MACHINE LEARNING

Perceptron

ernabrn@usu.ac.id

Weight

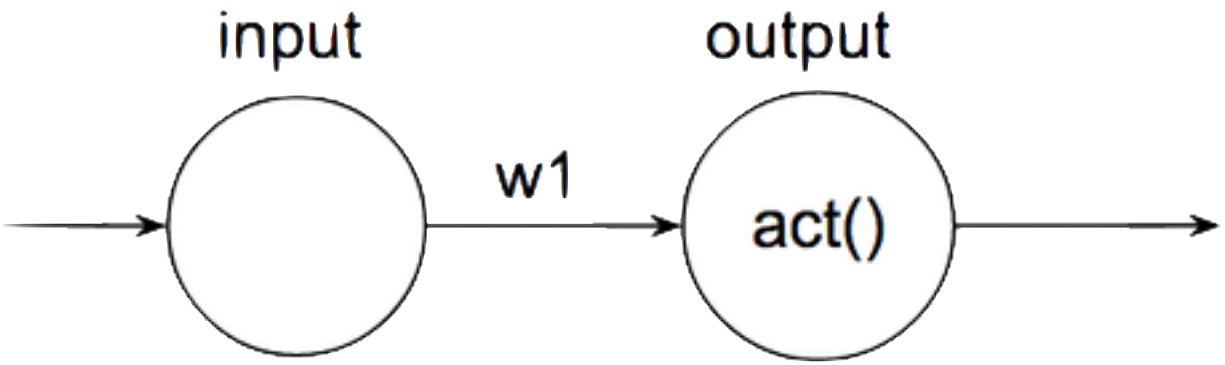
Dalam NN, **weight** adalah parameter yang “dipelajari” selama pelatihan. Digunakan untuk menentukan seberapa cepat aktivasi fungsi akan ‘bergerak’

Contoh: (dg sigmoid fn)

BIAS

- Unit bias adalah "ekstra" neuron yang ditambahkan ke setiap lapisan pra-output yang menyimpan nilai 1.
- **Bias** digunakan untuk mengatur (to adjust) jumlah input-weight ($\sum_i^n x_i w_i$) pada neuron

  Ilustrasi: act() adalah aktivasi fungsi (step function)

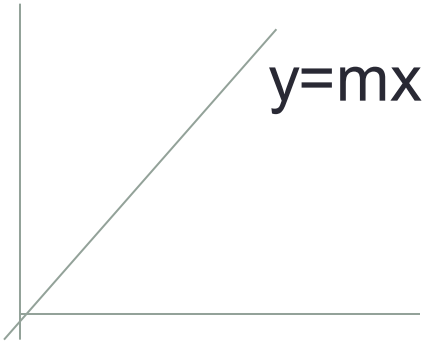


BIAS

- Contoh:

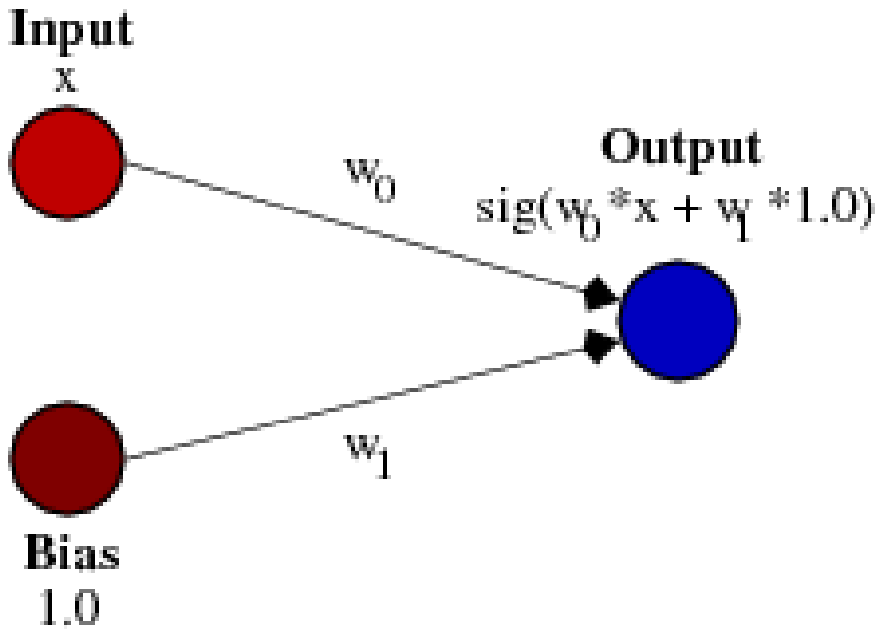
net = $(\sum_i^n x_i w_i)$+ b → net = wx+b (garis lurus)

Jika b=0 (tidak ada bias), hyperplane yang didapat adalah

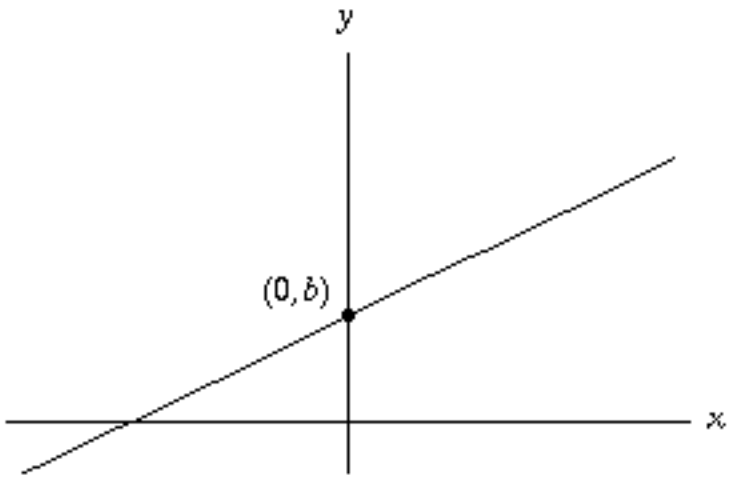


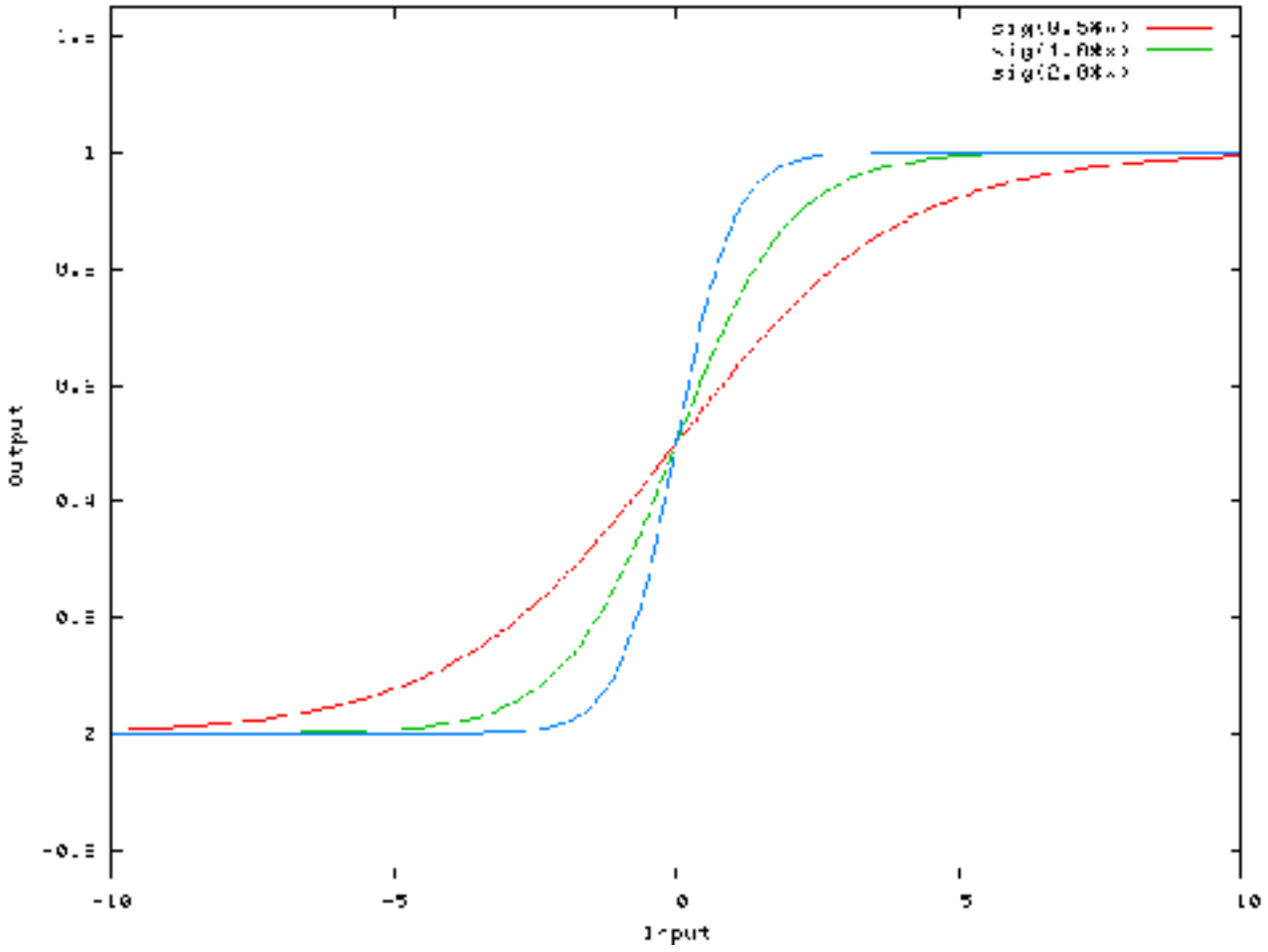
Karena tidak adanya bias, model akan melatih melewati titik asal saja, yang tidak sesuai dengan skenario dunia nyata.

- Karena itu, perlu ”ditambahkan" istilah input tambahan berupa nilai konstanta, yaitu 1. Satu dikalikan dengan nilai apa pun adalah nilai itu sendiri. Ini disebut unit bias.



- Istilah konstan ini, juga disebut "istilah intersep" (seperti yang ditunjukkan oleh contoh linear), menggeser fungsi aktivasi ke kiri atau ke kanan. Ini juga akan menjadi output ketika inputnya nol.




output
input
-10
-5
0
5
10
1
2

Perceptron

- Perceptron adalah sebuah algoritma untuk pengklasifikasi terawasi (supervised classification) yang memetakan sebuah input *x* kepada satu (atau beberapa) non binary output *f(x)*

- Perceptron dapat dilatih untuk menyelesaikan two-class classification problem dimana kelas-kelas tersebut terpisah secara linier (linearly separable).

- Dalam problem dua dimensi (dimana x adalah komponen dua dimensi), kelas-kelas tsb dipisahkan oleh sebuah garis lurus, untuk problem dg dimensi yg lebih tinggi, kelas-kelas dipisahkan oleh sebuah hyperplane.
- ..\..\Negnevitsky2002\Kuliah Expert Systems\Lecture07.pdf

- Karena kesederhanaannya, perceptron sering tidak memadai untuk dijadikan ‘model’ untuk berbagai permasalahan,
- Akan tetapi untuk beberapa problem klasifikasi ia dapat diselesaiakn dengan sederhana.
- Contoh: EXOR

Bagaimana perceptron belajar?

- Dengan melakukan sedikit penyesuaian pada bobot (weight) untuk mengurangi perbedaan antara actual output dan desire output dari perceptron

- Jika pada iterasi *p*, actual output *Y(p)* dan desire output $Y_d(p)$, maka *error* yang didapat:

$$e_p = Y_d(p) - Y(p)$$

Bagaimana perceptron belajar?

- Setiap perceptron mempunyai kontribusi $x_i(p) \times w_i(p)$ terhadap *X(p)* → input value positif, penambahan pada weight cenderung akan menaikkan nilai output *Y(p)*.

- *Perceptron Learning rule:*

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$
$$w_i(p+1) = w_i(p) + \alpha \times x_i(p) \times e(p)$$

$\alpha =$ learning rate

Perceptron Training pd Logika AND

- Input Variables AND OR EXOR

  x1 x2 $x1 \cap x2$

Step 1: Initialisation

Set initial weight $w_1, w_2 \ldots w_n$ and threshold θ to random numbers in the range [-0.5,0.5]

Step 2: Activation

Activate the perceptron by applying input $x_{(1)}p, x_2(p), \ldots x_n(p)$ and desired output $Y_d(p)$.

Calculate the actual output at iteration p=1

$$Y(p) = step\left[\sum_{i=1}^{n} x_i(p)w_i(p) - \theta\right]$$

Where n is the number of perceptron input, and step is a step activation function

Step 3: Weight Training

Update the weights of the perceptron

$$w_i(p+1) = w(p) + \Delta w_i(p)$$

$$\Delta w_i(p) = \alpha \times x_i(p) \times e(p)$$

Epoch	Inputs		Desired output	Initial weights		Actual output	Error	Final weights	
	x_1	x_2	Y_d	w_1	w_2	Y	e	w_1	w_2
1	0	0	0	0.3	−0.1	0	0	0.3	−0.1
	0	1	0	0.3	−0.1	0	0	0.3	−0.1
	1	0	0	0.3	−0.1	1	−1	0.2	−0.1
	1	1	1	0.2	−0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	−1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	−1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	−1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$

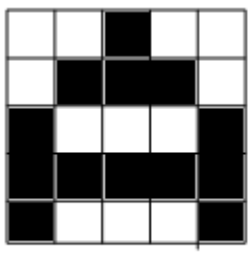
Latihan

- Buat tabel training weight untuk logika OR, dengan learning rate $\alpha=0.1$, threshold $\theta=0.2$, dan menggunakan step function, dimana

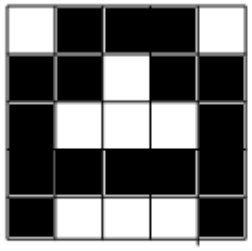
$$Y(p) = step\ [\sum_{i=1}^{n} x_i\ (p) w_i(p) - \theta]$$

Pengenalan karakter

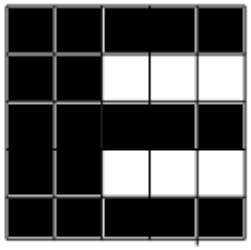
- Diketahui 6 pola masukan berupa huruf A, E dan F berikut



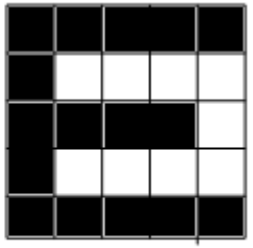
Pola 1



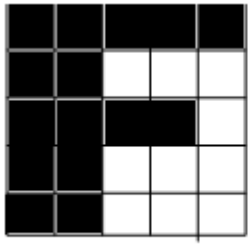
Pola 2



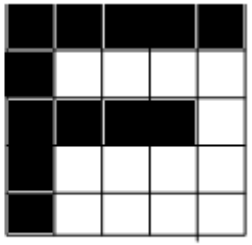
Pola 3



Pola 4



Pola 5

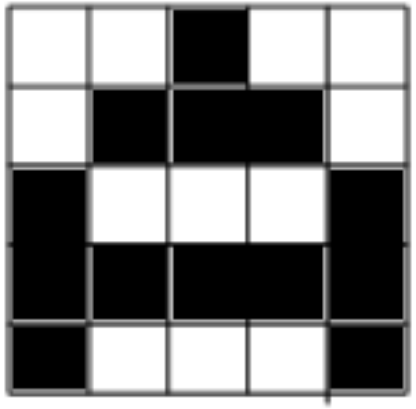


Pola 6

- Buatlah model perceptron untuk mengenali huruf A
- Setiap titik dalam pola merupakan satu variabel input.

X_1	X_2	X_3	X_4	X_5
X_6	X_7	X_8	X_9	X_{10}
X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}
X_{21}	X_{22}	X_{23}	X_{24}	X_{25}

- Bila tanda hitam diberi simbol “1” and putih diberi simbol “-1” maka pola masukan untuk



Pola 1

- Berikut adalah data pola masukan dan nilai targetnya

Pola Masukan	Nilai Target1
1	1
2	1
3	-1
4	-1
5	-1
6	-1

- Bobot Awal: $w_1 \ldots w_{25} = 0$
- Bias b = 0
- Learning rate $\alpha = 1$
- Threshold $\theta = 0.5$
- Fungsi Aktivasi = $\begin{cases} 1, & jika\ net > 0.5 \\ 0, & jika\ -0.5 \leq net \leq 0.5 \\ -1, & jika\ net < -0.5 \end{cases}$
- Lakukan perubahan bobot dan bias selama $y \neq target$