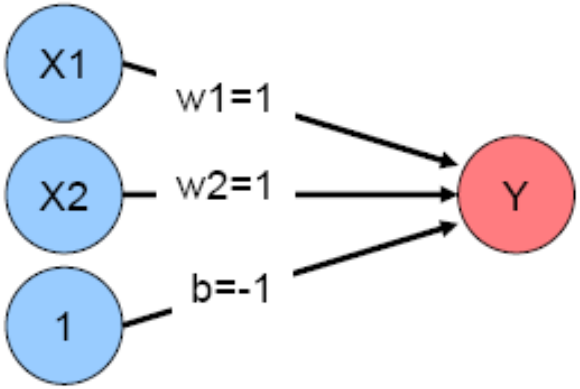
$w_i(baru) = w_i(lama) + \alpha t\, x_i$

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

- Bobot awal diperoleh dari epoch pertama
- Pada setiap iterasi dalam epoch kedua, semua pola f(net) sama dengan target t → tidak dilakukan perubahan bobot
- Jaringan sudah mengenal pola, iterasi dihentikan

Arsitektur Perceptron Diperoleh

Arsitektur jaringan Perceptron :



Tabel masukan dan target fungsi logika AND :

Masukan			Target
x1	x2	bias	t
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Masukan bipolar dan target bipolar

Fungsi aktivasi untuk $\theta = 0$:

$$y = f(net) = \begin{cases} 1 & , net > 0 \\ 0 & , net = 0 \\ -1 & , net < 0 \end{cases}$$

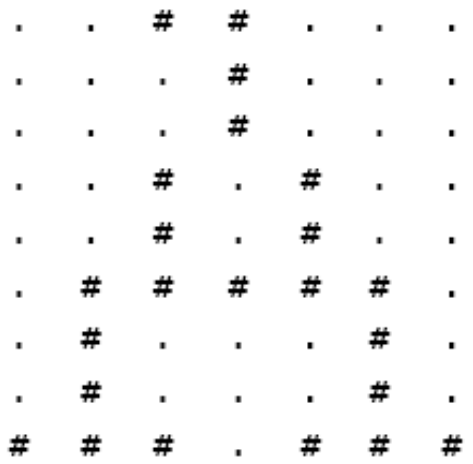
Pengenalan Pola Karakter

- Pengenalan pola karakter menggunakan perceptron
  - Pola masukan menyerupai huruf alfabet
  - Perceptron hendak dilatih untuk mengenal pola tersebut
- Algoritma pengenalan karakter
  - Nyatakan setiap pola masukan sebagai vektor bipolar yang elemennya adalah tiap titik dalam pola tersebut.
  - Berikan nilai target = 1 jika pola masukan menyerupai huruf yang diinginkan. Jika tidak, beri nilai target = -1.
  - Tentukan inisialisasi bobot, bias, learning rate dan threshold
  - Lakukan proses pelatihan perceptron

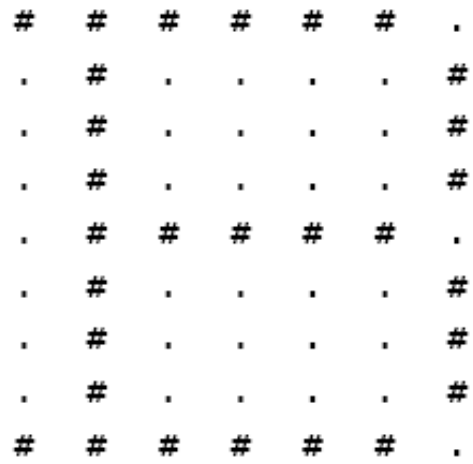
Kasus 2 : Pengenalan Karakter

Pengenalan sebuah pola karakter

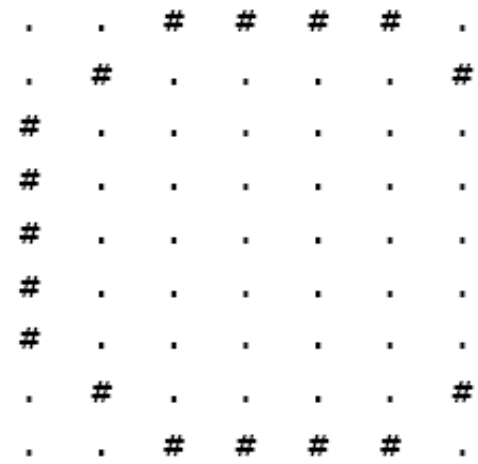
- Diketahui 6 buah pola masukan seperti pada slide berikut
- Buat model perceptron untuk mengenali pola menyerupai huruf "A"



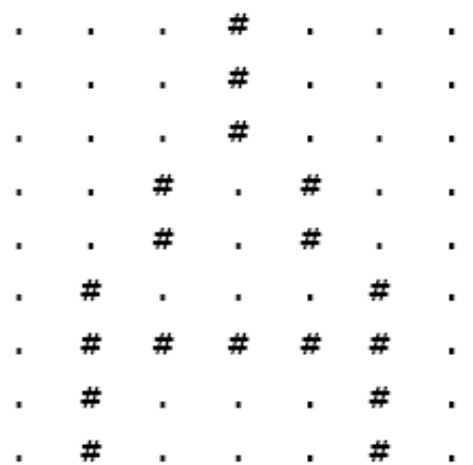
Pola 1



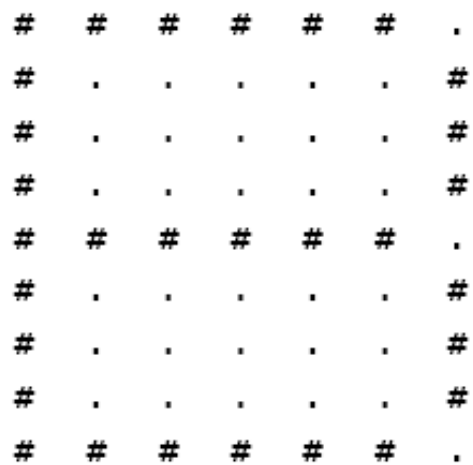
Pola 2



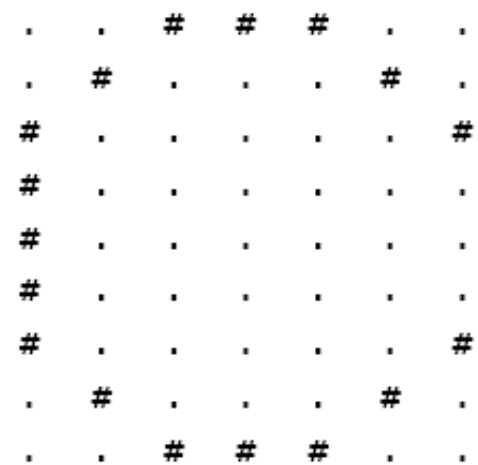
Pola 3



Pola 4



Pola 5



Pola 6

Representasi Kasus

- Setiap karakter pola dianggap sebagai sebuah unit masukan
  - Karakter "#" diberi nilai 1, karakter "." diberi nilai -1
  - Pembacaan pola dilakukan dari kiri ke kanan, dimulai dari baris paling atas

	1	2	3	4	5	6	7
1	.	.	#	#	.	.	.
2	.	.	.	#	.	.	.
3	.	.	.	#	.	.	.
4	.	.	#	.	#	.	.
5	.	.	#	.	#	.	.
6	.	#	#	#	#	#	.
7	.	#	.	.	.	#	.
8	.	#	.	.	.	#	.
9	#	#	#	.	#	#	#

→

-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

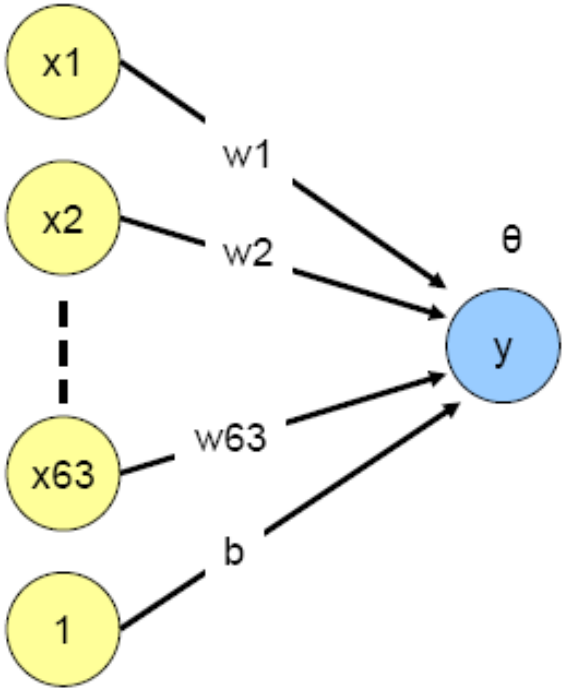
- Setiap pola terdiri dari 9 baris dan 7 kolom
  - Perceptron terdiri dari 63 unit masukan (x1 s/d x63) dan sebuah bias bernilai = 1

Representasi Kasus

- Target
  - Keluaran jaringan bernilai 1 jika diberi masukan menyerupai huruf "A" dan bernilai -1 jika tidak menyerupai huruf "A"
  - Pola yang menyerupai huruf "A" adalah pola 1 dan 4.

Pola Masukan	Target
Pola 1	1
Pola 2	-1
Pola 3	-1
Pola 4	1
Pola 5	-1
Pola 6	-1

Arsitektur Perceptron

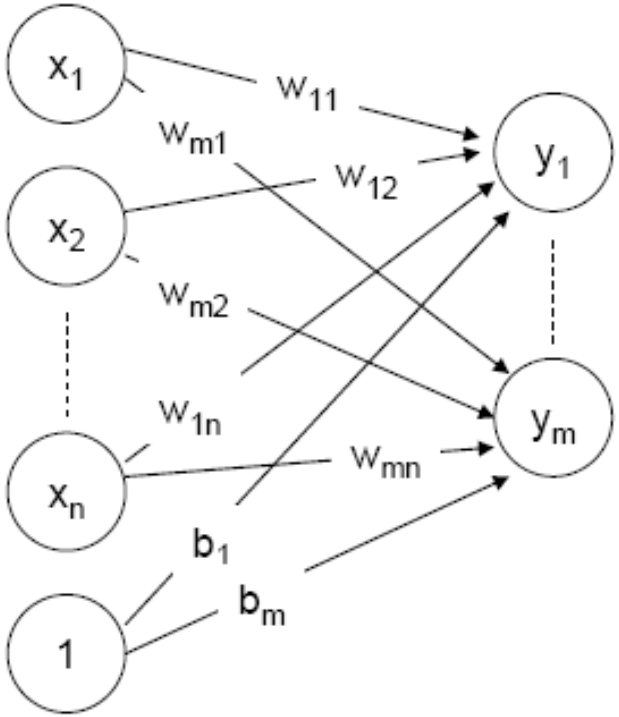


Perceptron memiliki 63 unit masukan, Sebuah bias dan sebuah unit keluaran

- Asumsi parameter :
  - Bobot awal = 0
  - Learning rate α = 1
  - Threshold θ = 0,5
- Pelatihan dilakukan dengan memasukkan seluruh pola huruf
- Hitung
  - $$net = \sum_{i=1}^{63} x_i w_i + b$$
  - $$y = f(net) = \begin{cases} 1, & net > 0,5 \\ 0, & -0,5 \le net \le 0,5 \\ -1, & net < 0,5 \end{cases}$$
- Bila f(net) ≠ target, bobot dan bias diubah
- Proses pelatihan terus dilakukan hingga semua keluaran sama dengan target.

Pengenalan Beberapa Karakter

- Pengenalan beberapa pola sekaligus dilakukan dengan menggabungkan beberapa perceptron
  - Terdapat beberapa unit keluaran
- Setiap unit masukan dan bias dihubungkan dengan setiap target
  - Bobot koneksi dari unit x_i ke y_j diberi label w_{ji}
  - Bobot bias diberi label b_1, b_2, ... b_m



Algoritma Pelatihan

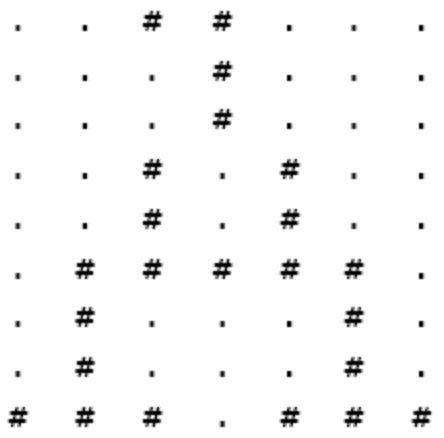
- Algoritma pelatihan perceptron untuk pengenalan beberapa pola :
  - Nyatakan tiap pola masukan sebagai vektor bipolar yang elemennya adalah tiap titik dalam pola tersebut
  - Berikan nilai target $t_j = 1$ jika pola masukan menyerupai huruf yang diinginkan, dan berikan nilai $t_j = -1$ jika sebaliknya (j=1, 2, ...m)
  - Inisialisasi semua bobot, bias dan learning rate
  - Lakukan proses pelatihan perceptron seperti dibahas sebelumnya
    - Untuk setiap unit keluaran, hitung respon unit keluaran net dan $y_j=f(net_j)$
    - Perbaiki bobot pola bila respon keluaran tidak sama dengan target ($y_j \neq t_j$) menurut persamaan

      $w_{ji}(baru)=w_{ji}(lama)+ \alpha t_j x_i$

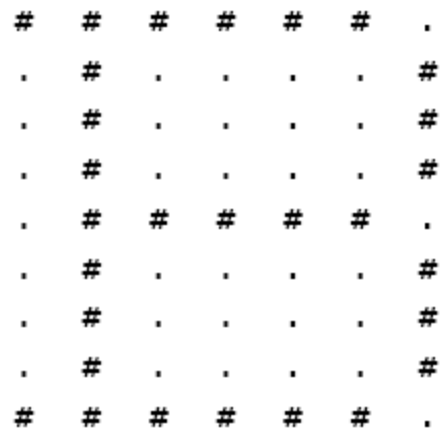
      $b_j(baru) = b_j(lama) + \alpha t_j$
    - Lakukan proses pelatihan hingga $y_j = t_j$ (j=1, 2, ... m)

Kasus 3 : Pengenalan Pola “A B C”

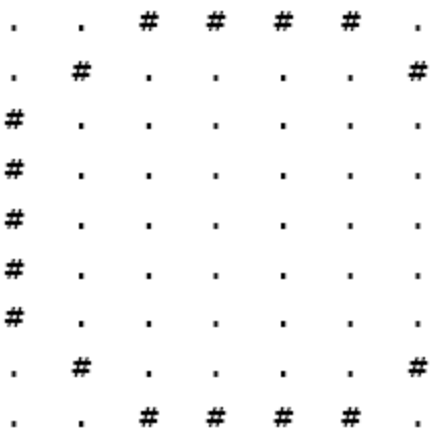
- Bila diketahui 6 buat pola masukan seperti pada kasus 2, buat model perceptron untuk mengenali pola menyerupai huruf “A”, “B” atau “C”.



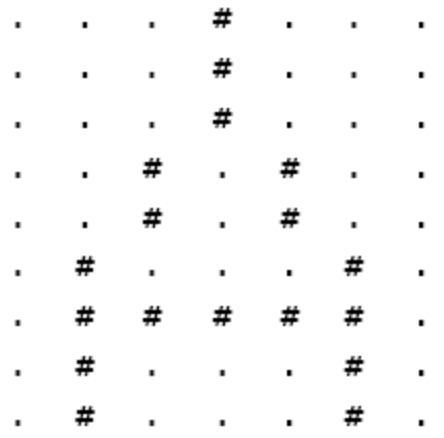
Pola 1



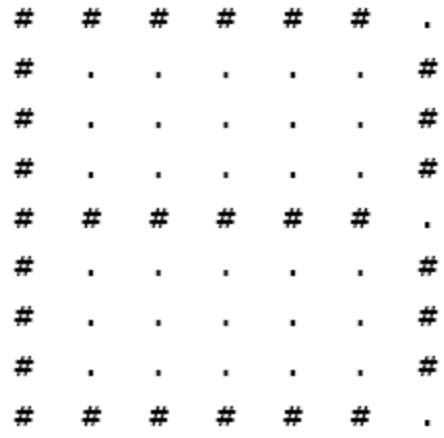
Pola 2



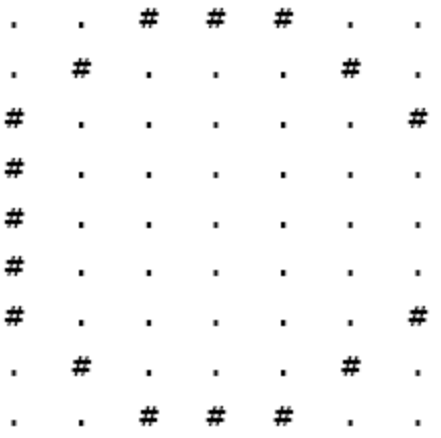
Pola 3



Pola 4



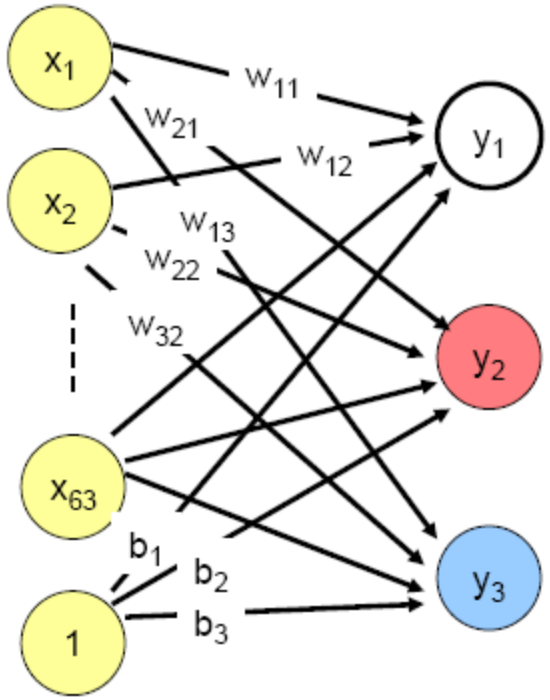
Pola 5



Pola 6

Arsitektur Perceptron

Arsitektur jaringan perceptron pengenalan pola menyerupai huruf A, B dan C dengan 6 pola masukan



Tabel pasangan pola target

Pola Masukan	Target		
	t1	t2	t3
Pola 1	1	-1	-1
Pola 2	-1	1	-1
Pola 3	-1	-1	1
Pola 4	1	-1	-1
Pola 5	-1	1	-1
Pola 6	-1	-1	1

Tabel Pelatihan

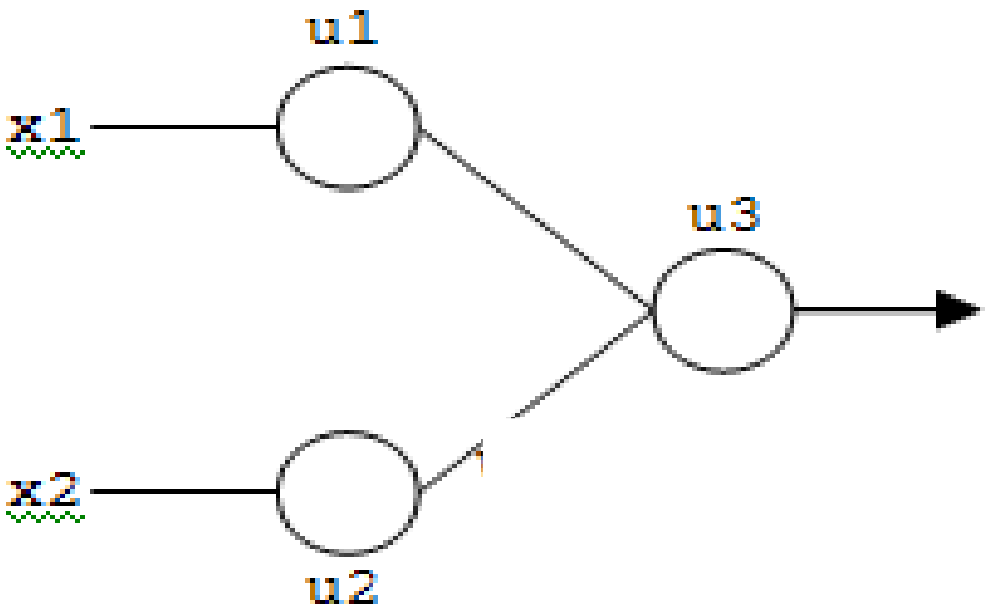
Masukan		Target	Keluaran		Perubahan Bobot		Bobot Baru	
x1-x63	bias	t1-t3	net1-net3	f(net1)-f(net3)	dw11-dw1.63, dw21-dw2.63, dw31-dw3.63	db1-db63	w11-w1.63, w21-w2.63, w31-w3.63	b1-b63
							00.....00	0..0
...	...	...	...	...	...	...	...	...

Terdiri dari :

- 63 unit masukan dan sebuah bias b (x1, x2, ... x63)
- 3 kolom target (t1, t2 dan t3)
- 3 kolom net (net1, net2 dan net3)
- 3 kolom fungsi aktivasi (y1=f(net1), y2=f(net2), y3=f(net3))
- 3*63 kolom perubahan bobot
  - dw11,dw12, ..., dw1.63, dw21, dw22, ... dw2.63, dw31, dw32, ... dw3.63
- 3*63 kolom bobot
  - w11,w12, ...,w1.63, w21, w22, ... w2.63, w31, w32, ... w3.63

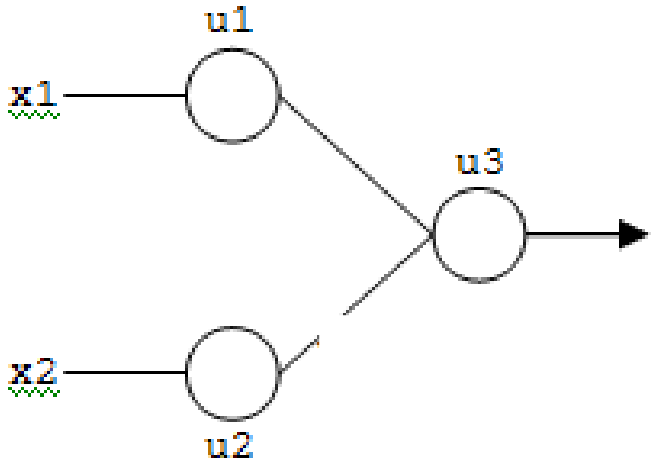
C. Perceptron dengan Delta Rule

- Termasuk bentuk jaringan syaraf yang sederhana.
- Jaringan dengan satu layer.



Kasus Awal

Input		Output		
X1	X2	OR	AND	XOR
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

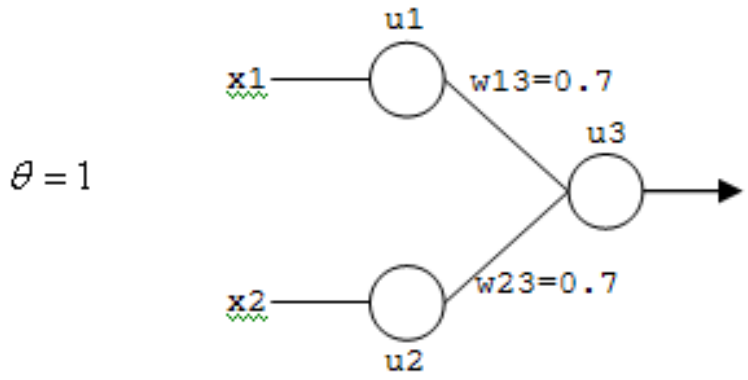


- Misalkan ANN akan digunakan untuk melakukan logical AND, OR dan XOR.
- Telah dibangun jaringan neural spt terlihat diatas
- **Misalkan bobot diberikan untuk tiap kasus**
- **Nilai threshold (θ) = 1**

$$y = f(\sigma) = \begin{cases} 1, \sigma > 1 \\ 0, \sigma \leq 1 \end{cases}$$

Implementasi AND

AND

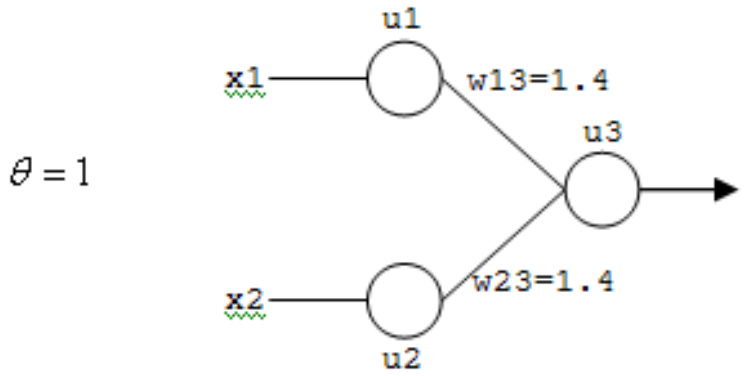


$$y = f(\sigma) = \begin{cases} 1, \sigma > 1 \\ 0, \sigma \leq 1 \end{cases}$$

INPUT	σ	OUTPUT
(x1=1,x2=1)	σ = (0.7)(1)+(0.7)(1)=1.4 > θ	1
(x1=0,x2=1)	σ = (0.7)(0)+(0.7)(1)=0.7 < θ	0
(x1=1,x2=0)	σ = (0.7)(1)+(0.7)(0)=0.7 < θ	0
(x1=0,x2=0)	σ = (0.7)(0)+(0.7)(0)=0.0 < θ	0

Implementasi OR

OR



$$y = f(\sigma) = \begin{cases} 1, \sigma > 1 \\ 0, \sigma \leq 1 \end{cases}$$

INPUT	σ	OUTPUT
(x1=1, x2=1)	σ = (1.4)(1)+(1.4)(1)=2.8 > θ	1
(x1=1, x2=0)	σ = (1.4)(1)+(1.4)(0)=1.4 > θ	1
(x1=0, x2=1)	σ = (1.4)(0)+(1.4)(1)=1.4 > θ	1
(x1=0, x2=0)	σ = (1.4)(0)+(1.4)(0)=0.0 < θ	0

Pertanyaan ????

1. Bagaimana kita tahu bobot yang tepat untuk menghasilkan output seperti yg kita inginkan?

   Dengan training, kita bisa melakukan inisialisasi weight sembarang.

2. Bagaimana untuk operasi XOR ?

   dengan konfigurasi jaringan neural seperti itu (2 layer) tidak akan mendapatkan hasil XOR. Lantas bagaimana cara mendapatkannya ?

Training Phase

- Adalah mengatur kembali besarnya bobot sebagai jawaban atas ketidaktepatan hasil (output).
- Yang dirubah pada training hanyalah bobot, nilai aktivasi/threshold value, susunan input tidak diubah.
- Prinsip training: Besarnya kontribusi link thd error menentukan besarnya perubahaan bobot pada link tsb.
- Gunakan delta rule:

$$\Delta w_{ij} = c(g_j - a_j)a_i$$

Modifikasi Bobot

- Bobot dimodifikasi sedemikian hingga error yang terjadi minimum
- Error adalah kuadrat selisih antara target t dan keluaran jaringan y = f(net) :

$$Error = E = (t - f(net))^2 = \left(t - \left(\sum_i x_i w_i + b \right) \right)^2$$

  - Error merupakan fungsi dari bobot wi
- Kecepatan penurunan Error :

$$\frac{\partial E}{\partial w_i} = -2\left(t - \left(\sum_i x_i w_i + b \right) \right) x_i = -2(t - y)x_i$$

- Maka perubahan bobot :

$$\Delta w_i = \alpha(t - y)x_i$$

  - α adalah bilangan positif bernilai kecil (α umumnya = 0,1)

Training Phase (cont'd)

- Gunakan delta rule: $\Delta w_{ij} = c(g_j - a_j)a_i$

  Dimana:

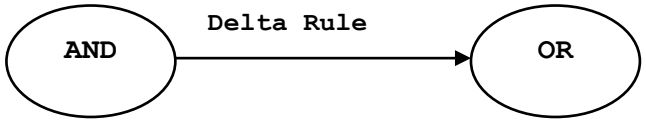
Δw_{ij} : Perubahan bobot pada link antara unit i dan j

c : learning constant (konstanta pembelajaran)

g_j : hasil yang diharapkan dari unit j (target)

a_j : hasil(ouput) yang didapat dari unit j

a_i : hasil(ouput) yand didapat dari unit i



$$\Delta w_{ij} = c(g_j - a_j) \cdot a_i$$

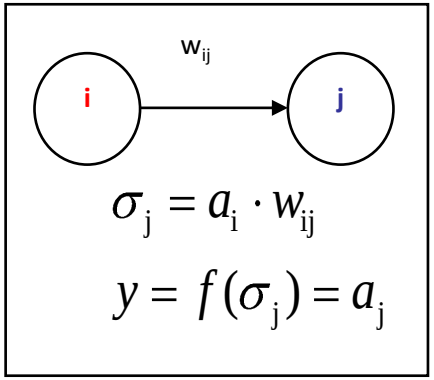
c = konstanta (learning rate),

biasanya c = 0.35 berhasil

g_j = nilai target unit j

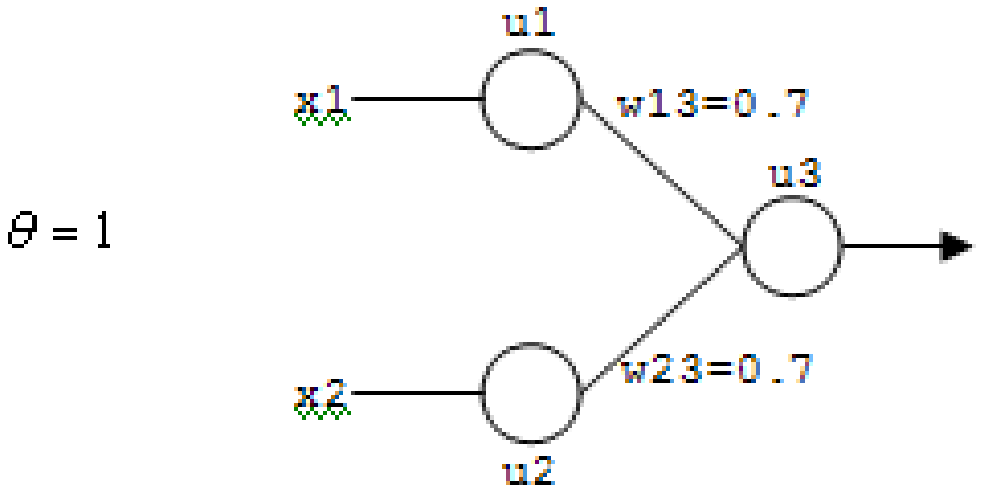
a_j = nilai output unit j

a_i = nilai output unit i



Implementasi Training Menggunakan delta rule

Lakukan Implementasi Ann
Dengan Bobot Seperti Pada Gambar
Untuk Mendapatkan Target (Or)



OR		
INPUT		TARGET
1	1	1
1	0	1
0	1	1
0	0	0

Output

1

0

0

0

Implementasi Training Menggunakan delta rule

Misal: Parameter c= 0.35

w_{13} dan w_{23} = 0.7

$$\Delta w_{ij} = c(g_j - a_j)a_i$$

g_j

(1,1) ---→ 1

$\Delta w_{13} = (0.35)(1-1)(1) = 0$

$\Delta w_{23} = (0.35)(1-1)(1) = 0$

Dihasilkan :

$w_{13} = 0.7 + 0 = 0.7$

$w_{23} = 0.7 + 0 = 0.7$

$$\Delta w_{ij} = c(g_j - a_j)a_i$$

g_j

(1,0) ---→ 1 $\quad \Delta w_{13} = (0.35)(1-0)(1) = 0.35$

$\Delta w_{23} = (0.35)(1-0)(0) = 0$

Dihasilkan : $\quad w_{13} = 0.7 + 0.35 = 1.05$

$w_{23} = 0.7 + 0 = 0.7$

g_j

(0,1) ---→ 1 $\quad \Delta w_{13} = (0.35)(1-0)(0) = 0$

$\Delta w_{23} = (0.35)(1-0)(1) = 0.35$

Dihasilkan : $\quad w_{13} = 1.05 + 0 = 1.05$

$w_{23} = 0.7 + 0.35 = 1.05$

g_j

(0,0) ---→ 0 $\quad \Delta w_{13} = (0.35)(0-0)(0) = 0$

$\Delta w_{23} = (0.35)(0-0)(0) = 0$

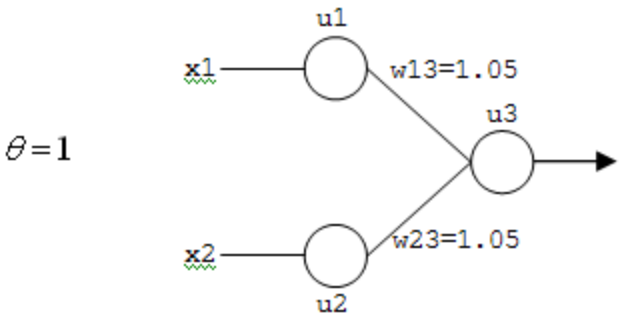
Dihasilkan : $\quad w_{13} = 1.05 + 0 = 1.05$

$w_{23} = 1.05 + 0 = 1.05$

Hasil

- Jadi diperoleh
  - $W_{13} = 1.05$
  - $W_{23} = 1.05$
- Update bobot pada jaringan, lakukan perhitungan

OR



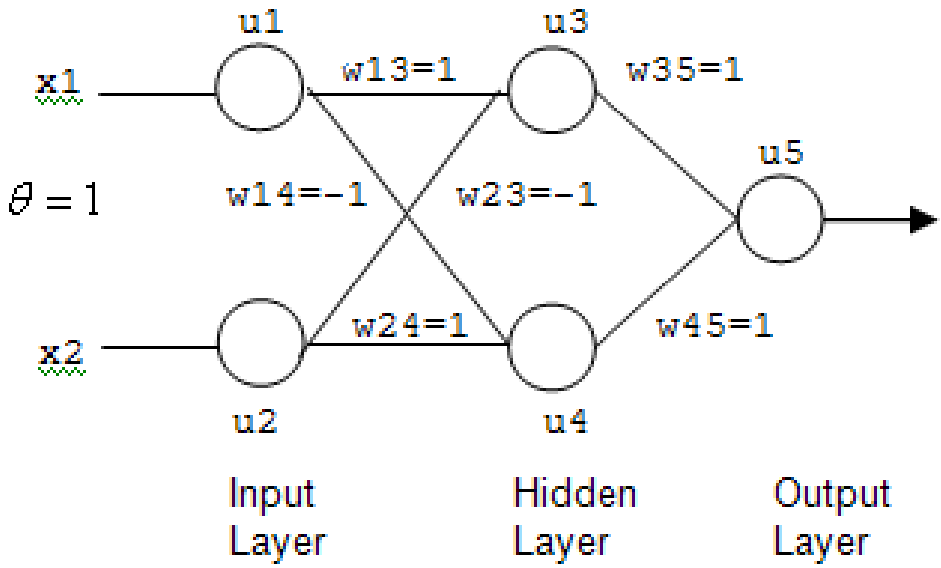
INPUT	σ	OUTPUT
(x1=1,x2=1)	σ = (1.05)(1)+(1.05)(1)=2.1 > θ	1
(x1=1,x2=0)	σ = (1.05)(1)+(1.05)(0)=1.05 > θ	1
(x1=0,x2=1)	σ = (1.05)(0)+(1.05)(1)=1.05 > θ	1
(x1=0,x2=0)	σ = (1.05)(0)+(1.05)(0)=0.0 < θ	0

- Lanjutkan iterasi kembali jika belum menghasilkan sesuai dengan target

XOR network dengan 1 Hidden Layer

XOR

XOR		
INPUT		OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0



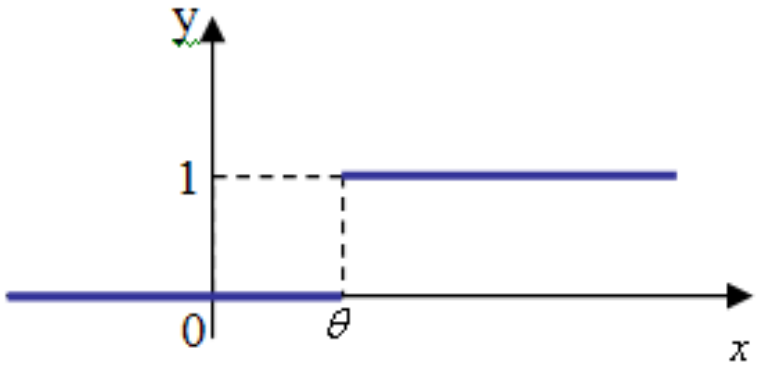
INPUT	σ	OUTPUT
(x1=1, x2=1)	Input untuk u3: σ3 = w13.x1+w23.x2 = (1)(1)+(-1)(1)=0 < θ Input untuk u4: σ4 = w14.x1+w24.x2 = (-1)(1)+(1)(1)=0 < θ Input untuk u5: σ5 = w35.y3+w45.y4 = (1)(0) +(1)(0)=0 < θ	Output untuk u3: y3 = f(σ3) = 0 Output untuk u4: y4 = f(σ4) = 0 Output untuk u5: y5 = f(σ5) = 0
(x1=0, x2=1)	Input untuk u3: σ3 = w13.x1+w23.x2 = (1)(0)+(-1)(1)=-1 < θ Input untuk u4: σ4 = w14.x1+w24.x2 = (-1)(0)+(1)(1)=1 = θ Input untuk u5: σ5 = w35.y3+w45.y4 = (1)(0) +(1)(1)=1 = θ	Output untuk u3: y3 = f(σ3) = 0 Output untuk u4: y4 = f(σ4) = 1 Output untuk u5: y5 = f(σ5) = 1

Key points

- Proses menghitung untuk mendapatkan output disebut **FEED FORWARD**.
- Proses mengubah bobot di tiap link disebut **LEARNING**.
- Algorithma standard untuk proses learning adalah Back Propagation.

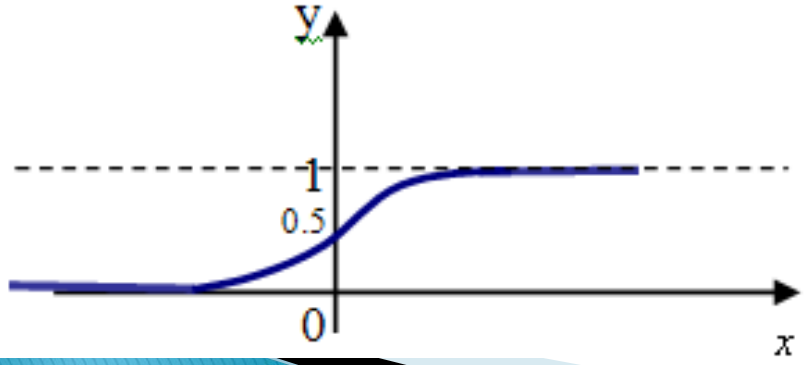
Fungsi Aktivasi

Fungsi Heaviside



$$y = f(x) = \begin{cases} 1, x > \theta \\ 0, x \le \theta \end{cases}$$

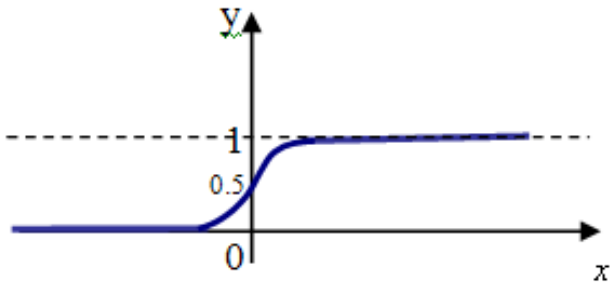
Fungsi Sigmoid



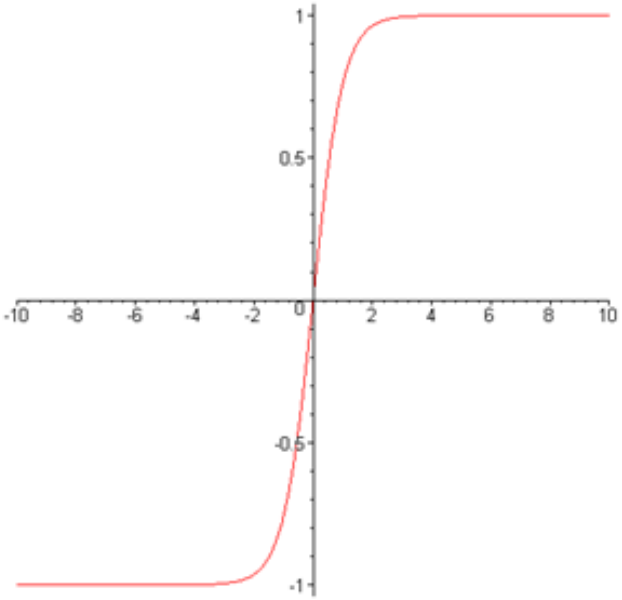
$$y = f(x) = \frac{1}{1+e^{-x}}$$

$$f'(x) = f(x)(1-f(x))$$

Fungsi Aktivasi



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



Fungsi f(x) = tanh(x)

$$y = f(x) = \tanh(x)$$

$$= \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = 2\frac{1}{1 + e^{-2x}} - 1$$

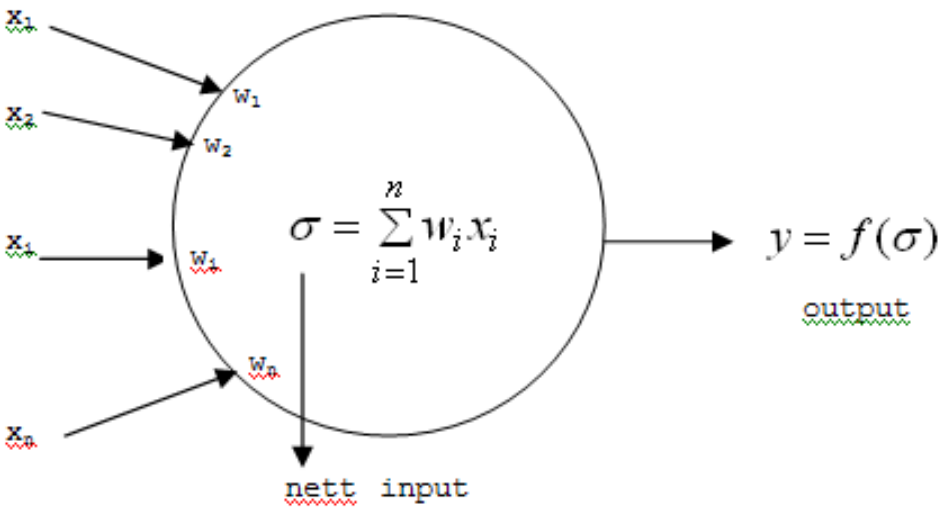
$$(\tanh(x))' = 1 - (\tanh(x))^{2}$$

Fungsi Aktifasi

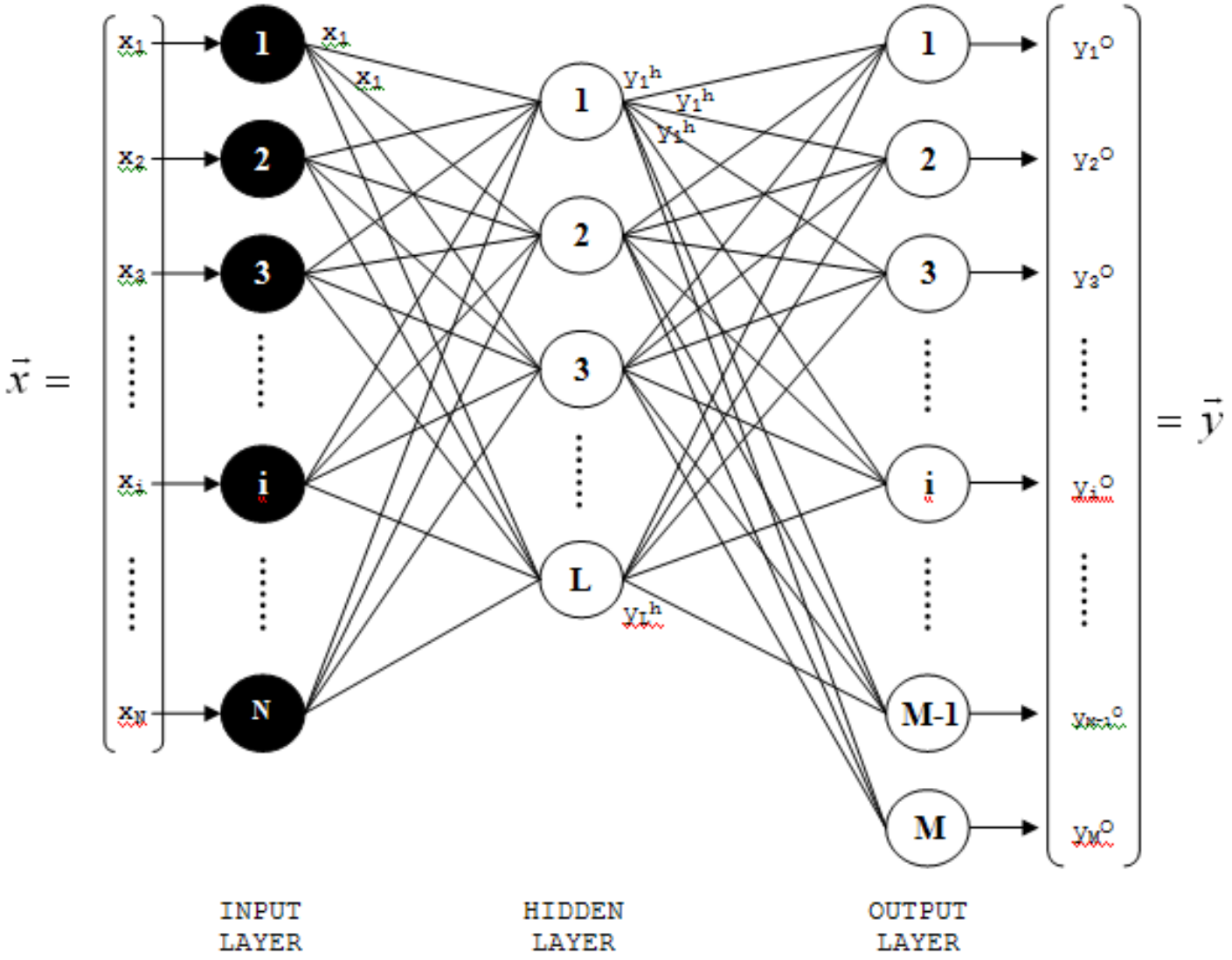
- Biasanya pilihannya adalah fungsi sigmoid dan tanh(x), keuntungannya fungsi-fungsi tersebut dapat diturunkan dengan mudah.

D. NN multi layer dengan Backpropagation

Representasi Lain:



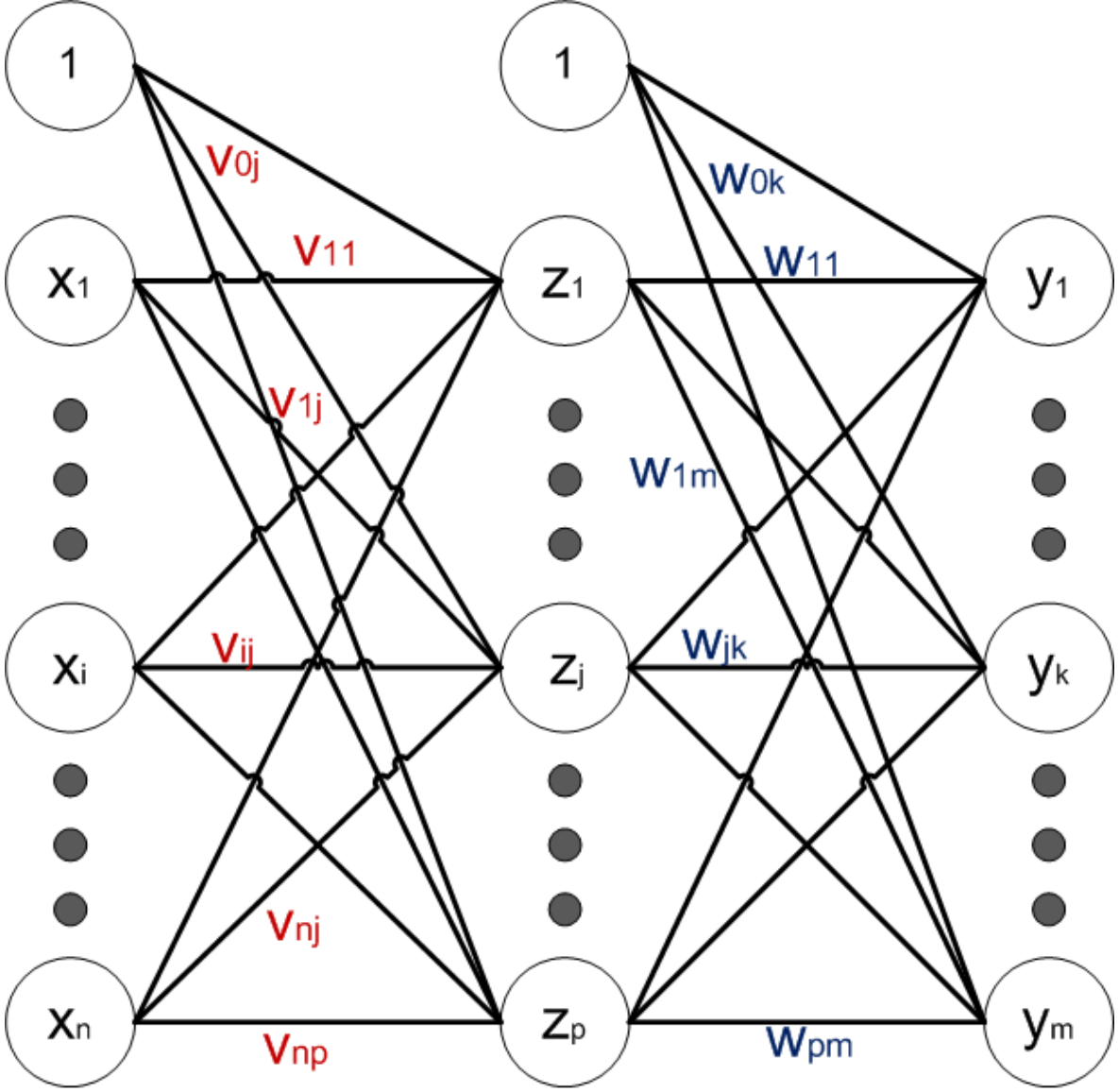
Gambar lingkaran - lingkaran putih di bawah menunjukkan diagram seperti yang dijelaskan di atas:


$\vec{x} =$
$= \vec{y}$
x1
x2
x3
xi
xN
1
2
3
i
N
x1
x1
1
2
3
L
y1h
y1h
y1h
yLh
1
2
3
i
M-1
M
y1O
y2O
y3O
yiO
yM-1O
yMO
INPUT LAYER
HIDDEN LAYER
OUTPUT LAYER
tidak ada bobot, hanya untuk menunjukkan layer.

Back Propagation

- Merupakan algorithma pebelajaran untuk melakukan update bobot.
- Biasanya digunakan untuk multilayer (input, hidden, dan output).
- Langkah komputasi: lakukan proses maju (forward propagation), kemudian update error dari bobot dengan backward propagation.
- Biasanya menggunakan fungsi aktifasi sigmoid. $y = f(x) = \frac{1}{1+e^{-x}}$

$$f'(x) = f(x)(1 - f(x))$$



$$X_i \xrightarrow{\Sigma} Z\_in\ j \xrightarrow{f(\sigma)} Z_j \xrightarrow{\Sigma} y\_in\ k \xrightarrow{f(\sigma)} y_k$$

Algorithma ANN Backpropagation (1)

- Inisialisasi bobot (ambil acak) → jangan terlalu besar/kecil biasanya [-0.5;0.5], [-1,1].
- Tetapkan: maksimum epoh, target error, learning rate (α).
- Inisialisasi: epoh = 0, MSE (mean square error) =1.
- Kerjakan langkah berikut selama (epoh < maks epoh) dan MSE > target error.
  1. Epoh = Epoh + 1
  2. Untuk tiap – tiap pasangan elemen yang akan dilakukan pembelajaran, lakukan :
  3. Feed Forward.
  4. Back Propagation.
  5. Hitung MSE
- Cek kondisi pemberhentiaan

Algorithma (2)

- Feed Forward:
  a. Setiap unit input (x_i, i=1,..n) menerima input x_i dan meneruskannya ke semua hidden layer.
  b. Hitung:

  c. Hitung: $$z\_in\,j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \;\; dan \;\; z_j = f(z\_in\,j), (j = 1, \ldots, p)$$

$$y\_ink = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \;\; dan \;\; y_k = f(y\_ink), (k = 1, \ldots, m)$$

Algorithm (3)

- Back Propagation of Error
  a. Tiap unit output menerima target t_k (k=1,…,m)
     hitung:
     $$\varsigma_{2k} = (t_k - y_k).f'(y\_ink), (k = 1,...,m)$$
     hitung koreksi bobot: $\Delta w_{jk} = \alpha.\varsigma_{2k}.z_j$
     hitung koreksi bobot pada bias : $\Delta w_{0k} = \alpha.\varsigma_{2k}$
  b. Hitung:

$$\varsigma\_inj = \sum_{k=1}^{m} \varsigma_{2k}.w_{jk}$$

$$\varsigma_{1j} = \varsigma\_inj.f'(z\_inj)$$

$$\Delta v_{ij} = \alpha.\varsigma_{1j}.x_i \ (koreksi\ bobot(input -> hidden)$$

$$\Delta v_{0j} = \alpha.\varsigma_{1j}(koreksi\ bobot\ bias(input -> hidden)$$

Algorithm (4)

c. Update bobot dan bias

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

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