

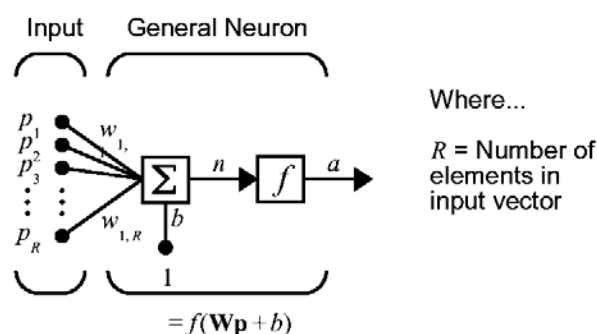
## BAB VII

### BACKPROPAGATION

- *Backpropagation* dibuat untuk menggeneralisasi aturan belajar Widrow-Hoff terhadap jaringan berlapis banyak dan fungsi transfer diferensiabel.
- Pada proses belajar, digunakan iterasi dengan menggunakan pasangan masukan-keluaran hingga terjadi pendekatan terhadap fungsi yang dikehendaki.
- *Backpropagation* standar adalah algoritma *gradient descent*, yaitu aturan belajar Widrow-Hoff.
- Variasi untuk *backpropagation* antara lain *conjugate gradient* dan metode Newton.
- Jaringan *backpropagation* yang sudah menjalani proses belajar dengan baik mampu mengeluarkan jawaban benar atas masukan yang belum pernah dipelajari sebelumnya.

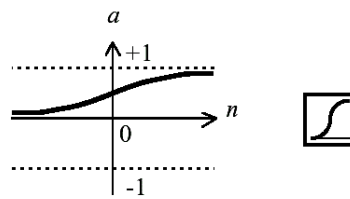
#### Arsitektur

- Pada gambar berikut ini diperlihatkan neuron elementer dengan  $R$  buah masukan.



- Jumlah masukan terbobot dan bias akan membentuk masukan untuk fungsi transfer  $f$ , yang keluarannya menjadi keluaran neuron.

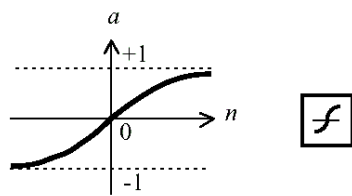
- Fungsi transfer yang sering digunakan di jaringan *backpropagation* adalah fungsi transfer log-sigmoid



$$a = \text{logsig}(n)$$

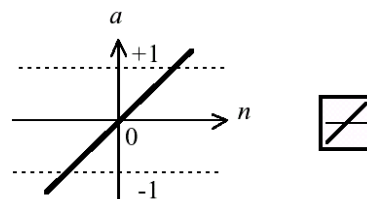
Log-Sigmoid Transfer Function

- Fungsi transfer alternatif untuk jaringan berlapis-banyak adalah fungsi *tan-sigmoid* dan fungsi *linier*



$$a = \text{tansig}(n)$$

Tan-Sigmoid Transfer Function

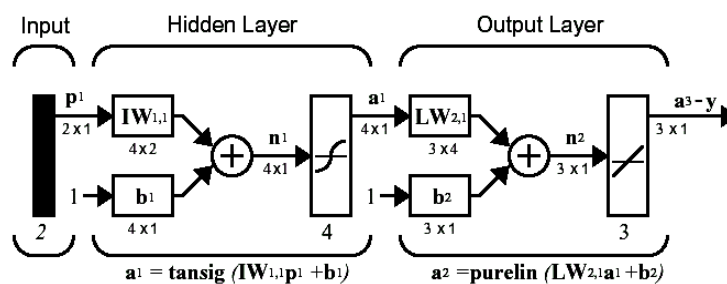


$$a = \text{purelin}(n)$$

Linear Transfer Function

### Jaringan Feedforward

- Jaringan *feedforward* biasanya memiliki satu atau lebih *hidden layer* (lapisan tersembunyi) sigmoid, yang diikuti oleh lapisan output yang terdiri atas neuron linier.



- Jaringan neuron berlapis-banyak dengan fungsi transfer nonlinier memungkinkan jaringan untuk mempelajari hubungan linier maupun nonlinier antara bagian input-output.

## 7.2 Algoritma Backpropagation

### Performance Index

$$F(\mathbf{x}) = E[\mathbf{e}^T \mathbf{e}] = E[(\mathbf{t} - \mathbf{a})^T (\mathbf{t} - \mathbf{a})]$$

### Approximate Performance Index

$$\hat{F}(\mathbf{x}) = \mathbf{e}^T(k) \mathbf{e}(k) = (\mathbf{t}(k) - \mathbf{a}(k))^T (\mathbf{t}(k) - \mathbf{a}(k))$$

### Sensitivity

$$\mathbf{s}^m \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{s^m}^m} \end{bmatrix}$$

### Forward Propagation

$$\mathbf{a}^0 = \mathbf{p},$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1} \mathbf{a}^m + \mathbf{b}^{m+1}) \text{ for } m = 0, 1, \dots, M-1,$$

$$\mathbf{a} = \mathbf{a}^M.$$

### Backward Propagation

$$\mathbf{s}^M = -2\hat{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}),$$

$$\mathbf{s}^m = \hat{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}, \text{ for } m = M-1, \dots, 2, 1,$$

where

$$\hat{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} f^m(n_1^m) & 0 & \dots & 0 \\ 0 & f^m(n_2^m) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & f^m(n_{s^m}^m) \end{bmatrix},$$

$$f^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}.$$

### Weight Update (Approximate Steepest Descent)

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T,$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m.$$