BAB II

LANDASAN TEORI

2.1. Sejarah Jaringan Syarat Tiruan

Pada tahun 1943, McCulloch dan Pitts mempublikasikan *"Boolean Brain"* yang merupakan model *neural network* yang pertama. Hal tersebut sangat berguna bagi John von Neumann yang menggambarkan ide McCulloch dan Pitts selama mendesain dijital komputer (John von Neumann, 1951:pp.1-32). Von Neumann menggunakan *"Switch Delay"* – mengambil dari *neuron* McCulloch dan Pitts – dalam membuat beberapa pondasi untuk reproduksi automata. Sekitar tahun 1949, Hebbs mempublikasikan The Organization of Behavior yang menawarkan pola ilmu yang pertama dengan menguraikan cara yang dapat mengubah kemampuannya. Pola ilmu Hebbs terus mengilhami bermacam-macam model *neural network*.

Publikasi dari test Turing pada 1950 mengusulkan bahwa suatu komputer dapat menjadi pertimbangan intellijen jika seseorang berhubungan melalui teletype, berdasar pada respon seseorang pada pertanyaan yang berbeda-beda. Beberapa tahun kemudian, pada 1954, Ross Ashbly mempublikasikan *The Design of Brain*. *Neural network* menyelidiki penerima dorongan kuat dengan formula dan model *perceptron* Rosenbalt pada 1958. Penghargaan atas kerjanya termasuk demonstrasi mesin pengenal pola, bukti skema ilmu perceptron, dan kemampuan untuk mengklasifikasikan pola linear yang terpisah-pisah.

STIKOMP SURABAYA

Pada tahun 1956, Dartmouth Summer Research telah membuat sebuah proyek mengenai *Artificial Intelligence* (AI) maupun jaringan syaraf. Hasil dari proyek ini juga memberikan sumbangan mengenai AI, maupun penjelasan mengenai bagian paling bawah otak. Pada tahun-tahun sesudah proyek Dartmouth, John von Neumann menyarankan bentuk sederhana tiruan fungsi *neuron* dengan menggunakan relay telegraph atau *vacuum tubes*. Juga Frank Rosenblatt, seorang *neuro-biologist* dari Cornell memulai penyelidikan mengenai *perceptron*. Ia mengulas mengenai cara kerja pada seekor lalat. Hasil dari kerjanya telah diimplementasikan pada *hardware* dan merupakan peralatan jaringan syaraf yang paling tua yang masih digunakan sekarang.

Pada tahun 1959, Bernard Widrow dan Marcian Hoff of Stanford membangun sebuah model yang diberi nama ADALINE dan MADALINE, nama ini diberikan karena adanya penggunaan Multiple Adaptive Linear Element. MADALINE adalah jaringan syaraf pertama yang digunakan pada peristiwa nyata. Alat itu berupa filter yang dapat beradaptasi dengan mengurangi gema jaringan telephone. Alat ini masih digunakan secara komersial.

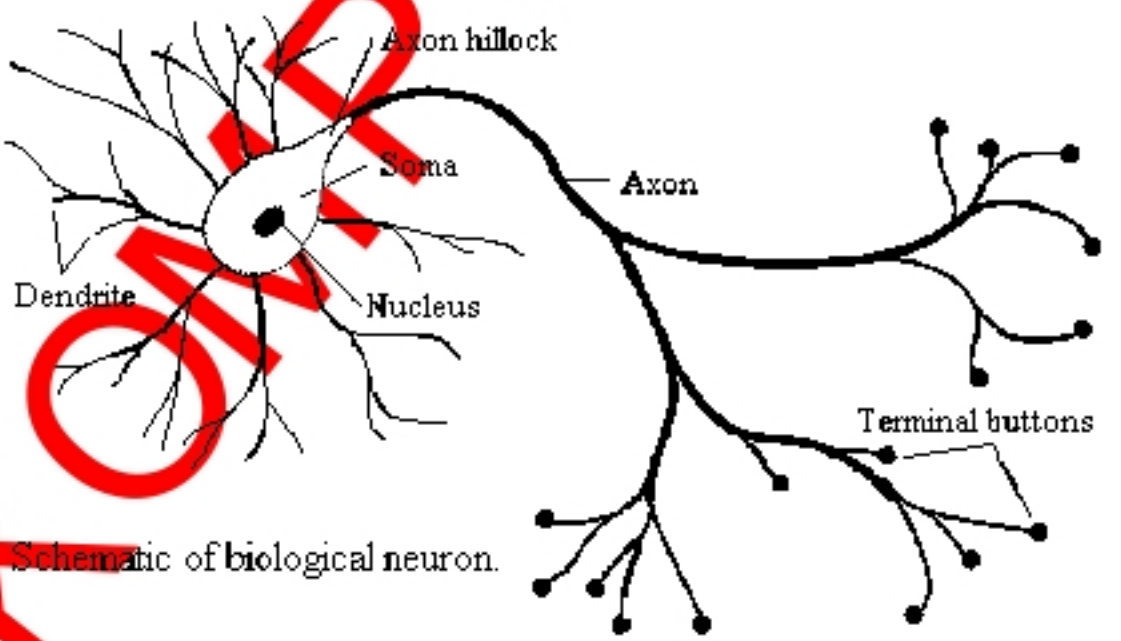
Tahun 1982, John Hopfield mempertunjukkan sebuah tulisan kepada *National Academy of Sciences*. Pendekatan yang dilakukannya rumit, namun dapat diterapkan untuk membuat peralatan yang berguna. Dengan analisis matematika ia menunjukkan bahwa jaringan syaraf mungkin untuk dibuat dan mereka dapat melakukan apapun. Sementara itu pada waktu yang sama di Kyoto, Jepang diadakan sebuah konferensi yang diadakan oleh Amerika dan Jepang, membahas mengenai jaringan syaraf yang konpentitif dan kooperatif. Jepang mengumumkan usahanya untuk mengadakan penelitian komputer generasi ke-5.

STIKOMP SURABAYA

Tahun 1985, *American Institute of Physics*, mulai mengadakan pertemuan tahunan tentang jaringan syaraf untuk komputasi dan menjelang tahun 1987, *Institute of Electrical and Electronic Engineer's (IEEE)* mengadakan pertemuan internasional pertama tentang jaringan syaraf yang dihadiri 1800 orang (Satish Kumar, 2004:9).

2.2. Jaringan Syaraf Biologi

Neuron pada manusia ditunjukkan seperti gambar 2.1 (Neil Fraser (neil@vv.carleton.ca). 21 September 1998. *The Biological Neuron*, (online), (http://vv.carleton.ca/~neil/neural/neuron-a.html, diakses 12 Februari 2006)).



Gambar 2.1 *Neuron* Pada Manusia

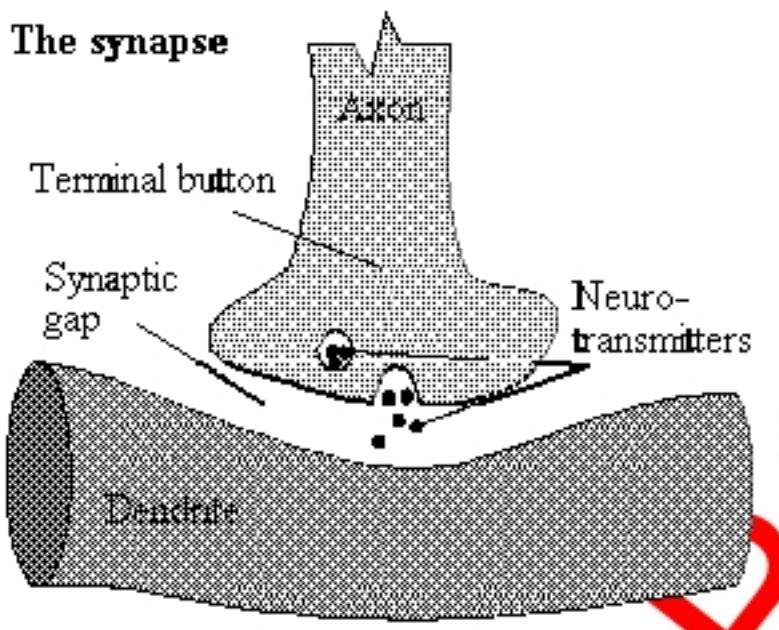
Sebuah *neuron* memiliki *dendrite* sebagai input, tubuh sel (*soma*) dan *axon* sebagai output. *Neuron* menerima input dari *neuron* lain melalui *dendrite*. Apabila

STIKOM SURABAYA

jumlah dari *electrochemical inputs* ini cukup kuat untuk mengaktifkan *neuron* pada bagian *axon hillock* (nilai lebih besar dari *threshold axon hillock*), ia akan melewatkan signal *electrochemical* ke *axon* dan melewatkan ke *neuron* lain yang *dendrite*nya terhubung dengan *terminal button* pada *axon*. Hubungan ini dipertemukan melalui gap kecil sebuah *synapse* (seperti pada gambar 2.2). Bentuk fisik dan karakteristik *neurochemical* dari *synapse* menyatakan kekuatan dan polaritas dari signal input baru.

Dengan demikian, seluruh otak dibuat dari hubungan-hubungan transmisi *electro-chemical neurons*. Dari sejumlah besar unit proses sederhana (yang masing-masing mempunyai jumlah berat dari tiap input, dan membangkitkan signal binary apabila total input melampui dari level tertentu) otak mengatur tugas komplek ini sepanjang waktu. Dengan jumlah jaringan *neuron* yang saling berhubungan berkisar sampai 10 miliar, maka *neuron* adalah sistem yang kompleks. Namun susunan syaraf yang demikian juga membuat otak sangat fleksibel, disamping juga sangat lemah. Salah satu saja berubah, yang lain pasti akan terpengaruh. Alkohol dan LSD mempunyai effek yang sangat dramatis pada pembuatan atau pengrusakan kimiawi dalam tempat ini.

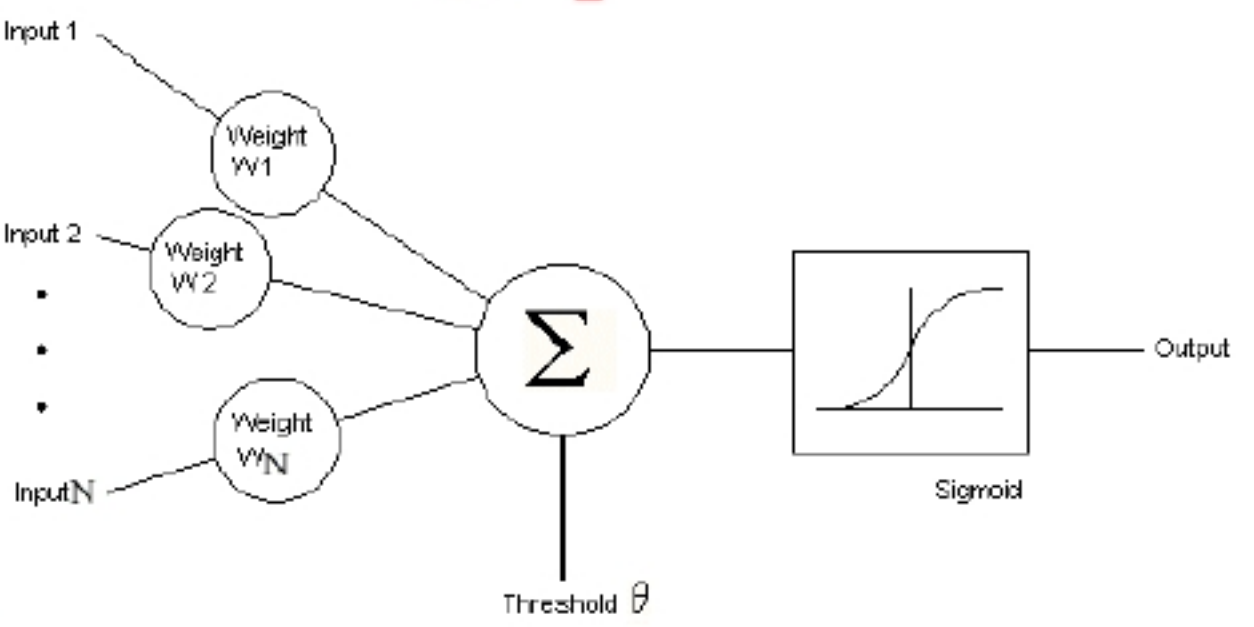
STIKOMP SURABAYA



Gambar 2.2. *Synapse* Pada *Neuron*

2.3. Jaringan Syaraf Tiruan

Gambar 2.3 adalah usaha manusia dalam memodelkan jaringan syaraf tiruan.



Gambar 2.3 *Neuron* Tiruan (*Perceptron*)

Perceptron menerima beberapa input namun hanya mempunyai satu output. Tiap input mempunyai bobot sendiri-sendiri. Perceptron akan menjumlah bobot dengan

SURABAYA

fungsi penjumlahan dan dihubungkan dengan nilai *threshold* serta fungsi aktifasi. Hasil dari proses ini adalah output perceptron. Secara matematis proses tadi dapat ditulis sebagai berikut:

$$u = \sum_{i=1}^{N} w_i x_i \quad \text{.....(2.1)}$$

dan

$$y = f(u - \theta) \quad \text{.....(2.2)}$$

dengan

$x_1, x_2, \cdots, x_N$ = signal input

$w_1, w_2, \cdots, w_N$ = bobot *synaptic*

u = *neuron activation potensial*

θ = *threshold*

y = signal output

f = fungsi aktifasi

untuk menyederhanakan notasi, maka $w_0 = \theta$ dan $x_0 = -1$ sehingga persamaan menjadi:

$$\sum_{i=1}^{N} w_i x_i - \theta = \sum_{i=0}^{N} w_i x_i \quad \text{.....(2.3)}$$

sehingga

$$y = f\left(\sum_{i=0}^{N} w_i x_i\right) \quad \text{.....(2.4)}$$

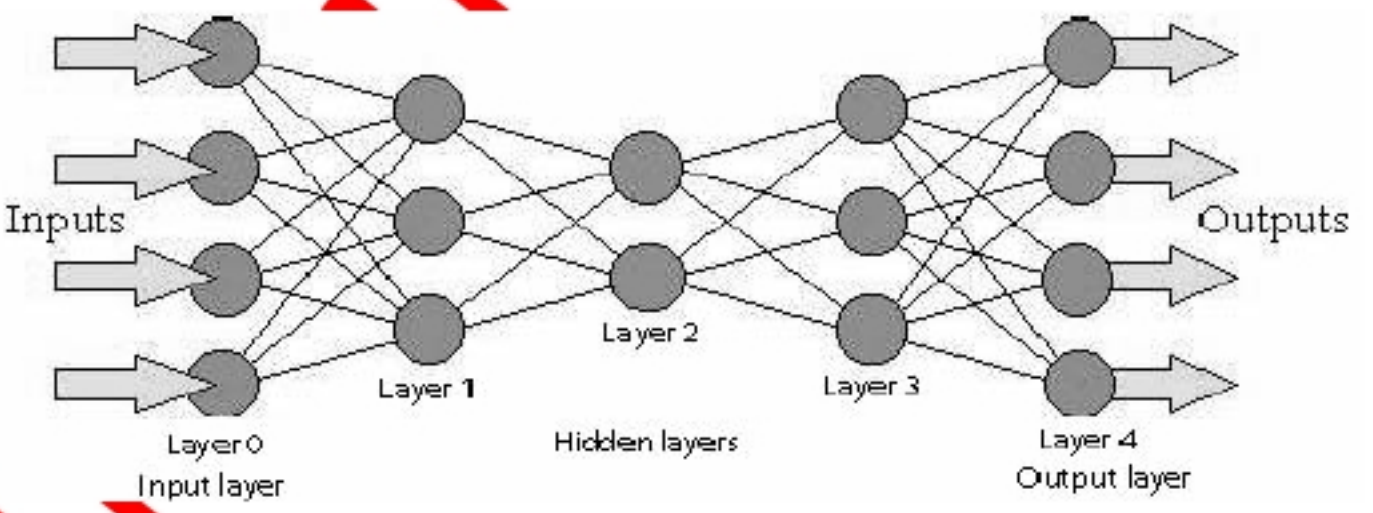
kombinasi antara fixed input $x_0 = -1$ dan bobot tambahan $w_0 = \theta$ dikenal sebagai bias input.

STIKOMP SURABAYA

Fungsi aktifasi yang sering digunakan adalah fungsi *sigmoid*, yang didefinisikan sebagai:

$$f(u) = \frac{1}{1+\exp^{(-au)}} \qquad \text{.....(2.5)}$$

Dimana *a* adalah *slope* parameter dari fungsi sigmoid. Dengan mengubah-ubah nilai *a* , maka kita akan mendapatkan slope yang berbeda pula. Namun pada kenyataannya *neuron* yang beraksi tidak hanya satu namun banyak. Sehingga diperlukan cara menghitung hasil output yang tepat. Salah satu cara yang dipakai adalah dengan feedforward network. Secara grafis menurut (*Neural Networks*, (online), (http://cse.stanford.edu/class/sophomore-college/projects-00/neural-networks/Architecture/feedforward.html, diakses 15 Februari 2006), feedforward network dapat digambarkan seperti gambar 2.4.



Gambar 2.4. *Feedforward Network*

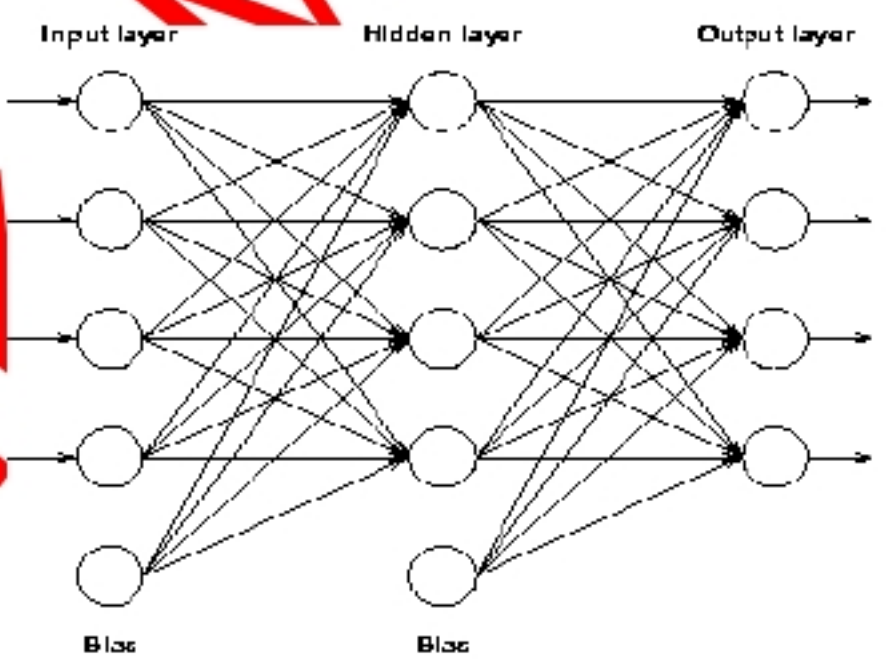
Model jaringan feedforward network mempunyai karakterisitik sebagai berikut:

STIKOM SURABAYA

1. *Perceptron* telah diatur menurut layer-layer tertentu. Layer pertama sebagai input dan layer terakhir sebagai output. Layer yang tengah adalah layer yang tidak terhubung ke dunia luar, sehingga disebut layar *hidden*
2. Tiap *perceptron* pada satu layer pasti berhubungan dengan setiap *perceptron* pada layer berikutnya.
3. Tidak ada hubungan antara *perceptron* pada layer yang sama

2.4 Pembelajaran Dengan Backpropagation

Algoritma Backpropagation pertama kali diperkenalkan oleh Paul Werbos pada tahun 1970-an. Dan diungkapkan kembali pada tahun 1986 oleh Rumelhart and McClelland. Sejak itu algoritma ini telah meluas digunakan. Jaringan pada backpropagation berjalan maju dari input ke output melalui hubungan bobot. Untuk jaringan syaraf dengan satu hidden layer digambarkan seperti gambar 2.5 berikut :



Gambar 2.5. Jaringan Syaraf dengan 1 Hidden Layer

STIKOM SURABAYA

Input pada suatu node ditentukan oleh:

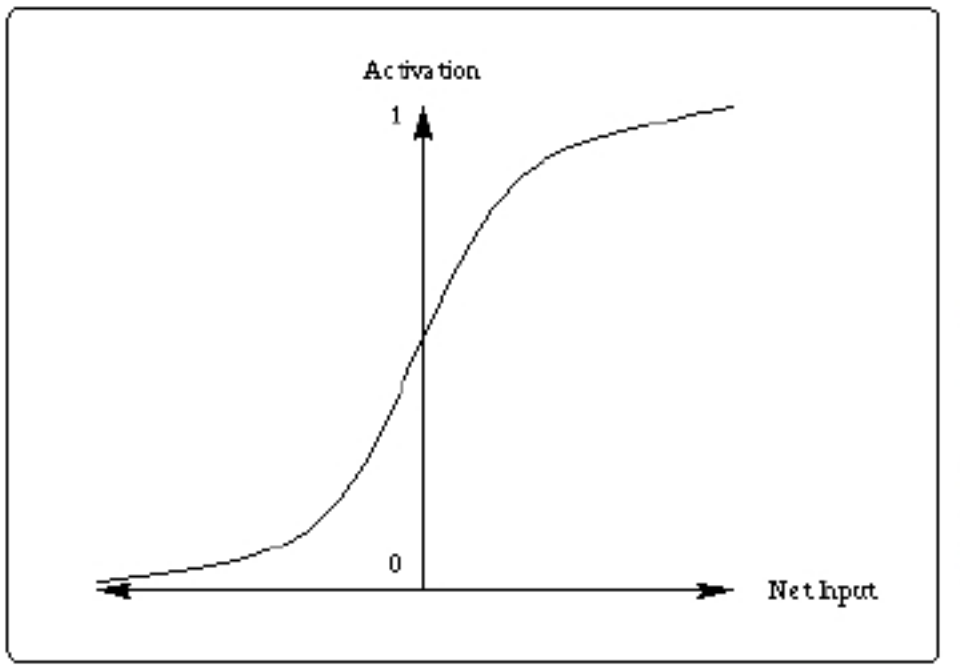
$$net_j = \sum_i w_{ji} a_i \quad \text{.....(2.6)}$$

Dimana, a_i adalah aktifasi input dari unit ke *i* dan w_{ji} berat koneksi dari unit i ke j. Kemudian setelah ditambah dengan unit bias θ, hasilnya dimasukkan ke fungsi sigmoid, sehingga secara notasi matematis menjadi:

$$F(net_j) = \frac{1}{\left(1 + e^{-net_j + \theta_j}\right)} \quad \text{.....(2.7)}$$

Kehadiran bias pada fungsi ini sama maksudnya dengan *threshold* pada *perceptro*n. Dalam backpropoagation, fungsi aktivasi yang dipakai harus memenuhi beberapa syarat yaitu : kontinu, terdiferensial dengan mudah dan merupakan fungsi yang tidak turun. Salah satu fungsi yang memenuhi ketiga syarat tersebut sehingga sering dipakai adalah fungsi sigmoid biner yang memiliki range (0,1). Fungsi *sigmoid* sering disebut fungsi *squashing*. Karena memetakan input kedalam jangkauan tertentu. Fungsi sigmoid bisa dilihat pada gambar 2.6 (Devin McAuley, 1997. *The Backpropagation Network : Learning by Example*, (online), (http://www2.psy.uq.edu.au/~brainwav/Manual/BackProp.html, diakses 2 februari 2006).

SURABAYA
STIKOMP S



Gambar 2.6 Fungsi Aktifasi *Sigmoid*

Setelah mencapai output keluaran, maka sistem mulai melakukan perhitungan *error*. Ada beberapa macam error yang akan dihitung oleh sistem. Error tersebut adalah: *error* sistem, *error node output layer*, dan *error node hidden layer*. Setelah perhitungan error dilakukan, sistem akan mengganti bobot yang ada dengan bobot baru hasil dari perhitungan sistem. Bobot yang diubah akan dipengaruhi oleh *learning rate* dan momentum. Learning rate adalah parameter yang digunakan untuk mengontrol kecepatan perubahan nilai bobot serta menentukan efektifitas dan konvergensi dari pelatihan. Learning rate dilambangkan dengan β dengan nilai antara 0 dan 1. Momentum digunakan untuk konvergensi data dan menghindari agar tidak terjebak pada local minima, dilambangkan dengan m dan bernilai antara 1 dan 0.

Error sistem diketahui dengan persamaan sebagai berikut:

$$Es = \sum_i |t_i - a_i| \quad \text{.....(2.8)}$$

STIKOM SURABAYA

Dengan,

- Es = Error sistem
- t_i = output target dari node ke-i
- a_i = output jaringan dari node ke i

Error sistem digunakan untuk menghitung informasi kesalahan, dan dipergunakan untuk memperbaiki bobot.

Sedangkan *error output layer* didapat dengan persamaan berikut:

$$Eo_i = (t_i - a_i)a_i(1-a_i) \quad \text{.....(2.9)}$$

Dengan,

- Eo_i = error untuk node ke i pada output layer
- t_i = Output target dari node i pada output layer
- a_i = Aktual aktifasi dari node ke i pada output

Error output dipergunakan untuk menghitung kesalahan pada output layer dan digunakan sebagai masukan untuk error node hidden layer.

Error dari *node hidden layer* diketahui dengan persamaan sebagai berikut:

$$Eh_i = a_i(1-a_i)\sum_j Eo_j w_{ij} \quad \text{.....(2.10)}$$

Dengan,

- Eh_i = *error* node ke *i* pada *hidden layer*
- a_i = aktifasi dari node ke *i* pada hidden layer
- Eo_j = error untuk node ke *j* pada output layer

STIKOM SURABAYA

- w_{ij} = bobot antara node ke *i* pada hidden layer dan node ke *j* pada layer output

Setelah diketahui error-error tadi, maka sekarang saatnya memperbaiki semua bobot yang bersesuaian. Sebelum bobot diubah, maka perubahan bobot harus dihitung terlebih dahulu. Perubahan bobot dapat dihitung dengan persamaan berikut:

$$\Delta w_{ij} = \beta E_j a_i + m\Delta w_{ij} \quad \text{.....(2.11)}$$

Dengan,

- Δw_{ij} = Perubahan bobot antara node ke i pada layer sebelumnya dan node ke j pada output layer atau hidden layer
- β = Learning rate
- E_{ij} = error node ke *j* pada output layer atau pada hidden layer
- a_i = aktifasi dari node ke *i* pada layer sebelumnya
- m = momentum
- $\Delta w_{ij}^{'}$ = Δw_{ij} sebelumnya

Persamaan berikut berguna untuk memperbaiki bobot yang ada:

$$w_{ij} = w_{ij} + \Delta w_{ij} \quad \text{.....(2.12)}$$

Dengan,

- w_{ij} = bobot antara node *i* pada layer sebelumnya dan node *j* pada layer output atau pada hidden layer

STIKOMPS SURABAYA

- Δw_{ij} = perubahan bobot antara node ke *i* pada layer sebelumnya dan node ke *j* pada output layer atau pada hidden layer

Secara ringkas cara kerja backpropagation adalah sebagai berikut:

1. Beri nilai awal pada semua bobot yang ada, mulai dari Bobot antara Bias dan Hidden, Bias dan Output, Input dan Hidden, Hidden dan Output. Nilai bobot ini harus sekecil mungkin. Biasanya antara 0 dan 1

   wih (bobot antara input-hidden) = data random

   wbh (bobot antara bias-hidden) = data random

   who (bobot antara hidden-output) = data random

   wbo (bobot antara bias-output) = data random

   dwih (perubahan bobot input-hidden) = 0

   dwho (perubahan bobot hidden-output) = 0

   learning rate = isian pemakai

   momentum = isian pemakai

2. Kerjakan nomor 3 dan 5 sampai sejumlah node hidden, ini adalah menghitung semua input untuk hidden berdasarkan sinyal input yang masuk
3. Hitung inputan untuk hidden layer *(sum_hid)* sampai sejumlah *hidden (j)*
4. Untuk *i* dari 1 sampai sejumlah input, lakukan:

   *sum_hid(j) = sum_hid() + (input(i) * wih(i,j))*
5. *sum_hid(j) = sum_hid(j) + wbh(j)*
6. Hitung output semua hidden

   act_hid(i) = 1 / (1+exp(-sum_hid(i)))

STIKOM SURABAYA

7. Hitung input untuk output (sum_out), dengan melakukan point 8 dan 9 sampai sejumlah output. Ini adalah tahap menghitung input yang masuk ke output
8. *sum_out(j) = sum_out(j) + act_hid(i) * wbo(i,j)*
9. *sum_out(j) = sum_out(j) + wbo(j)*
10. Pada tahap ini ialah menghitung hasil keluaran dari node output *(act_out).* Kerjakan point 11 sampai sejumlah output
11. *act_out(i) = 1/(1+exp(-sum_out(i)))*
12. Setelah semua output dari node output didapatkan. Sekarang sistem akan menghitung error dari sistem dengan memperhatikan output target. Dimulai dengan langkah 13, menghitung *error* per bit
13. *error_bout(i) = output_target(i) – act_out(i)*
14. Hitung *error* sistem, dengan i sampai sejumlah output:

    error_sis = error_sis + | error_bout(i)|
15. Hitung *error* output dengan i sampai sejumlah output:

    *error_bout(i) = error_bout(i) * act_out(i) * (1-act_out(i))*
16. Hitung jumlah *error* output kali bobot antara hidden dan output *(sum_er_ow).* Kerjakan point 17 dengan i sampai sejumlah hidden
17. *sum_er_ow(i) = sum_er_ow(i) + (error_out(j) * who(i,j))*
18. Hitung *error_hid*, yaitu *error hidden* dengan *i* sampai sejumlah hidden:

    *error_hid(i) = act_hid(i) * (1-act_hid(i) * sum_er_ow(i))*
19. Hitung perubahan bobot antara input dan *hidden (dwih),* dengan *i* adalah *counter input* dan *j* adalah counter untuk *hidden:*

    *dwih(i,j) = (learning_rate * error_hid(j) * input(i)) + momentum & dwih(i,j))*

    wih(i,j) = wih(i,j) + dwih(i,j)

STIKOMP SURABAYA

20. Hitung perubahan bobot antara *hidden* dan output, dengan *i* adalah *counter hidden* dan *j counter output:*

    *dwho(i,j) = (learning_rate * error_out(j) * input(i)) + momentum & dwho(i,j))*

    who(i,j) = who(i,j) + dwho(i,j)

21. Hitung perubahan bobot antara bias dengan *hidden:*

    *dwbh(i,j) = (learning_rate * error_hid(j) * input(i)) + momentum & dwbh(i,j))*

    wbh(i,j) = wbh(i,j) + dwbh(i,j)

22. Hitung perubahan bobot antara bias dengan output:

    *dwbo(i,j) = (learning_rate * error_hid(j) * input(i)) + momentum & dwbo(i,j))*

    wbo(i,j) = wih(i,j) + dwbo(i,j)

2.5 Component ActiveX

ActiveX adalah kontrol yang dibuat oleh Microsoft dengan menggabungkan dua teknologi dari Microsoft, yaitu: OLE (*Object Linking and Embedding*) dan COM (*Component Object Model*). Dengan adanya ActiveX, seorang programmer dapat membuat kode program yang dapat dipakai ulang dengan hanya drag and drop pada form yang akan dijadikan proyek. Kode program ini tidak harus dalam bahasa program yang sama. Karena ActiveX dapat dibuat pada berbagai macam bahasa pemrograman yang mendukung ActiveX. Bahasa pemrograman itu antara lain: Visual Basic, Visual C++, Delphi, atau C++

STIKOMPSURABAYA

Builder.Tanpa adanya ActiveX, seorang programmer harus selalui menulis ulang kode program yang sama.

ActiveX mendukung *Iunknown Interface* yang akan membuat client dapat memegang pointer pada interface lain pada objek yang diberikan melalui metode *QueryInterface* dan mengatur keberadaan objek melalui *Iunknown::AddRef* dan *Iunknown::Release*. Seluruh interface COM adalah inherited baik secara langsung maupun tidak langsung dari Iuknown.

Setelah sebuah ActiveX selesai dibuat, ActiveX tersebut harus di daftarkan pada registry windows terlebuh dahulu. Biasanya file ActiveX akan berada pada folder *C:\Windows\System* atau di *C:\Program Files\Common Files\Microsoft Shared*. Misalkan kita mempunyai file ActiveX *linda.ocx* berada di *C:\Windows\System*, maka cara mendaftarkannya adalah sebagai berikut. Click tombol Start pada windows. Pilih *RUN* kemudian ketik:

Regsvr32.exe C:\Windows\System\linda.ocx

Kemudian klik tombol OK. Untuk membuang daftar ActiveX pada Windows, ketik perintah berikut:

Regsvr32.exe /U C:\Windows\System\linda.ocx

STIKOMP SURABAYA