BAB II

LANDASAN TEORI

2.1 Sejarah Singkat Jaringan Saraf Tiruan

JST (Jaringan Saraf Tiruan) mulai dikembangkan sejak tahun 1940. Pada tahun ini belum terdapat buku-buku yang mendefinisikan tentang Jaringan Saraf Tiruan, sehingga JST kurang lebih merupakan suatu sistem komputasi yang didasarkan atas pemodelan sistem saraf biologis (*neuron*) melalui pendekatan sifat-sifat komputasi biologis (*Biological Computation*) [SEK02].

Perkembangan selanjutnya dapat didefinisikan bahwa JST merupakan sistem pemrosesan informasi yang memiliki karakteristik kinerja seperti jaringan saraf biologis (manusia). Hal ini dapat dilihat dari pendapat *Robert Hect - Nielsend* [HEC90] yang mendefinisikan JST adalah suatu struktur pemroses informasi yang terdistribusi dan bekerja secara paralel, yang terdiri dari elemen pemroses dan diinterkoneksi bersama dengan alur sinyal searah, yang disebut koneksi. Setiap elemen pemroses memiliki koneksi keluaran tunggal yang bercabang (*fan out*) ke sejumlah koneksi *Collateral* yang diinginkan (setiap koneksi membawa sinyal yang sama dari keluaran elemen pemroses tersebut). Keluaran dari elemen pemroses tersebut dapat merupakan sembarang jenis persamaan matematis yang diinginkan. Seluruh proses yang berlangsung pada setiap elemen pemroses, harus benar-benar dilakukan secara lokal, pengertiannya

keluaran hanya bergantung pada nilai masukan yang diperoleh melalui koneksi dan nilai yang tersimpan dalam *memory* lokal.

Mulai dari ditemukannya, jaringan syaraf tiruan telah mengalami tahap – tahap perkembangan, antara lain :

- Pada tahun 1940-an, para ilmuwan menemukan bahwa psikologi dari otak sama dengan mode pemrosesan yang dilakukan oleh peralatan komputer.
- Pada tahun 1943, *McCulloh dan Pitts* merancang model formal yang pertama kali sebagai perhitungan dasar *neuron*.
- Pada tahun 1949, *Hebb* menyatakan bahwa informasi dapat disimpan dalam koneksi – koneksi dan mengusulkan adanya skema pembelajaran untuk memperbaiki koneksi – koneksi antar *neuron* tersebut.
- Pada tahun 1954, *Farley dan Clark* mensetup model – model untuk relasi adaptif stimulus – respon dalam jaringan random.
- Pada tahun 1958, *Rosenblatt* mengembangkan konsep dasar tentang *perceptron* untuk klasifikasi pola.
- Pada tahun 1960, *Widrow dan Hoff* mengembangkan *ADALINE* untuk kendali adaptif dan pencocokan pola yang dilatih dengan aturan pembelajaran *Least Mean Square (LMS)*.
- Pada tahun 1974, *Werbos* memperkenalkan algoritma *backpropagation* untuk melatih *perceptron* dengan banyak lapisan.

- Pada tahun 1975, *Little dan Shaw* menggambarkan jaringan syaraf dengan menggunakan model *probabilistik.*
- Pada tahun 1982, *Kohonen* mengembangkan metode pembelajaran jaringan syaraf yang tidak terawasi *(unsupervised learning)* untuk pemetaan.
- Pada tahun 1982, *Grossberg* mengembangkan teori jaringan yang diinspirasi oleh perkembangan psikologi. Bersama *Carpenter*, mereka memperkenalkan sejumlah arsitektur jaringan, antara lain : *Adaptive Resonance Theory (ART), ART 2 dan ART 3.*
- Pada tahun 1982, *Hopfield* mengembangkan jaringan syaraf *reccurent* yang dapat digunakan untuk menyimpan informasi dan optimasi.
- Pada tahun 1985, algoritma pembelajaran dengan menggunakan mesin *Boltzmann* yang menggunakan model jaringan syaraf *probabilistik* mulai dikembangkan.
- Pada tahun 1987, *Kosko* mengembangkan jaringan *Adaptive Biderectional Associative Memory (BAM).*
- Pada tahun 1988, mulai dikembangkan fungsi radial basis.

2.2 Karakteristik Jaringan Saraf Tiruan

JST atau *ANNS (Artificial Neural Network System)*, merupakan suatu pemodelan yang mampu menirukan kemampuan otak manusia untuk

mengklasifikasi pola atau untuk memprediksi atau keputusan-keputusan yang didasarkan pada pengalaman-pengalaman masa lalu. Pada dasarnya otak manusia mengandalkan *input-input* dari lima panca indera sedangkan JST dari sekumpulan data.

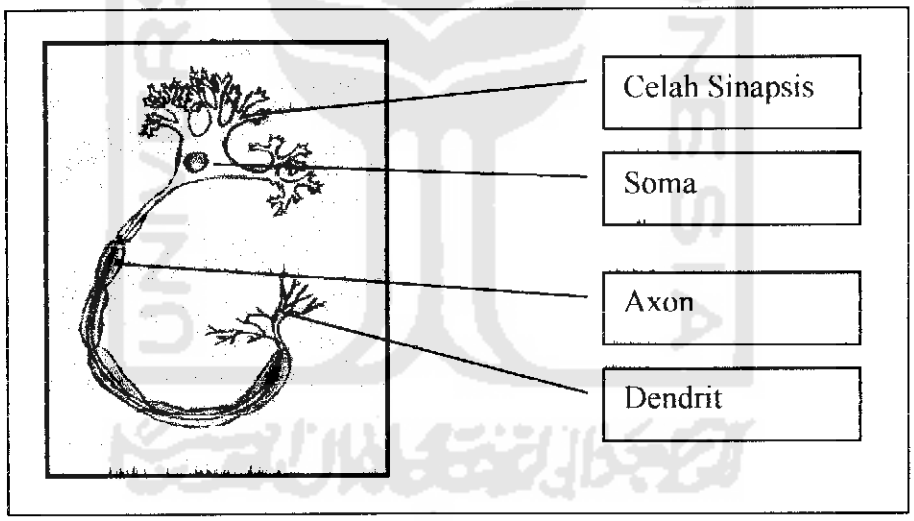
JST yang telah dikembangkan sebagai generalisasi model matematika dari pengertian manusia, didasarkan pada asumsi bahwa :

a. Pemrosesan informasi terjadi pada elemen-elemen sederhana yang disebut saraf (*neuron*).
b. Sinyal-sinyal yang melalui saraf (*neuron*) dihubungkan melalui jalur-jalur hubungan (*connection link*).
c. Setiap jalur hubungan mempunyai sebuah bobot hubungan. Bobot hubungan ini dikalikan dengan sinyal yang ditransmisikan.
d. Setiap saraf (*neuron*) memiliki fungsi aktivasi ke jaringan *input*, untuk menentukan sinyal *output*.

JST memuat sejumlah elemen-elemen pemroses sederhana yang disebut *Neuron* atau *Nodes*. Setiap *neuron* memiliki bobot yang bersesuaian dan akan berhubungan satu sama lain melalui jaringan komunikasi langsung. Bobot pada *neuron* memberikan gambaran informasi yang digunakan oleh jaringan untuk memecahkan masalah. Setiap *neuron* memiliki *internal state* yang disebut aktivasi atau tingkat aktivitas yang berfungsi menerima *input* yang diterimanya. Biasanya *neuron* mengirimkan aktivasinya berupa satu sinyal ke beberapa *neuron* lainnya.

2.3 Tinjauan Umum Jaringan Saraf Tiruan

JST diperoleh dengan menggabungkan beberapa simpul sehingga membentuk arsitektur tertentu. Setiap simpul pada jaringan menerima atau mengirim sinyal dari dan ke simpul-simpul lainnya. Pengiriman sinyal disampaikan melalui *synapses*. Kekuatan hubungan yang terjadi antara setiap simpul yang saling terhubung dikenal sebagai bobot (*weight*). *Neuron* sebagai unit pemroses informasi merupakan suatu cara yang mendasar pada pembangunan Jaringan Saraf Tiruan. Hal ini dapat dilihat secara nyata pada gambar 2.1 dibawah ini :



Gambar 2.1 Jaringan Saraf Biologis

Pada gambar tersebut menunjukkan bahwa cara kerja JST diinspirasikan oleh fisik otak manusia. Komponen otak yang menyediakan kemampuan pengolahan informasi adalah *neuron*. *Neuron* terdiri dari tiga wilayah dasar yaitu *dendrit*, *soma*, dan *axon*.

Dendrit mengkhususkan diri pada beberapa *input* sinyal, sedangkan *Soma* melaksanakan pengolahan sinyal-sinyal tersebut dan *axon* menyediakan jalur *output* untuk sinyal yang telah diproses. Pada proses biologis, *dendrit* akan membentuk pohon *dendrit*, di mana terdapat suatu wilayah menyerupai cabang yang sangat halus dari serat-serat tipis di sekeliling tubuh sel. *Dendrit* ini juga merupakan komponen *input* dari sel. Sedangkan *axon* merupakan serat panjang yang membawa sinyal dari *soma*. Ujung *axon* membelah menjadi suatu struktur menyerupai pohon, dan tiap cabang berakhir di bintil ujung kecil yang hampir menyentuh *dendrit* dari *neuron-neuron* lain. Bintil ujung ini disebut sinapsis(*synapse*). Setiap *neuron* dimungkinkan berhubungan dengan ribuan *neuron* lainnya, melalui jaringan *dendrit* dan *axon* ini.

Soma merupakan komponen *prosesor* dari *neuron*. *Soma* pada dasarnya adalah alat penjumlahan yang dapat melakukan respon pada total *input*nya dalam jangka waktu yang singkat. Keseluruhan sinyalnya dibandingkan dengan ambang *output*, yang merupakan tingkat rangsangan agar *neuron* mengirim *impuls* melalui *axon*nya ke *neuron* lain yang berhubungan.

2.4 Pemodelan *Backpropagation* Jaringan Saraf Tiruan

JST dengan Pemodelan *Backpropagation* pertama kali diperkenalkan oleh *Paul Werbos*, dalam *Thesis Doktorial*nya di *Universitas Harvard* pada tahun 1970-an, yang menyatakan bahwa *Backpropagation* merupakan pemodelan pelatihan untuk jaringan catu maju *multilapis* yang terdiri dari elemen proses

dengan *differensi continue* fungsi aktivasi.

Pada Pemodelan *Backpropagation*, sejumlah *node* dikelompokkan dalam kolom-kolom yang dinamakan lapisan atau *layer*. *Layer* yang menerima masukan dari luar disebut *Input Layer* dan *layer* yang mengeluarkan data *output* disebut dengan *Output Layer*, sedangkan *layer* di antara *Input layer* dan *Output Layer* disebut *Hidden Layer*. Pada Pemodelan *Backpropagation* ini, proses pelatihannya, terdiri atas 3 tahap, yaitu:

a. Perambatan maju masukan pola pelatihan (*feedforward*).

Selama perambatan arah maju, setiap unit masukan(X_i) menerima sinyal masukan(X_i), kemudian diteruskan ke setiap unit lapisan tersembunyi $Z_j,...Z_p$. Perhitungan nilai aktivasi akan dikerjakan di dalam unit tersembunyi dan unit keluaran. Sedangkan nilai aktivasi dari unit tersembunyi(Z_j) dijadikan sebagai keluaran jaringan. Nilai aktivasi diperoleh dari jumlah keseluruhan hasil perkalian semua sinyal masukan dengan bobotnya, yang dilewatkan melalui sebuah fungsi aktivasi.

b. Perambatan mundur perhitungan kesalahan (*error*).

Selama perambatan arah mundur, unit keluaran(Y_k) yang dihasilkan dari unit keluaran(Y_k) dibandingkan dengan pola target. Kemudian dihitung selisih antara keluaran dengan pola target tersebut. Dan hasil yang berupa simpangan atau *error*, disimbolkan dengan $\delta_k(k=1,..m)$. Dari selisih yang dihasilkan di dalam unit keluaran ini diteruskan ke unit tersembunyi, yang pada akhirnya

digunakan untuk mengubah bobot-bobot di antara lapisan keluaran dengan lapisan tersembunyi.

Dengan cara yang serupa, di dalam unit tersembunyi(Z_j) dilakukan perhitungan untuk mendapatkan simpangan(δ_j). Simpangan ini akan diteruskan ke unit masukan untuk mengubah bobot-bobot di antara lapisan tersembunyi dengan lapisan masukan, koreksi bobot-bobot ini ditujukan untuk mengurangi besarnya simpangan.

Persamaan yang digunakan untuk mengubah bobot-bobot tersebut dirancang untuk mengurangi jumlah simpangan (*error*) kuadrat Jaringan Saraf Tiruan.

Propagasi balik *error* inilah yang memberi nama JST sebagai JST *Backpropagation*.

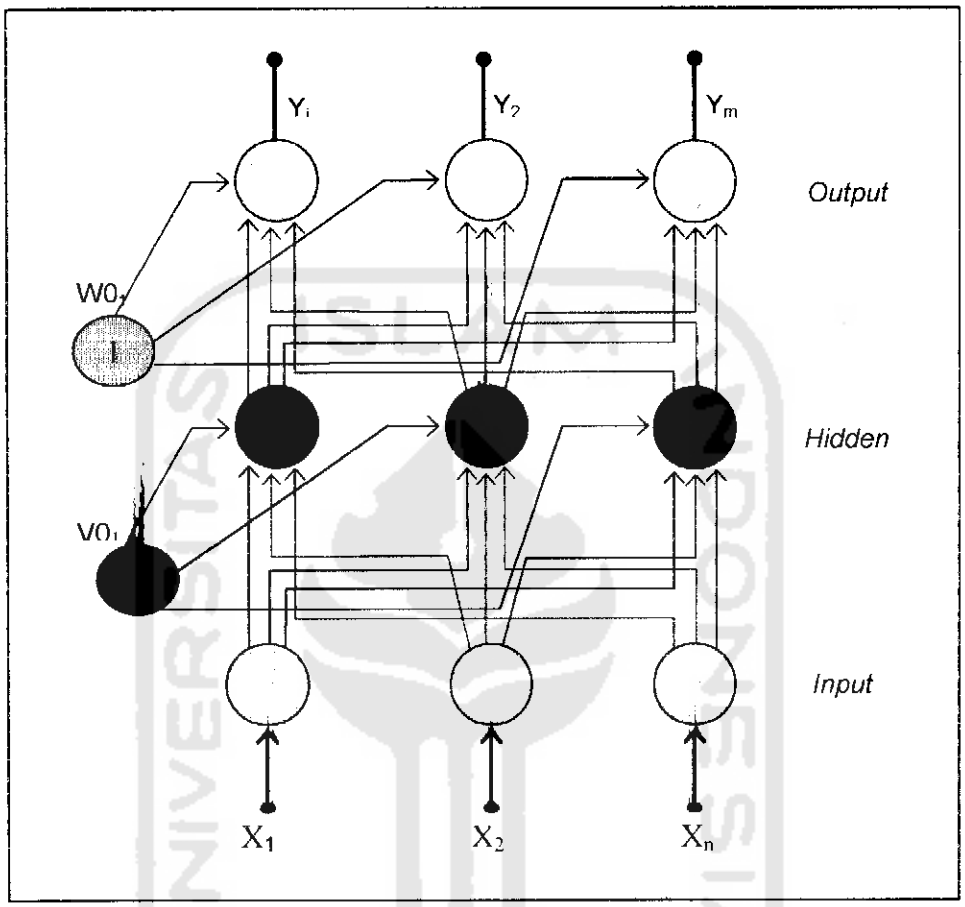
c. Penyesuaian bobot-bobot koneksi.

Selama penyesuaian bobot-bobot koneksi, bobot-bobot W_{jk} lama (yang terletak diantara unit keluaran dengan unit tersembunyi) dijumlahkan dengan hasil perhitungan koreksi bobot pada langkah perambatan arah mundur ($\triangle W_{jk}$) sehingga diperoleh W_{jk} baru.

Begitu pula untuk bobot-bobot V_{ij} lama (yang terletak antara unit tersembunyi dengan unit masukan) dijumlahkan dengan hasil perhitungan koreksi bobotnya($\triangle V_{ij}$) sehingga didapat V_{ij} baru.

Proses pelatihan ini terus dilakukan sampai simpangan dari semua

pasangan pelatihan berkurang. Dan ditunjukkan pada gambar 2.2 di bawah ini:



Gambar 2.2. Jaringan Saraf *Backpropagation* dengan Satu (1) Lapisan Tersembunyi

Pada arsitektur tersebut dapat dijelaskan bahwa setelah semua unit masukan dikalikan dengan bobotnya, hasil dari jumlah total perkalian tersebut ditambahkan dengan bobot biasnya dan diteruskan ke sebuah fungsi aktivasi schingga dihasilkan sebuah keluaran jaringan. Pembatasan yang dilakukan dalam fungsi aktivasi tersebut sangat berkaitan dengan jangkauan tertentu yang ditangani oleh lapisan selanjutnya dalam jaringan.

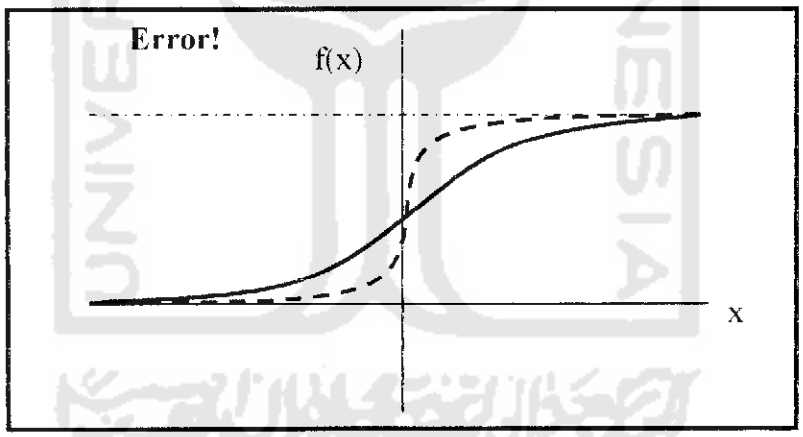
Fungsi aktivasi merupakan suatu fungsi matematis yang berguna untuk

Fungsi aktivasi merupakan suatu fungsi matematis yang berguna untuk membatasi dan menentukan jangkauan *output* suatu *neuron*. Fungsi aktivasi yang akan digunakan pada Jaringan Saraf *Backpropagation*, memiliki beberapa karakteristik penting yaitu kontinyu, dapat didiferensialkan, dan fungsi turunannya mudah dihitung [FAU94]. Fungsi aktivasi ada dua bagian, dan yang digunakan pada bagian ini adalah Fungsi *Biner Sigmoid* yang mempunyai kisaran [0,1] dengan rumusan sebagai berikut :

$$f_1(x) = \frac{1}{1+\exp(-x)} \qquad (2.1)$$

dengan

$$f_1'(x) = f_1(x)[1-f_1(x)] \qquad (2.2)$$



Gambar 2.3. Fungsi Aktivasi Biner Sigmoid

Tata nama dari penjelasan proses diatas dalam algoritma *Backpropagation*, ditunjukkan pada tabel 2.1 :

Tabel 2.1. Tata Nama Yang Digunakan Pemodelan *Backpropagation*

Simbol	Keterangan
x	masukan *vektor* pelatihan $x = (x_1,.....,x_i,.....,x_n)$
t	keluaran *vektor* target $t = (t_1,...,t_k,...,t_m)$
δ_k	nilai bobot koreksi kesalahan penyesuaian untuk w_{jk} yang merupakan kesalahan pada unit *output* Y_k, juga sebagai informasi kesalahan pada unit Y_k yang melakukan perambatan balik ke unit tersembunyi yang merupakan umpan pada unit Y_k
δ_j	nilai bobot koreksi kesalahan penyelesaian untuk v_{ij} yang merupakan informasi kesalahan *Backpropagation* dari lapis keluaran menuju ke unit tersembunyi Z_j.
α	laju belajar
X_i	masukan unit i: Untuk sebuah unit masukan, sinyal masukan dan keluaran adalah sama dengan nama x_i;
V_{oj}	Bias unit tersembunyi j;
Z_j	Unit tersembunyi j Masukan ke Z_j, dinotasikan dengan z_in_j; $z\_in_j = v_{oj} + \sum_i x_i v_{ij}$ Sinyal keluaran (aktivasi) dari Z_j dinotasikan z_j $z_j = f(z\_in_j)$;
W_{ok}	Bias pada unit keluaran k;
Y_k	Keluaran unit k : Masukan ke Y_k dinotasikan dengan y_in_k : $y\_in_k = w_{ok} + \sum_j z_j w_{jk}$ Sinyal keluaran (aktivasi) dari Y_k dinotasikan y_k : $y_k = f(y\_in_k)$

Sumber : Tatanama Pemodelan *Backpropagation* [SAN00].

2.5 Algoritma Pelatihan Backpropagation

- Inisialisasi bobot (ambil bobot awal dengan nilai random yang cukup kecil)
- Selama kondisi berhenti bernilai *FALSE*, kerjakan langkah berikut ini :

1. Untuk tiap - tiap pasangan elemen yang akan dilakukan pembelajaran, kerjakan langkah – langkah :

Feedforward :

Langkah 1 : Setiap unit *input* (X_i, i=1,2,3,...,n) menerima sinyal masukan. x_i dan mengirimkan ke semua unit lapisan tersembunyi

Langkah 2 : Setiap lapisan tersembunyi(Z_j,j=1,2,3...,p),jumlahkan sinyal *input* bobotnya:

$$Z\_in_j = v_{oj} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.3)$$

Gunakan fungsi aktivasi untuk menghitung sinyal keluaran:

$$z_j = f(z\_in_j) \qquad (2.4)$$

dan kirim sinyal tersebut ke semua unit di lapisan atasnya (unit – unit

Langkah 3 : Setiap unit *output* (Y_k,k=1,2,3...m), jumlahkan sinyal *input* bobotnya,

$$y\_in_k = w_{oj} + \sum_{j=1}^{p} z_j w_{jk} \qquad (2.5)$$

dan gunakan fungsi aktivasinya untuk menghitung sinyal keluaran

$$y_k = f(y\_in_k) \qquad (2.6)$$

Kesalahan *Backpropagation* :

Langkah 4 : Setiap unit keluaran (Y_k,k=1,...,m), menerima pola target yang berhubungan dengan pola pelatihan *input*, hitung informasi kesalahan,

$$\delta_k = (t_k - y_k)f'(y\_in_k) \quad (2.7)$$

menghitung koreksi bobot

$$\triangle W_{jk} = \alpha\delta_k Z_j \quad (2.8)$$

menghitung koreksi bias

$$\triangle W_{ok} = \alpha\delta_k \quad (2.9)$$

mengirimkan harga δ_k ke unit-unit lapisan bawah.

Langkah 5 : Setiap unit tersembunyi (Z_j,j=1,...,p),jumlahkan *input* delta

$$\delta\_in_j = \sum_{j=1}^{m} \delta_k w_{jk} \quad (2.10)$$

kalikan dengan turunan fungsi aktivasi untuk menghitung informasi *error*

$$\delta_j = \delta\_in_j f'(z\_in_j) \quad (2.11)$$

menghitung koreksi bobot :

$$\triangle v_{ij} = \alpha\delta_j x_i \quad (2.12)$$

dan menghitung koreksi bias

$$\triangle v_{oj} = \alpha\delta_j \quad (2.13)$$

Perbarui bobot dan bias :

Langkah 6 : Setiap unit *output* (Y_k,k=1,...m) memperbarui bias dan

bobot(j=0,...,p):

$$w_{jk}(\text{baru})=w_{jk}(\text{lama}) + \triangle w_{jk} \quad (2.14)$$

Setiap unit tersembunyi(Z_j,j=1,...,p) memperbarui bias dan bobot (i=0,...,n)

$$v_{ij}(\text{baru})=v_{ij}(\text{lama}) + \triangle v_{ij} \quad (2.15)$$

2. Tes kondisi berhenti[KUS03]

Prosedur Aplikasi *Backpropagation*

Setelah proses pelatihan , jaringan saraf backpropagation diaplikasikan dengan fase *feedforward* dari algoritma pelatihan.

Langkah 0 : Inisialisasi bobot (dari algoritma pelatihan)

Langkah 1 : Untuk setiap vektor masukan kerjakan langkah 2 – 4

Langkah 2 : Untuk i = 1,...,n : set aktivasi unit masukan xi

Langkah 3 : Untuk j = 1,...,p :

$$Z\_in_j = v_{oj} + \sum_{i=1}^{n} x_i v_{ij} \quad (2.16)$$

$$z_j=f(z\_in_j) \quad (2.17)$$

Langkah 4 : Untuk k = 1,...,m :

$$y\_in_k=w_{oj}+ \sum_{j=1}^{p} z_j w_{jk} \quad (2.18)$$

$$y_k=f(y\_in_k) \quad (2.19)$$

2.6 Algoritma Pelatihan Sederhana

Seperti halnya jaringan saraf yang lain, pada jaringan *feedforward* pelatihan dilakukan dalam rangka melakukan pengaturan bobot, sehingga pada akhir pelatihan akan diperoleh bobot–bobot yang baik. Selama proses pelatihan, bobot bobot diatur secara iteratif untuk meminimumkan fungsi kinerja jaringan. Fungsi kinerja yang sering digunakan untuk *backpropagation* adalah *mean square error (ms)*, fungsi ini akan mengambil rata–rata kuadrat *error* yang terjadi antara *output* jaringan dan target.

Sebagian besar algoritma pelatihan untuk jaringan *feedforward* menggunakan *gradien* dari fungsi kinerja untuk menentukan bagaimana mengatur bobot–bobot dalam rangka meminimumkan kinerja. *Gradien* ini ditentukan dengan menggunakan suatu teknik yang disebut dengan nama *backpropagation.* Pada dasarnya, algoritma pelatihan *backpropagation* akan menggerakkan bobot dengan arah *gradien* negatif.

Prinsip dasar dari algoritma *backpropagation* sederhana adalah memperbaiki bobot–bobot jaringan dengan arah yang membuat fungsi kinerja menjadi turun dengan cepat. Ada 2 fungsi pembelajaran untuk bobot–bobot yang menggunakan gradient *descent*, yaitu :

a. Gradient Descent (learngd)

Fungsi ini menggunakan algoritma dasar *gradient descent.* Bobot – bobot akan diperbaiki, dengan perubahan seperti yang ditunjukkan

pada persamaan (4.11), (4.12), (4.17), (4.18).

Parameter yang berhubungan dengan *learngd*, yaitu *learning rate*. Semakin besar nilai learning rate akan berimplikasi pada semakin besarnya langkah pembelajaran. Jika *learning rate* diset terlalu besar, maka algoritma akan menjadi tidak stabil. Sebaliknya, jika *learning rate* diset terlalu kecil, maka algoritma akan *konvergen* dalam jangka waktu yang sangat lama.

b. Gradient Descent dengan *Momentum (learngdm)*

Fungsi ini tidak hanya merespon *gradien* lokal saja, namun juga mempertimbangkan kecenderungan yang baru saja terjadi pada permukaan *error*. Besarnya perubahan bobot ini dipengaruhi oleh suatu konstanta (yang dikenal dengan nama *momentum), mc*, yang bernilai antara 0 sampai 1.[KUS04]

Pemodelah *Backpropagation* dengan *Momentum*

Pada JST dengan Pemodelan *Backpropagation* telah ditentukan prosedur pembaharuan bobot. Dalam hal ini proses pembelajaran yang dijalankan adalah melalui prosedur pembaharuan bobot *Backpropagation* dengan *momentum*.

Momentum akan berpengaruh pada proses pembaharuan bobot pada proses pelatihan. Proses pembaharuan bobotnya akan berjalan secara lambat bila konstanta belajarnya kecil dan dapat terombang-ambing dalam rentang yang luas jika konstanta belajarnya terlalu besar. Namun hal tersebut dapat dihindari dengan

memberi setiap bobotnya sebuah *momentum* α.

Dengan pola *momentum*, bobot (prosedur pembaharuan bobot) dari satu atau lebih pola pembelajaran sebelumnya harus dilakukan penyimpanan. Adapun prosedur pembaharuan bobot untuk *Backpropagation* dengan *momentum*, yaitu:

$$\Delta W_{jk}(t+1) = \alpha\delta_k Z_j + \mu.\Delta W_{jk}(t) \text{ dan } \Delta V_{ij}(t+1) = \alpha\delta_j X_i + \mu.\Delta V_{ij}(t) \qquad (2.20)$$

dengan parameter momentum (μ) dibatasi pada jangkauan 0 sampai 1.

Pemodelan *backpropagation* dengan *momentum* memperbolehkan suatu jaringan membuat kelayakan penyesuaian bobot selama pembaharuan, dalam kasus yang sama dan dalam beberapa pola. Dengan cara menggunakan laju pembelajaran yang kecil untuk mencegah *error* yang terlalu besar. *Momentum* membentuk jumlah bobot *eksponen* (dengan μ sebagai dasarnya dan waktu sebagai *eksponen*nya) dari bobot sebelumnya dan bobot sekarang.

Batas *efektifitas momentum*, termasuk laju pembelajaran, menempatkan batas tertinggi pada sebagian bobot yang dapat diubah dan pada kenyataannya *momentum* dapat menyebabkan bobot diubah dalam arah yang akan menambah *error*.

2.7 Tinjauan Umum Prediksi Banjir

2.7.1 Prediksi

Prediksi pada dasarnya merupakan suatu dugaan atau perkiraan mengenai terjadinya suatu kejadian atau peristiwa di waktu yang akan datang. Prediksi bisa

bersifat *kualitatif* (tidak berbentuk angka) dan bisa bersifat *kuantitatif* (berbentuk angka). Pada prediksi banjir ini lebih berpatokan pada prediksi yang *kuantitatif*, yaitu proses prediksinya diperlukan suatu perhitungan untuk mengambil suatu keputusan.

Kegiatan matematis yang berupa prediksi tersebut sebenarnya telah dilakukan selama bertahun-tahun, sebelum dilakukannya penggunaan komputer yang akan memungkinkan para pemrediksi (*predicator*) membuat perhitungan secara lebih cepat dan mudah. Ada tiga fakta dasar dalam proses prediksi yang telah dilakukan para pemrediksi (*predicator*), yaitu :

1. Semua prediksi adalah proyeksi dari masa lalu.

   Dasar terbaik untuk memperkirakan apa yang akan terjadi di masa depan adalah dengan melihat masa lalu. Semua jenis prediksi mengikuti pendekatan ini.

2. Semua prediksi terdiri dari keputusan semi terstruktur.

   Keputusan prediksi merupakan contoh yang baik dari jenis keputusan semi-terstruktur yang didukung oleh *DSS* (*Decission Support System*). Keputusan tersebut didasarkan oleh beberapa *variabel* yang dapat diukur dengan mudah dan beberapa yang tidak dapat diukur.

3. Tidak ada teknik prediksi yang sempurna. Dalam proses prediksi tidak diharapkan untuk memprediksi masa depan dengan akurasi atau ukuran 100%.

2.7.2 Banjir

Banjir dan bencana akibat banjir dapat terjadi karena faktor alamiah, ataupun faktor dari pengaruh perlakuan masyarakat terhadap alam dan lingkungannya. Faktor alamiah lebih terlihat dari faktor-faktor alam yang mempengaruhinya. Sedangkan faktor pengaruh perlakuan masyarakat terhadap alam dan lingkungannya, terlihat dari segi topologi dan hidrologi. Dari faktor-faktor tersebut, yang sangat berpengaruh atas terjadinya banjir adalah faktor alamiah.

Dalam proses pengendalian banjir, bangunan tanggul yang terdapat pada palung sungai merupakan bangunan yang dibuat oleh masyarakat paling sederhana dan paling banyak digunakan. Berdasarkan dari data penelitian yang ada dapat dilaporkan, bahwa kerusakan bangunan pengendali yang paling banyak dan sering terjadi adalah karena kerusakan tanggul. Terutama di tempat yang tak berbantaran, aliran sungai langsung mengenai tebing tanggul. Hal ini pulalah yang dapat merupakan salah satu penyebab langsung terjadinya kerusakan tanggul. Kerusakan tanggul yang merupakan faktor pengaruh perlakuan sekitarnya, juga dapat merupakan penyebab terjadinya banjir.

Suatu lokasi yang pengelolaan sungainya cukup baik, memiliki prasarana yang dapat dimanfaatkan untuk peramalan banjir. Prasarana tersebut terdiri dari pos pengamatan tinggi muka air sungai, sarana komunikasi seperti radio, telepon yang dihubungkan dengan pos pusat, bahkan ada pula beberapa lokasi yang telah menggunakan peralatan yang paling modern seperti radar pencatat hujan,

telemetri, dan komputer pengolah data.

Data-data dari hasil studi yang pernah dilaksanakan pada beberapa sungai juga dapat digunakan untuk memprediksi banjir. Prediksi banjir ini memang penting sebagai dasar untuk persiapan, guna menanggulangi terjadinya banjir. Prediksi banjir tersebut dapat dilakukan dengan tiga cara teknis, yaitu :

1. Berdasarkan keadaan cuaca

   Pengamatan terhadap cuaca atau ramalan cuaca yang dapat menyimpulkan perkiraan akan terjadinya hujan pada suatu waktu, daerah dan intensitasnya. Sehingga didapatkan data hujan yang terjadi, dan dengan memperhatikan karakteristik pada daerah tersebut, dapat diperkirakan tempat-tempat yang mungkin akan mengalami banjir. Pengalaman dan pengenalan karakteristik daerah akan sangat membantu ketepatan peramalan dengan cara ini.

2. Hubungan tinggi muka air antara pos-pos pengamatan

   Cara ini dilaksanakan berdasarkan hasil pengamatan tinggi muka air pada beberapa pos pengamatan dan saat pengamatan dilaksanakan. Hubungan tinggi muka air antara pos pengamatan satu dengan yang lainnya terletak di sebelah hilirnya serta waktu yang diperlukan untuk perambatan banjir antara pos-pos. Tinggi muka air sangat ditentukan oleh besarnya *debit* air.

3. Menggunakan pemodelan-pemodelan perhitungan secara hidrologi dalam proses prediksi banjirnya.

Pemodelan-pemodelan perhitungan secara hidrologi terhadap prediksi banjir sangatlah bervariasi. Pemodelan perhitungan untuk memprediksi banjir dapat diperhitungkan dari kondisi debit sungai. Proses pengukuran debit sungai ini memperhatikan luas penampang basah, kecepatan aliran dan tinggi muka air sungainya. Tinggi muka air sungai tersebut merupakan tinggi permukaan yang diukur dari titik tertentu yang telah ditetapkan. Titik nol duga air ditentukan pada suatu titik tetap dari ketinggian muka air laut rata-rata atau suatu titik referensi tertentu yang telah dipilih, ini dimaksudkan untuk keseragaman penggunaan data tinggi muka air tersebut.

Pada umumnya kondisi banjir terjadi pada musim penghujan. Dari hasil penelitian dibeberapa tempat menunjukkan bahwa pekerjaan penanggulangan yang dilaksanakan mulai saat teramatinya gejala awal akan terjadinya banjir, telah berhasil mencegah terjadinya banjir. Oleh karena itu, penanggulangan atau pengantisipasian banjir dipersiapkan pada awal musim penghujan.

2.7.3 Debit

Debit sungai adalah perbandingan lurus antara besarnya luas penampang basah, kecepatan aliran dan tinggi muka air sungai tersebut.

2.7.3.1 Pengukuran Debit Sungai

Prinsip pengukuran debit sungai adalah mengukur luas penampang basah, kecepatan aliran dan tinggi muka air sungai tersebut.

Debit dapat dihitung dengan rumus

$$Q \quad (a \times v) \quad \text{.....(2.21)}$$

Keterangan :

Q = debit (m3 / detik)

a = luas bagian penampang basah (m2)

v = kecepatan aliran rata - rata pada luas bagian penampang basah (m / detik)

2.8 Basis Data

2.8.1 Konsep Sistem Basis Data

Sistem Basis Data merupakan sistem yang terdiri atas kumpulan *file* (tabel) yang saling berhubungan (dalam sebuah basis data di sebuah sistem komputer) dan sekumpulan program (DBMS) yang memungkinkan beberapa pemakai dan/atau program lain untuk mengakses dan memanipulasi *file-file* (tabel-tabel) tersebut [FAT99].

2.8.2 Basis Data Relasional

Penggunaan basis data relasional, subuah database akan dengan mudah dikelola walaupun jumlah datanya banyak dan kompleks, seperti pendefinisian data, mana data yang akan dimuat ke dalam sebuah *database*, bagaimana mengelolanya, serta bagaimana membagi data.

Pada prinsipnya sebuah *database* relasional terdiri dari tiga bagian:

1. *Data Definition*

   Mendefinisikan jenis data yang akan dibuat (seperti berapa angka atau huruf), cara relasi data, validasi data dan lainnya.

2. *Data Manipulation*

   Data yang telah dibuat dan didefinisikan tersebut akan dilakukan beberapa pengerjaan, seperti menyaring data, melakukan proses *query* dan sebagainya.

3. *Data Control*

   Bagian ini berkenaan dengan cara mengendalikan data, seperti siapa saja yang bisa melihat isi data, bagaimana data bisa digunakan oleh banyak *user* dan sebagainya.

2.9 Teknik Normalisasi

Normalisasi merupakan cara pendekatan dalam membangun desain logika basis data relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal.

Terdapat 3 bentuk normal yang diidentifikasikan dalam struktur *file* pada suatu relasi. Masing-masing bentuk normal mempunyai batasan tertentu yang dapat dipergunakan untuk menentukan nama bentuk normalnya.

1. Bentuk Normal Pertama (1NF), suatu relasi berada dalam bentuk normal

pertama jika semua atribut mempunyai nilai yang bersifat atomik, tetapi masih terdapat beberapa atribut yang muncul secara berulang (*redudancy*).

2. Bentuk Normal Kedua (2NF), jika suatu relasi berada pada kondisi normal pertama dan mengalami kondisi sebagai berikut :
   a. Jika *primary key* hanya terdiri dari satu atribut.
   b. Atribut lain selain *primary key* tidak dapat berperan seperti *primary key,* melainkan hanya merupakan bagian atau komponen dari *primary key*.
   c. Setiap *record* dalam relasi tersebut, fungsinya tergantung pada fungsi *primary key*.
3. Bentuk Normal Ketiga (3NF), jika relasi berada pada kondisi normal kedua dan tidak terdapat *transitive dependency* yaitu kondisi dimana terdapat pengulangan *record*.

2.10 Teknik Perancangan Sistem Dengan Pemodelan Berorientasi Pada Aliran Data atau Informasi

2.10.1 Diagram Konteks (*Context Diagram*)

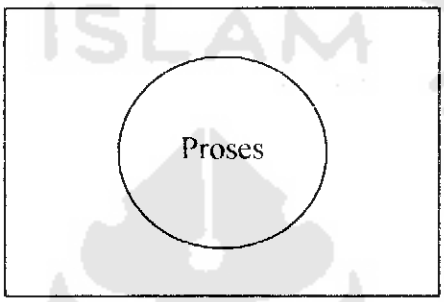
Context Diagram (CD) adalah kasus khusus dari *Data Flow Diagram* (DFD) yaitu bagian dari DFD yang berfungsi memetakan model lingkaran yang direpsentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem [HUS97].

2.10.2 *Data Flow Diagram* (DFD)

Komponen-komponen *Data Flow Diagram* adalah:

1. Proses

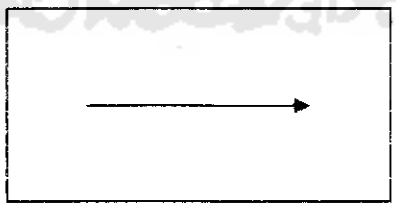
Proses menunjukkan transformasi dari masukkan menjadi keluaran, dalam hal ini sejumlah masukan dapat menjadi hanya satu keluaran ataupun sebaliknya. Proses direpresentasikan dalam bentuk lingkaran, oval, atau bujursangkar.Gambar dari notasi proses yang berbentuk oval dapat dilihat pada gambar 2.4 di bawah ini.



Gambar 2.4. Notasi Proses

2. Alir Data

Komponen ini direpresentasikan dengan menggunakan panah yang menuju ke atau dari proses. Digunakan untuk menggambarkan gerakan paket data dari suatu bagian ke bagian lain dari sistem dimana penyimpanan mewakili lokasi penyimpanan data. Notasi alir data seperti terlihat pada gambar 2.5 di bawah ini

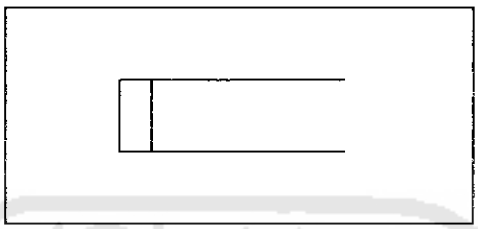


Gambar 2.5. Notasi aliran data

3. Penyimpanan Data

Komponen ini digunakan untuk memodelkan kumpulan data atau paket data. Notasi yang digunakan adalah garis sejajar, segi empat dengan sudut

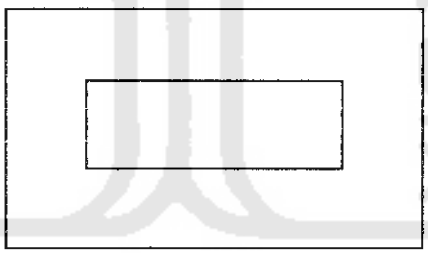
melengkung atau persegi panjang. Gambar notasi simpanan data ini seperti terlihat pada gambar 2.6 di bawah ini.



Gambar 2.6. Notasi simpanan data

4. Terminator

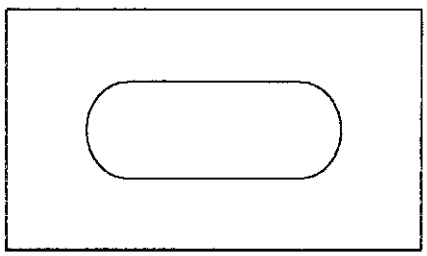
Komponen berikutnya adalah *terminator* yang direpsentasikan menggunakan persegi panjang, yang mewakili *entiti* luar dimana sistem berkomunikasi [HUS97]. Notasi *terminator* atau *entiti* luar sistem dapat dilihat pada gambar 2.7 di bawah ini.



Gambar 2.7. Notasi Terminator

2.11 Teknik Perancangan Proses Dengan Pemodelan *Flow Chart*

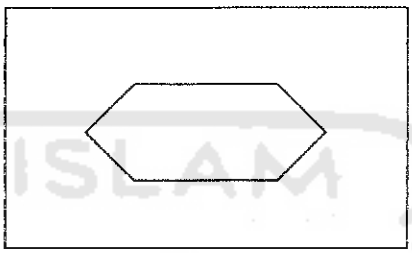
2.11.1 Terminator



Gambar 2.8. Terminator

Gambar 2.8 diatas merupakan simbol terminal *(terminal symbol)*. Digunakan untuk menunjukkan awal dan akhir dari program.

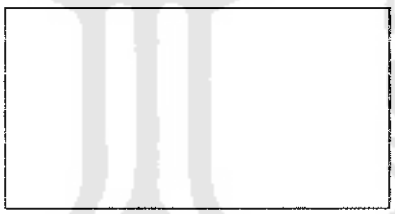
2.11.2 Preparation



Gambar 2.9. *Preparation*

Gambar 2.9 diatas merupakan simbol persiapan *(preparation symbol)*. Digunakan untuk memberikan nilai awal pada suatu *variabel counter*.

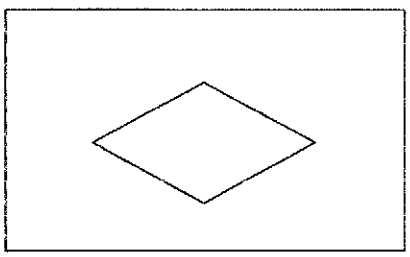
2.11.3 Process



Gambar 2.10. *Process*

Pada gambar 2.10 diatas merupakan simbol pengolahan *(processing symbol)*. Untuk pengolahan aritmatika dan pemindahan data.

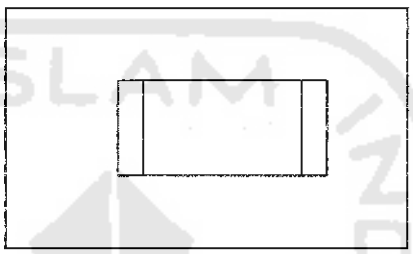
2.11.4 Keputusan



Gambar 2.11. Keputusan

Gambar 2.11 merupakan simbol keputusan *(decision symbol)*. Digunakan untuk mewakili operasi perbandingan logika.

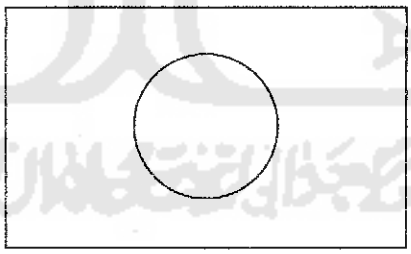
2.11.5 Predefined process



Gambar 2.12. *Predefined process*

Gambar 2.12 diatas merupakan gambar proses terdefinisi (*predefined process symbol*). Digunakan untuk proses yang detailnya dijelaskan terpisah.

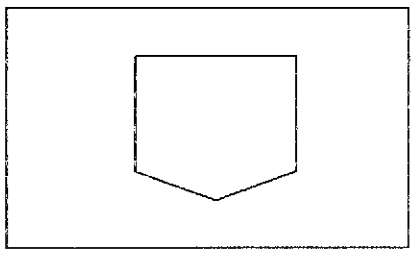
2.11.6 Connector



Gambar 2.13. *Conector*

Gambar 2.13 diatas merupakan simbol penghubung (*connector symbol)*. Digunakan untuk menunjukkan hubungan arus proses yang terputus masih dalam hal yang sama.

2.11.7 Off Page Connector



Gambar 2.14 Off Page *Connector*

Gambar 2.14 merupakan simbol penghubung halaman lain (*off page connector symbol).* Digunakan untuk menunjukkan hubungan arus proses yang terputus masih dalam hal yang sama.