Bab 6
Sistem Intelijensia Adaptif

Bab ini membahas tiga jenis sistem intelijensia adaptif, yaitu: *artificial neural network, genetic algorithm,* dan *fuzzy system*. Kita akan mempelajari karakteristik dan perbedaan dari ketiga sistem.

6.1 Jaringan Syaraf Tiruan (Artificial Neural Network)

Jaringan Syaraf Tiruan (JST) adalah prosesor tersebar paralel yang sangat besar (*massively paralel distributed processor*) yang memiliki kecenderungan untuk menyimpan pengetahuan yang bersifat pengalaman dan membuatnya siap untuk digunakan (Aleksander & Morton 1990).

JST menyerupai otak manusia dalam dua hal, yaitu:
1. Pengetahuan diperoleh jaringan melalui proses belajar.
2. Kekuatan hubungan antar sel syaraf (*neuron*) yang dikenal sebagai bobot-bobot sinaptik digunakan untuk menyimpan pengetahuan.

JST mempunyai sifat dan kemampuan:
1. Nonlinieritas (*Nonlinearity*)
2. Pemetaan Input-Output (*Input-Output Mapping*)
3. Adaptivitas (*Adaptivity*)
4. Respon Yang Jelas (*Evidential Response*)
5. Informasi Yang Sesuai Dengan Keadaan (*Contextual Information*)
6. Toleransi Kesalahan (*Fault Tolerance*)
7. Kemampuan Implementasi Pada VLSI (*VLSI Implementability*)
8. Keseragaman Analisis Dan Perancangan (*Unifomity of Analysis and Design*)
9. Analogi Sel Syaraf Biologi (*Neurobiological Analogy*)

6.1.1 Model Sel Syaraf (Neuron)

Satu sel syaraf dapat dimodelkan secara matematis seperti diilustrasikan oleh gambar 6.1. Satu sel syaraf terdiri dari tiga bagian, yaitu: fungsi penjumlah (*summing function*), fungsi aktivasi (*activation function*), dan keluaran (*output*).

Secara matematis kita bisa menggambarkan sebuah *neuron k* dengan menuliskan pasangan persamaan sebagai berikut :

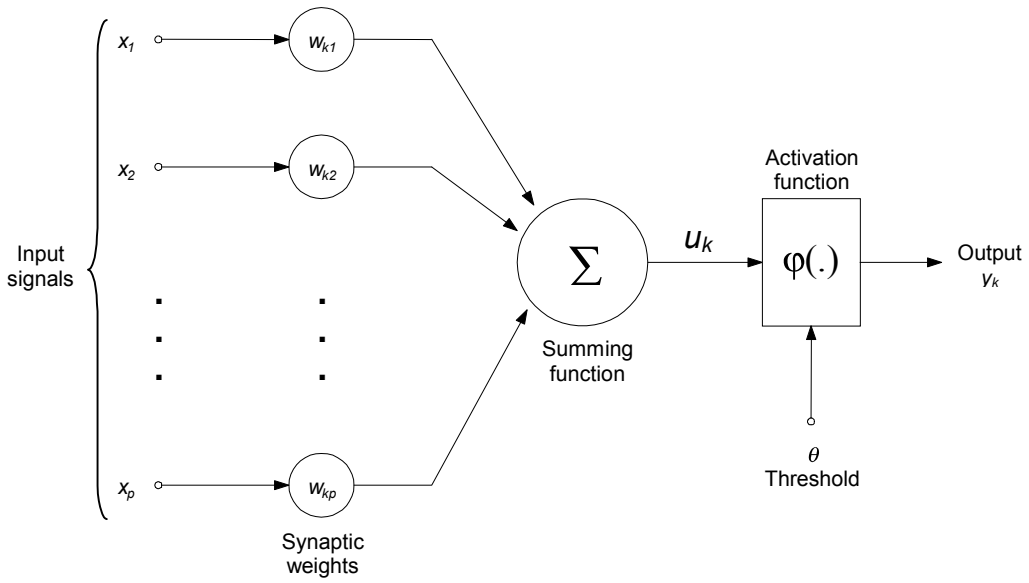
$$u_k = \sum_{j=1}^{p} w_{kj} x_j$$

dan

$$y_k = \varphi(u_k - \theta_k)$$

dimana $x_1, x_2, \ldots, x_p$ adalah sinyal input; $w_{k1}, w_{k2}, \ldots, w_{kp}$ adalah bobot-bobot sinaptik dari *neuron k*; u_k adalah *linear combiner output*; θ_k adalah threshold; $\mu(.)$ adalah fungsi aktivasi; dan yk adalah sinyal output dari *neuron*. Penggunaan threshold memberikan pengaruh adanya *affine transformation* terhadap output u_k dari linear combiner pada model gambar 1 sebagai berikut:

$$v_k = u_k - \theta_k$$



Gambar 6.1 Model Matematis Nonlinier Dari Suatu *Neuron*[HAY94].

6.1.2 Fungsi Aktivasi

Terdapat berbagai macam fungsi aktivasi yang dapat digunakan tergantung karakteristik masalah yang akan diselesaikan. Tiga diantara fungsi aktivasi adalah sebagai berikut:

1. *Threshold Function*

$$\varphi(v) = \begin{cases} 1 & if\ v \geq 0 \\ 0 & if\ v < 0 \end{cases}$$

2. *Piecewise-Linear Function*

$$\varphi(v) = \begin{cases} 1 & v \geq \frac{1}{2} \\ v & \frac{1}{2} > v > -\frac{1}{2} \\ 0 & v \leq -\frac{1}{2} \end{cases}$$

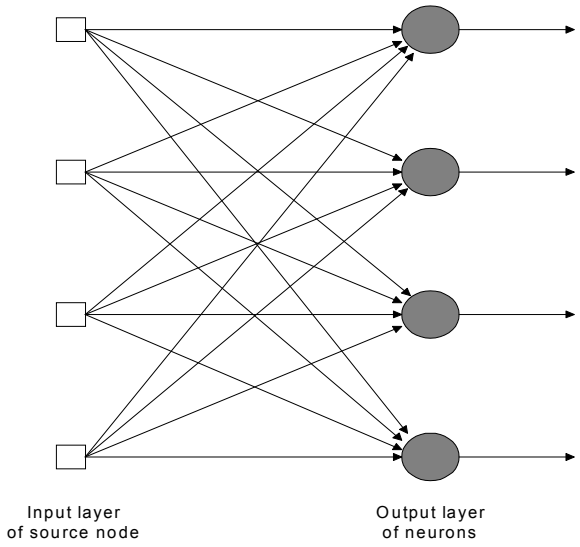
3. Sigmoid Function

$$\varphi(v) = \frac{1}{1+\exp(-av)}$$

6.1.3 Arsitektur Jaringan

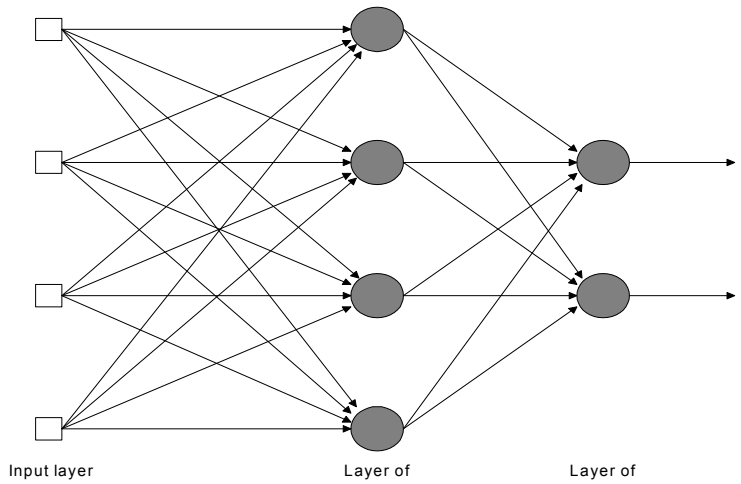
Pola dimana *neuron-neuron* pada JST disusun berhubungan erat dengan algoritma belajar yang digunakan untuk melatih jaringan.

1. Single-Layer Feedforward Networks

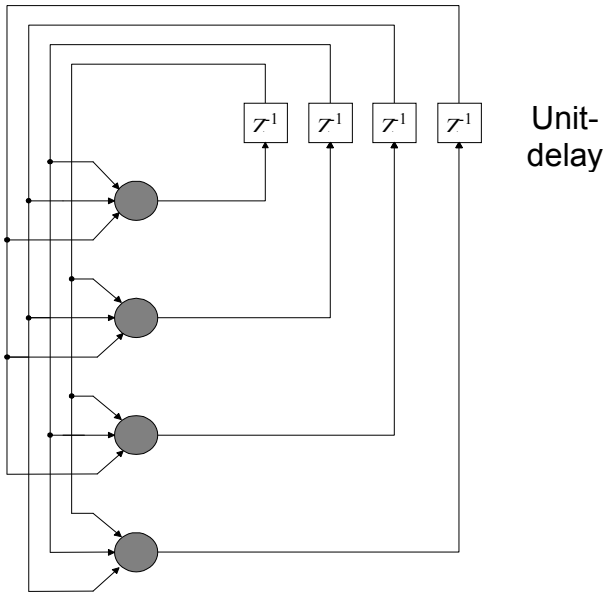


Gambar 6.2 Feedforward Network dengan satu lapisan *neuron*s

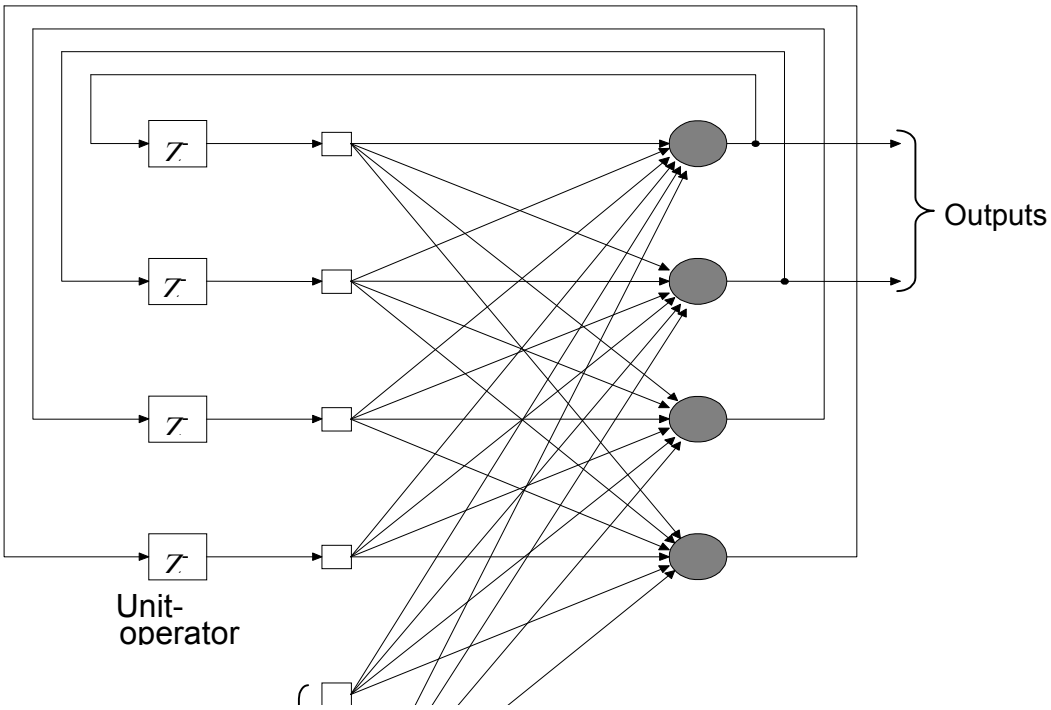
2. Multi-Layer Feedforward Networks



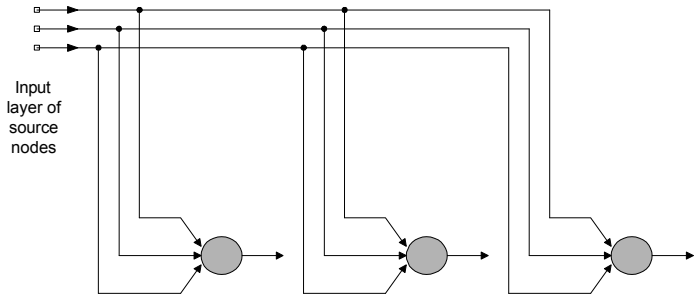
3. Re*current* Networks



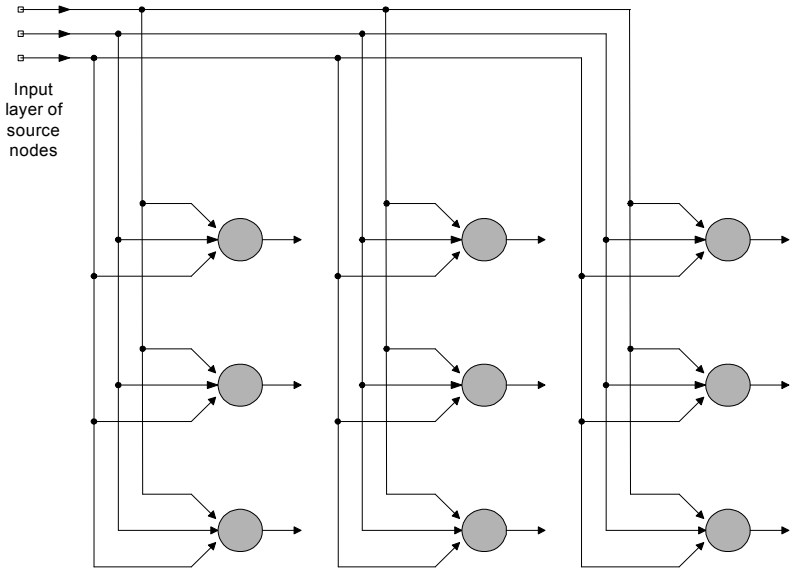
Gambar 6.4 Recurrent network tanpa *self-feedback loop* dan tanpa *hidden neurons*.



4. Lattice Structure



(a)



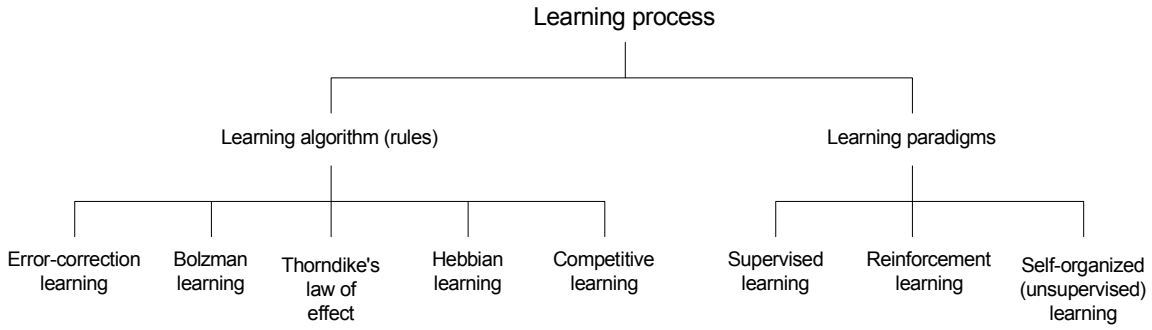
(b)

Gambar 6.6 (a) *Lattice* satu dimensi dengan 3 *neuron*s. (b) *Lattice* dua dimensi dengan 3 kali 3 *neuron*s.

6.1.4 Proses Belajar

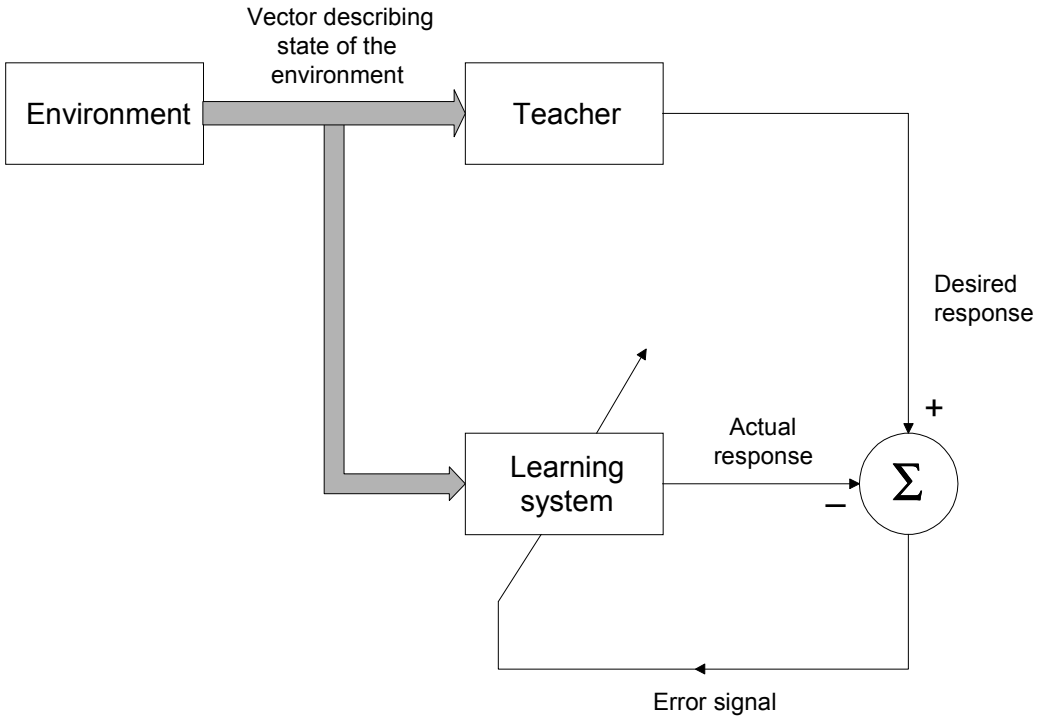
Belajar: suatu proses dimana parameter-parameter bebas JST diadaptasikan melalui suatu proses perangsangan berkelanjutan oleh lingkungan dimana jaringan berada.

Definisi proses belajar ini menyebabkan urutan kejadian sebagai berikut:
1. JST dirangsang oleh lingkungan
2. JST mengubah dirinya sebagai hasil rangsangan ini.
3. JST memberikan respon dengan cara yang baru kepada lingkungan, disebabkan perubahan yang terjadi dalam struktur internalnya sendiri.



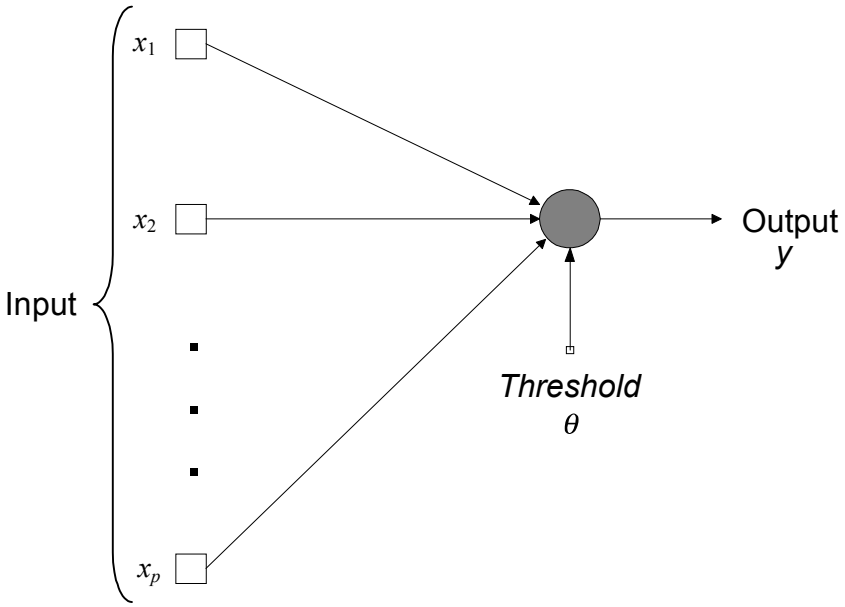
Gambar 6.7 Taksonomi Proses Belajar.

Supervised *Learning* (Belajar Dengan Pengawasan)

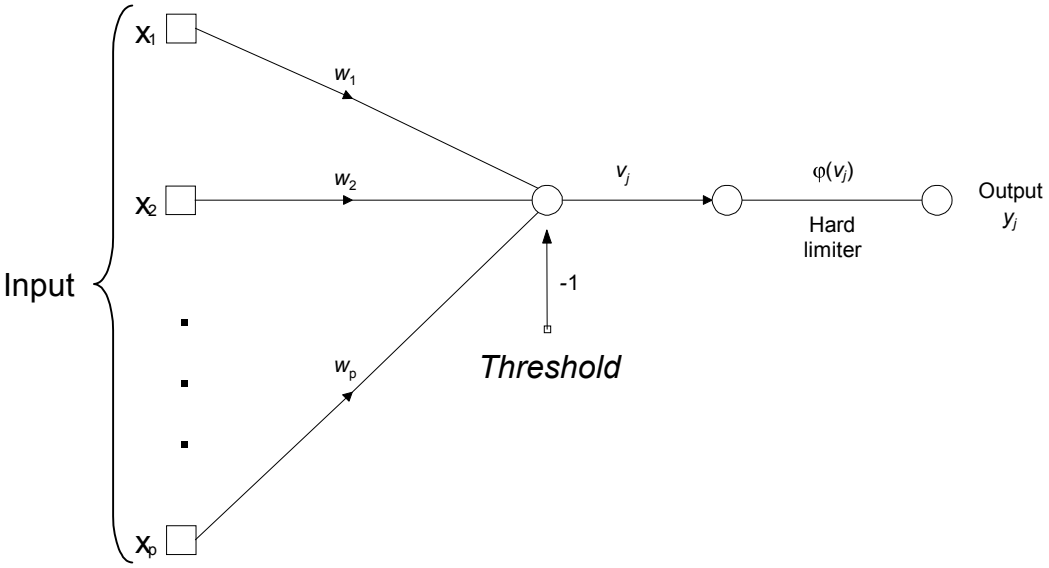


6.1.5 Perceptron

Perceptron adalah bentuk paling sederhana dari JST yang digunakan untuk pengklasifikasian jenis pola khusus yang biasa disebut *linearly separable* (pola-pola yang terletak pada sisi yang berlawanan pada suatu bidang).



Gambar 6.8 Single-layer perceptron.

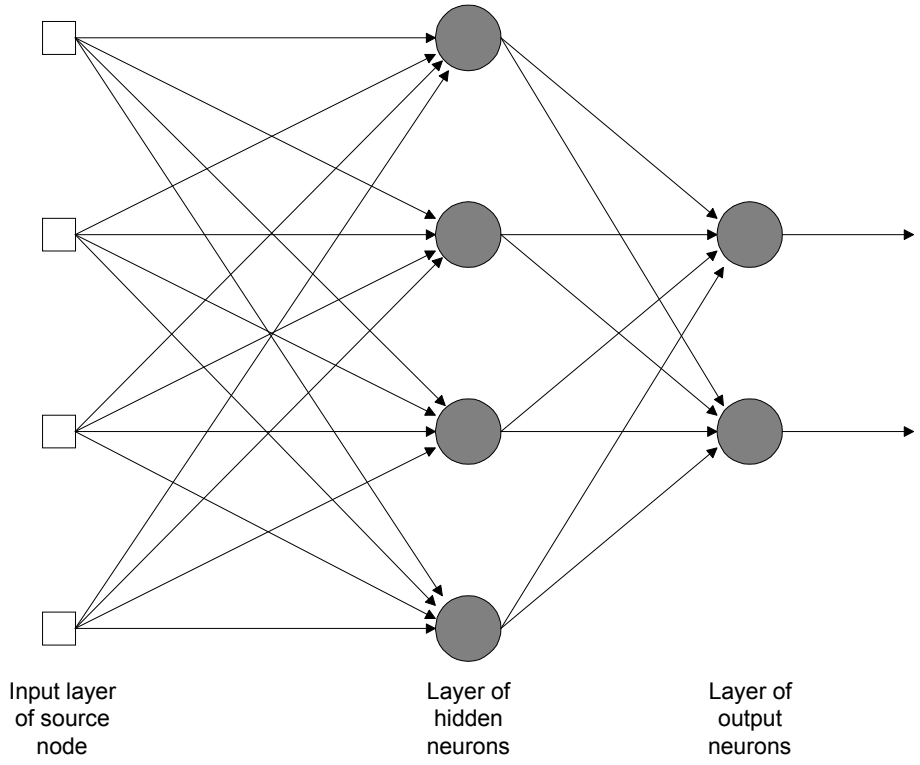


Gambar 6.9 Graph aliran sinyal (*Signal-flow graph*) dari perceptron.

6.1.6 JST Dengan Supervised Learning

6.1.6.1 JST Propagasi Balik

JST Propagasi Balik merupakan model JST yang paling banyak digunakan dalam edukasi. Arsitektur dan proses belajar yang sederhana sangat memudahkan untuk dipelajari. Arsitektur JST Propagasi Balik diilustrasikan oleh gambar berikut:



Gambar 6.10 ST Propagasi Balik dengan empat node pada input layer, satu hidden layer dengan empat node, dan dua node pada output layer.

Algoritma Pelatihan JST Propagasi Balik

1. Definisi masalah, misalkan matriks masukan (*P*) dan matriks target (*T*).
2. Inisialisasi, menentukan bentuk jaringan dan menetapkan nilai-nilai bobot sinaptik (*W1* dan *W2*) dan *learning rate* (*lr*).
3. Pelatihan Jaringan
   - **Perhitungan Maju**

     Keluaran dari *hidden layer* dan *output layer*:

$$A1 = \frac{1}{1 + e^{-\sum_{i=1}^{m} P_i * W1_{ji}}} \quad (1)$$

$$A2 = \frac{1}{1 + e^{-\sum_{j=1}^{n} A1_j * W2_{kj}}} \quad (2)$$

Galat (E) dan *Sum Square Error* (SSE) didefinisikan sebagai berikut:

$$E = T - A2 \quad (3)$$

$$SSE = \sum E^2 \quad (4)$$

- **Perhitungan Balik (Perbaikan Bobot-bobot Sinaptik)**

$$D2 = A2 * (1 - A2) * E \quad (5)$$

$$dW2 = dW2 + (lr * D2 * A1) \quad (6)$$

$$D1 = A1 * (1 - A1) * (W2 * D2) \quad (7)$$

$$dW1 = dW1 + (lr * D1 * P) \quad (8)$$

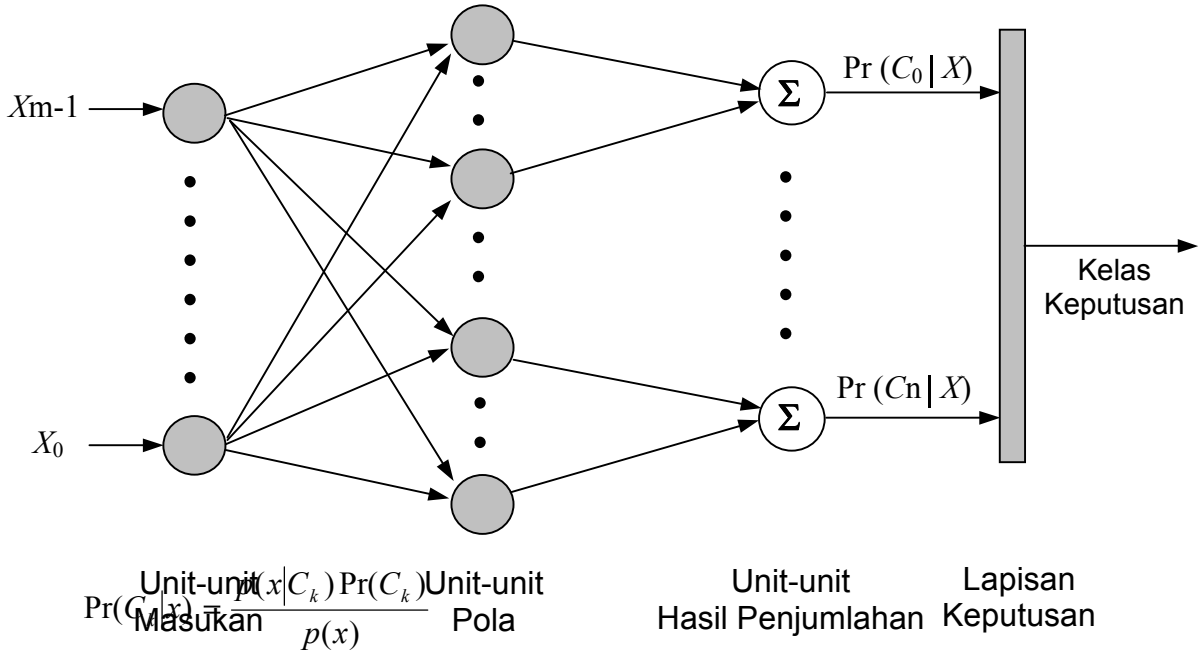
$$W2 = W2 + dW2 \quad (9)$$

$$W1 = W1 + dW1 \quad (10)$$

4. Langkah-langkah diatas adalah untuk satu kali siklus pelatihan (satu *epoch*). Proses pelatihan diulang sampai jumlah *epoch* tertentu atau telah tercapai *SSE* yang diinginkan.
5. Hasil akhir pelatihan jaringan adalah didapatkannya bobot-bobot $W1$ dan $W2$ yang kemudian disimpan untuk pengujian jaringan.

Pada prakteknya, perancangan arsitektur JST Propagasi Balik sangat tergantung pada masalah yang akan diselesaikan. Untuk himpunan masukan berdimensi besar atau jumlah kelas keluaran yang dinginkan besar, maka diperlukan jumlah *node* pada *hidden layer* yang lebih banyak. Atau diperlukan lebih dari satu *hidden layer*, tetapi tentu saja ada batas optimumunya untuk kedua parameter tersebut.

6.1.6.2 JST Probabilistik



$$\Pr(C_k|x) = \frac{p(x|C_k)\Pr(C_k)}{p(x)}$$

$$d(x) = C_k \text{ jika } \quad p(x|C_k)\Pr(C_k) > p(x|C_j)\Pr(C_j) \text{ untuk semua } j \neq k.$$

$$p(x|C_k) \approx \frac{1}{(2\pi)^{m/2}\,\sigma_k^{m}\,|C_k|} \sum_{\rho_i \in C_k} \exp\left[-\|x - w_i\|^2 / \left(2\sigma_k^2\right)\right]$$

(* Tahap Pertama *)
for setiap pola ρ_i
  begin
  $w_i = \rho_i$;
  Bentuk unit pola dengan masukan vektor bobot w_i;
    Hubungkan unit pola pada unit penjumlah untuk masing-masing kelas;
  end;
Tentukan konstanta $|C_k|$ untuk setiap unit penjumlah;

(* Tahap ke dua *)
for setiap pola ρ_i
  begin
  k = kelas ρ_i;
  Cari jarak, d_i, dengan pola terdekat pada kelas k;
  d_{tot}[k] = d_{tot}[k] + d_i;
  end;

Algoritma Pelatihan JST Probablistik.

Pemilihan konstanta *g*, agar jaringan memiliki akurasi pengklasifikasian yang tinggi diperoleh melalui percobaan, karena konstanta *g* dipengaruhi oleh jumlah kelas, dimensi pola latih, dan jumlah anggota himpunan pelatihan.

Perbedaan mendasar antara JST Propagasi Balik dengan JST Probabilistik adalah:

JST Propagasi Balik	**I. JST Probabilistik**
Jumlah *neuron* **tetap**, tetapi bobot-bobot sinaptiknya **berubah**.	Jumlah *neuron* **bertambah** sebanding dengan banyaknya vektor input, tetapi bobot-bobot sinaptiknya **tetap**.
Proses pelatihan memerlukan waktu yang lama saat iterasi pengubahan bobot sampai mencapai *steady state*.	JST Probabilistik memerlukan waktu yang sangat singkat, karena hanya dilakukan untuk satu tahap pelatihan saja.
Memerlukan memori yang kecil karena hanya menyimpan bobot-bobot sinaptik hasil pelatihan	Memerlukan memori yang besar sebanding dengan vektor pola latihnya.