# BAB II

# KAJIAN PUSTAKA

## 2.1 Peramalan

Menurut Nachrowi (2004), peramalan didefinisikan sebagai alat atau teknik untuk memprediksi atau memperkirakan suatu nilai pada masa yang akan datang dengan memperhatikan data atau informasi yang relevan, baik data/informasi masa lalu maupun data/informasi saat ini. Dilihat dari cara memperolehnya, peramalan dapat menggunakan cara kualitatif dan kuantitatif (Nachrowi, 2004). Teknik kualitatif lebih menitikberatkan pada intuisi atau pendapat para pakar. Sedangkan teknik kuantitatif mendasarkan ramalannya pada metode-metode statistik dan matematik.

## 2.2 Curah Hujan

Hujan atau presipitasi adalah air dalam bentuk cair atau padat yang jatuh dari atmosfer sampai ke permukaan bumi (Wirastuti, 2010). Besarnya curah hujan dijabarkan sebagai banyaknya air yang mencapai permukaan tanah dalam waktu tertentu. Jumlah curah hujan berbeda-beda, tergantung wilayah dan faktor penyebabnya. Untuk mengukur jumlah curah hujan, digunakan pengukur hujan atau ombrometer yang terletak di stasiun-stasiun hujan.

Curah hujan dinyatakan dalam inci atau milimeter (1 inci = 25 mm). Curah hujan satu milimeter artinya air hujan yang jatuh pada setiap permukaan seluas 1 $m^2$ setinggi 1 mm, dengan tidak menguap, meresap, atau mengalir. Dengan kata

lain, sejumlah air hujan yang jatuh sebanyak 1 liter pada setiap luasan 1 $m^2$ (BMG, 2008). Jadi jumlah curah hujan yang diukur adalah ketinggian permukaan air hujan yang menutupi suatu daerah luasan di permukaan bumi atau tanah.

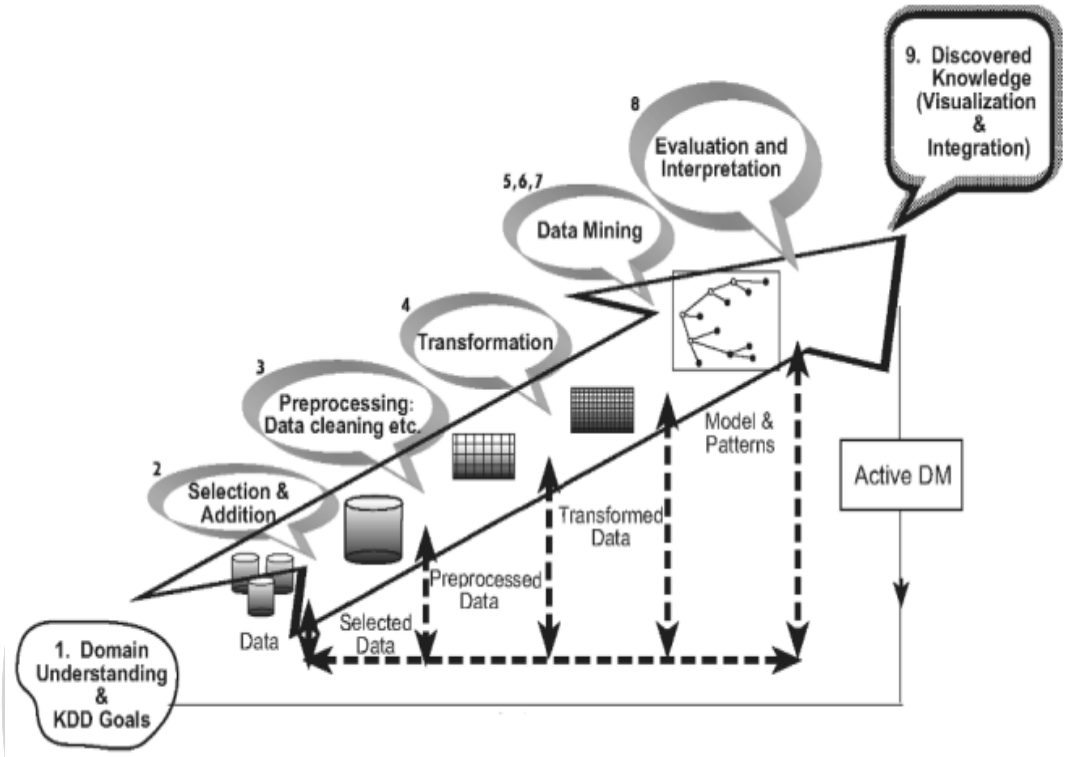
## 2.3 *Knowledge Discovery in Database* (KDD) dan *Data Mining*

*Knowledge Discovery in Database* (KDD) atau penemuan pengetahuan *database* adalah otomatisasi, analisis eksplorasi, dan pemodelan repositori data yang besar (Maimon, 2010). KDD adalah suatu proses yang terorganisir untuk mengidentifikasi pola yang valid, baru, berguna, dan dapat dipahami dari data set yang besar dan kompleks (Maimon, 2010). *Data mining* (DM) adalah inti dari proses KDD, melibatkan algoritma yang mengeksplorasi data, mengembangkan model dan menemukan pola yang sebelumnya tidak diketahui (Maimon, 2010). *Data mining* merupakan istilah untuk menggambarkan proses pemilihan *database* yang besar untuk menemukan pola menarik dan keterkaitannya. Model ini digunakan untuk memahami fenomena dari data, analisis, dan prediksi.

### 2.3.1 Proses KDD

*Knowledge discovery* merupakan suatu proses yang iteratif dan interaktif, yang terdiri dari sembilan langkah. Proses iteratif di setiap langkah berarti tidak menutup kemungkinan dapat kembali ke proses sebelumnya. Proses ini juga interaktif karena memiliki banyak “seni”, dalam artian terdapat taksonomi untuk menentukan pilihan yang tepat sesuai kebutuhan. Proses dimulai dengan

menentukan tujuan KDD dan berakhir dengan implementasi dari *knowledge discovery*. Berikut ini merupakan tahapan proses KDD: (Maimon, 2010)




Gambar 2.1 Proses KDD (Maimon, 2010)

a. Mengembangkan pemahaman dari domain aplikasi. Ini merupakan langkah persiapan awal, dengan mempersiapkan skenario untuk memahami apa yang harus dilakukan dengan banyak keputusan (tentang transformasi, algoritma, representasi, dsb). Hal pertama yang harus dilakukan yaitu memahami dan mendefinisikan tujuan pengguna akhir dan lingkungan dimana proses *knowledge discovery* akan digunakan. Untuk memahami tujuan KDD, dimulai dengan *preprocessing* data yang disajikan pada tiga langkah berikutnya.
b. Memilih dan membuat kumpulan data. Ini termasuk mencari tahu data apa yang tersedia, memperoleh data tambahan yang diperlukan, dan

mengintegrasikan semua data ke dalam satu set, termasuk atribut yang akan dipertimbangkan dalam proses. Proses ini sangat penting karena *data mining* belajar dan menemukan solusi dari data yang ada. Ini adalah dasar untuk membangun model. Jika beberapa atribut penting hilang, maka seluruh studi mungkin akan gagal.

c. *Preprocessing* dan *cleansing*. Pada tahapan ini kehandalan data ditingkatkan. Salah satu caranya yaitu dengan pembersihan data, seperti penanganan nilai data yang hilang dan menghilangkan *noise* atau *outlier* (data asing).

d. Transformasi data. Pada tahapan ini, generasi data yang lebih baik untuk *data mining* disiapkan dan dibangun. Transformasi data disesuaikan dengan spesifikasi proyeknya.

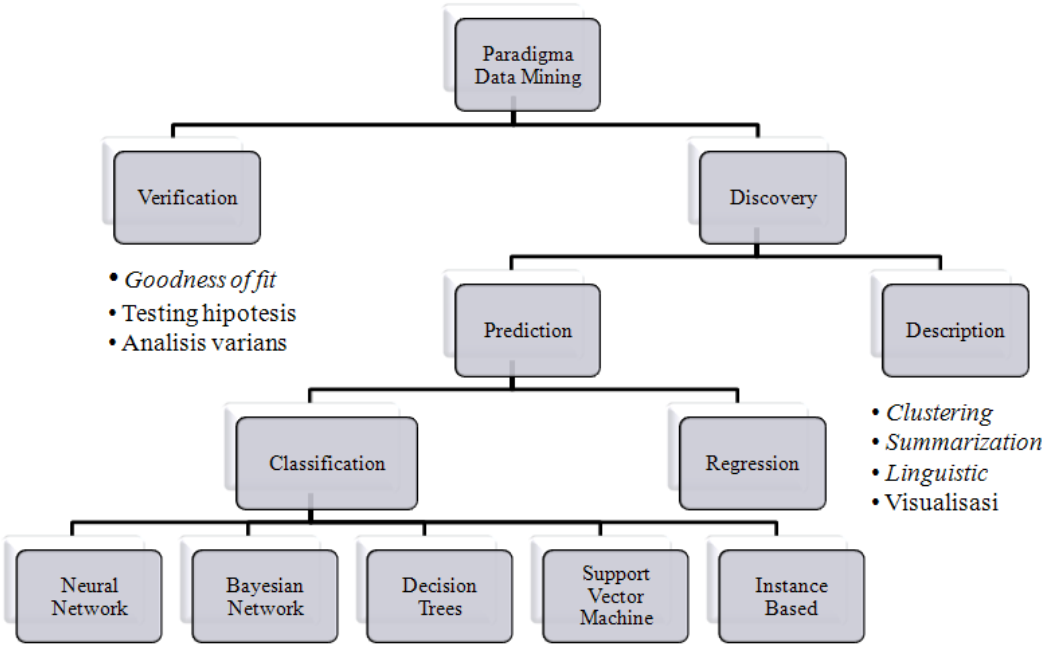
e. Memilih tugas *data mining* yang sesuai. Tahapan ini menentukan tipe *data mining* yang akan digunakan. Contohnya klasifikasi, regresi, atau *clustering*. Ini sangat tergantung pada tujuan KDD dan langkah sebelumnya. Terdapat dua tujuan *data mining*, prediksi dan deskripsi. Prediksi biasanya mengacu pada *data mining supervised* (terawasi), sedangkan deskripsi *data mining* termasuk pada aspek *unsupervised* (tidak terawasi) dan visualisasi *data mining*.

f. Memilih algoritma *data mining*. Tahapan ini memilih metode yang digunakan dalam pencarian pola. Dalam memilihnya disesuaikan dengan tahap sebelumnya. Misalnya untuk kasus prediksi dapat menggunakan algoritma *neural network supervised*.

g. Menggunakan algoritma *data mining*. Akhirnya implementasi algoritma *data mining* tercapai. Di tahap ini algoritma perlu digunakan beberapa kali sampai hasil yang diinginkan tercapai.
h. Evaluasi. Tahapan ini mengevaluasi dan menginterpretasi pola (aturan, kehandalan, dsb) dengan tujuan yang telah ditentukan sebelumnya. Pada tahap ini *discovered knowledge* juga didokumentasikan untuk penggunaan selanjutnya.
i. Menggunakan *discovered knowledge*. Tahapan terakhir, pengetahuan siap diterapkan pada sistem.

### 2.3.2 Taksonomi *Data mining*

Terdapat beberapa metode yang dapat digunakan untuk tujuan yang berbeda. Taksonomi digunakan untuk membantu memahami jenis metode, hubungan dan kelompoknya. Hal ini berguna untuk membedakan dua tipe utama *data mining*: verifikasi (sistem membuktikan hipotesis pengguna) dan *discovery* (sistem menemukan aturan baru dan pola mandiri). Berikut ini merupakan taksonomi *data mining*. (Maimon, 2010)




Gambar 2.2 Taksonomi *Data mining* (Maimon, 2010)

Metode *discovery* secara otomatis mengidentifikasi pola data. Metode *discovery* terbagi ke dalam dua metode, yaitu prediksi dan deskripsi. Metode deskripsi mengacu pada interpretasi data yang fokus pada pemahaman (misalnya dengan visualisasi) berdasarkan hubungan data dengan bagiannya. Metode prediksi mengacu pada pembangunan secara otomatis model kelakuan yang menghasilkan sampel baru dan dapat memprediksi nilai dari satu atau lebih variabel berdasarkan contoh.

Metode verifikasi berkaitan dengan evaluasi hipotesis yang didasarkan pada sumber eksternal (misalnya seorang ahli). Metode-metode umum yang digunakan yaitu statistika, seperti uji kelayakan, tes hipotesis, dan analisis varians (ANOVA). Metode ini kurang terkait dengan *data mining* karena kebanyakan masalah *data mining* terkonsentrasi pada penemuan suatu hipotesis daripada pengujian.

Metode lainnya yang dapat digunakan yaitu *machine-learning*, mengacu pada metode prediksi dengan *supervised learning*, selain itu metode yang dapat digunakan yaitu *unsupervised learning*. Unsupervised learning digunakan untuk pemodelan distribusi seperti *high-dimensional input space*.

Pembelajaran *unsupervised* merupakan teknik pengelompokkan tanpa spesifikasi sebelumnya, tergantung atribut. *Unsupervised* digunakan dalam hal clustering bukan visualisasi. Teknik ini digunakan untuk metode deskripsi. Metode lainnnya yaitu *supervised* yang dapat menemukan pengetahuan berdasarkan atribut input dan target. Terdapat dua model *supervised*, yaitu *classification* dan *regression*. Sebagai contoh, *regression* dapat memprediksi permintaan untuk produk tertentu yang diberikan karakteristiknya. Sedangkan *clasification* dapat digunakan untuk mengklasifikasikan konsumen hipotek yang baik (pengembalian kredit tepat waktu) dan buruk (pengembalian tertunda), atau sebagai sasaran kelas yang diperlukan. Ada banyak alternatif untuk mewakili *classification*. Contohnya *support machine vector*, *decision trees*, *probabilistic summaries*, atau fungsi aljabar.

## 2.4 Proses KDD dan Data Mining pada Kasus Peramalan Curah Hujan

Berdasarkan proses penemuan pengetahuan *database* (KDD) yang telah dijelaskan sebelumnya, maka proses KDD untuk kasus peramalan curah hujan adalah sebagai berikut.

a. Tujuan KDD yaitu meramalkan nilai curah hujan untuk satu bulan ke depan.

b. Data yang tersedia untuk meramalkan curah hujan adalah data historis curah hujan tanpa adanya data faktor penyebab curah hujan.
c. Tahapan awal pada data curah hujan yang tersedia yaitu melakukan *data preprocessing* dan *data cleansing*. Tahapan *data preprocessing* dijelaskan pada poin 2.5.
d. Selanjutnya data curah hujan ditransformasi ke dalam rentang tertentu sesuai dengan algoritma *data mining* yang digunakan. Transformasi data dijelaskan pada poin 2.5.3.
e. Berdasarkan taksonomi *data mining*, peramalan termasuk dalam kategori prediksi model *regression*.
f. Algoritma *data mining* yang dipilih adalah jaringan saraf tiruan *backpropagation* yang dijelaskan secara detail pada poin 2.6.
g. Selanjutnya algoritma diterapkan pada kasus peramalan curah hujan. Proses penerapan algoritma dijelaskan pada Bab IV.
h. Nilai peramalan yang dihasilkan jaringan saraf tiruan *backpropagation* kemudian dievaluasi atau diukur nilai akurasinya. Pengukuran nilai akurasi dijelaskan pada poin 2.7.4. Bila diperlukan, tahapan KDD pada poin c sampai g dievaluasi untuk dipilih metode yang optimum.
i. Tahapan terakhir, proses KDD pada kasus peramalan curah hujan kemudian diterapkan pada sistem. Agar lebih jelas, tahapan ini dijelaskan pada Bab IV.

## 2.5 *Data Preprocessing*

*Data preprocessing* merupakan tahap awal pada proses KDD. Pada tahap ini terdapat proses *data cleansing* (pembersihan data) dan transformasi data yang dijelaskan pada poin 2.5.1 dan 2.5.2 berikut.

### 2.5.1 *Data Cleansing*

*Data cleansing* dapat didefinisikan sebagai suatu proses yang menerapkan metode komputasi dalam analisis *database*, pendeteksian kehilangan dan kesalahan data, dan perbaikan kesalahan (Maletic, 2010). Data cleansing terdiri dari dua fase, yaitu penanganan data yang hilang dan pendeteksian data asing (*outlier*).

a. Penanganan Data yang Hilang

Pada kelompok data yang besar, seperti data curah hujan, tidak menutup kemungkinan terdapat data yang tidak lengkap atau hilang. Maka dari itu, untuk menutupi data yang hilang dapat dilakukan dengan cara menggantinya dengan nilai rata-rata dari kelompok data tersebut.

b. Pendeteksian *Outlier* dengan Metode Statistika

Data *outlier* merupakan data asing atau tidak normal yang terdapat dalam kumpulan data (Maletic, 2010). Contoh sederhananya, dalam persentase yang besar (misal 99,9%) dari elemen data pada umunya, perlu dicurigai adanya sedikit (0,1%) elemen data yang salah. Elemen data salah inilah yang dimaksud *outlier*. Dua hal yang dilakukan di sini adalah mengidentifikasi *outlier* dan

mengidentifikasi normalitas data. Salah satu cara untuk mendeteksi *outlier* yaitu dengan metode statistika.

Metode yang digunakan adalah metode Tukey. Aturan menggunakan metode ini adalah sebagai berikut. (Seo, 2002)

a. Data diurutkan dari yang terkecil ke terbesar.
b. IQR (jangkauan kuartil dalam) adalah jarak antara kuartil bawah (Q1) dan kuartil atas (Q3).
c. Pagar dalam terletak pada jarak (1,5 x IQR) di bawah Q1 dan di atas Q3. Atau dapat pula dituliskan dengan rumus 2.1 berikut.

   [Q1 – 1,5*IQR ; Q3 + 1,5*IQR] .....(2.1)
d. Pagar luar terletak pada jarak (3 x IQR) di bawah Q1 dan di atas Q3. Atau dapat pula dituliskan dengan rumus 2.2 berikut.

   [Q1 – 3*IQR ; Q3 + 3*IQR] .....(2.2)
e. Nilai yang terletak di antara pagar dalam dan pagar luar kemungkinan adalah *outlier*. Juga di luar pagar luar kemungkinan adalah nilai *outlier*. Dengan kata lain, nilai *outlier* kemungkinan terletak di luar pagar dalam.

Keterangan:

Qi = nilai data ke- $\left(\frac{i\,(n+1)}{4}\right)$ .....(2.3)

dimana: i = 1, 3 ; n = banyak data

### 2.5.2 Transformasi Data

Setelah data dibersihkan, langkah selanjutnya pada *preprocessing* yaitu transformasi data. Tujuan transformasi data adalah agar kestabilan taburan data

dicapai. Selain itu berguna untuk menyesuaikan nilai data dengan algoritma yang digunakan. Ada beberapa transformasi yang dapat digunakan, yaitu transformasi polinomial, transformasi normal dan transformasi linear. Nilai hasil transformasi polinomial, normal dan linear dapat diperoleh dengan persamaan sebagai berikut. (Anugerah, 2007)

a. Transformasi Polinomial

$x' = \ln x$ .....(2.4)

dengan

x’ = nilai data setelah transformasi polinomial

x = nilai data aktual

b. Transformasi Normal

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \quad \text{.....(2.5)}$$

dengan,

x’ = nilai data normal

x = nilai data aktual

$x_{min}$ = nilai minimum data aktual keseluruhan

$x_{max}$ = nilai maksimal data aktual keseluruhan

c. Transformasi Linear pada Selang [a,b]

$$x' = \frac{(x - x_{min})\,(b - a)}{x_{max} - x_{min}} + a \quad \text{.....(2.6)}$$

dengan,

x’ = nilai data setelah transformasi linear

x = nilai data aktual

$x_{min}$ = nilai minimum data aktual keseluruhan

$x_{max}$ = nilai minimum data aktual keseluruhan

## 2.6 Jaringan Saraf Tiruan

### 2.6.1 Pengertian Jaringan Saraf Tiruan

Menurut Siang (2009), jaringan saraf tiruan (JST) adalah "sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan saraf biologi". JST dibentuk sebagai generalisasi model matematika dari jaringan saraf biologi, dengan asumsi bahwa:

a. Pemrosesan informasi terjadi pada banyak elemen sederhana (neuron).
b. Sinyal dikirimkan diantara neuron-neuron melalui penghubung-penghubung.
c. Penghubung antar neuron memiliki bobot yang akan memperkuat atau memperlemah sinyal.
d. Untuk menentukan output, setiap neuron menggunakan fungsi aktivasi yang dikenakan pada penjumlahan input yang diterima. Besarnya output ini selanjutnya dibandingkan dengan suatu batas ambang.

JST ditentukan oleh tiga hal: (Siang, 2009)

a. Pola hubungan antar neuron (arsitektur jaringan).
b. Metode untuk menentukan bobot penghubung (metode *training*/*learning*/*algoritma*).
c. Fungsi aktivasi.

### 2.6.2 Model Neuron

Neuron adalah unit pemroses informasi yang menjadi dasar dalam pengoperasian JST dan terdiri dari tiga elemen pembentuk, yaitu: (Siang, 2009)

a. Himpunan unit-unit yang dihubungkan dengan jalur koneksi dengan bobot yang berbeda-beda. Bobot yang bernilai positif akan memperkuat sinyal dan yang bernilai negatif akan memperlemah sinyal yang dibawa. Jumlah, struktur, dan pola hubungan antar unit-unit tersebut akan menentukan arsitektur jaringan.
b. Suatu unit penjumlah yang akan menjumlahkan input-input sinyal yang sudah dikalikan dengan bobotnya.
c. Fungsi aktivasi yang akan menentukan apakah sinyal dari input neuron akan diteruskan ke neuron lain atau tidak.

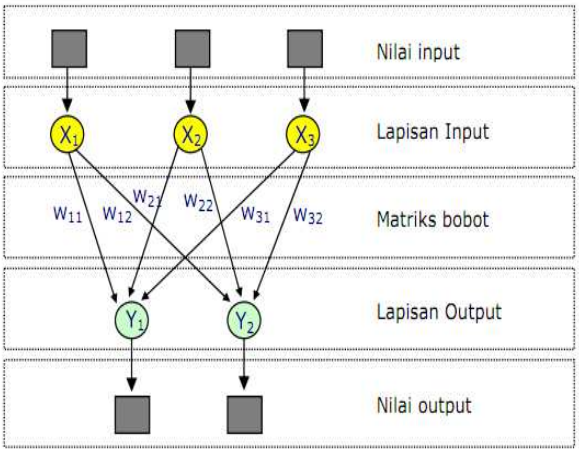
### 2.6.3 Arsitektur Jaringan

Untuk menentukan pola hubungan antar neuron, digunakan arsitektur jaringan. JST memiliki beberapa arsitektur jaringan, yaitu: (Fausset, 1994)

a. Jaringan layar tunggal (*single-layer network*)

   Jaringan layar tunggal memiliki satu lapisan bobot penghubung yang terdiri dari satu unit input dan satu unit output. Unit input terhubung penuh ke unit output tetapi tidak terhubung ke unit input lainnya. Begitupun dengan unit output, tidak terhubung dengan unit output lainnya. Jaringan ini hanya menerima input kemudian secara langsung akan mengolahnya menjadi output

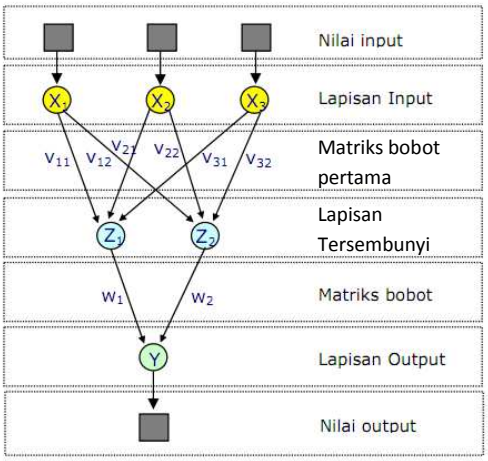
tanpa harus melalui lapisan tersembunyi. Contoh algoritma JST yang menggunakan metode ini yaitu: ADALINE, Hopfield, dan Perceptron.




Gambar 2.3 Arsitektur Layar Tunggal (Elista, 2008)

b. Jaringan layar jamak (*multi-layer network*)

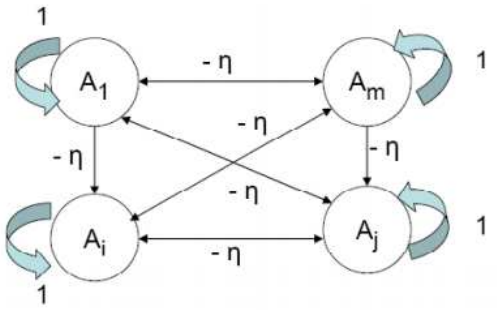
Jaringan layar jamak memiliki satu atau lebih lapisan unit (node) di antara unit input dan unit output. Cirinya, ada lapisan bobot di antara dua level unit yang berdekatan (input, hidden, atau output). Jaringan ini bisa memecahkan masalah yang lebih rumit dibandingkan dengan jaringan lapisan tunggal, tetapi pembelajarannya pun bisa lebih sulit. Bagaimanapun, di beberapa kasus, pembelajarannya lebih berhasil, karena memungkinkan memecahkan masalah yang jaringan lapisan tunggal tidak bisa lakukan. Contoh algoritma JST yang menggunakan lapisan ini, yaitu: MADALINE, *backpropagation*, neocognitron.




Gambar 2.4 Arsitektur Layar Jamak (Elista, 2008)

c. Jaringan dengan layar kompetitif (*competitive layer network*)

Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi aktif. Contoh algoritma yang menggunakan jaringan ini yaitu LVQ.




Gambar 2.5 Arsitektur Layar Kompetitif (Elista, 2008)

### 2.6.4 Fungsi Aktivasi

Ada beberapa fungsi aktivasi yang sering digunakan dalam JST, antara lain:

a. Fungsi Undak Biner (*Threshold*)

Fungsi undak biner dengan menggunakan nilai ambang sering juga disebut dengan fungsi nilai ambang (*treshold*) atau fungsi Haeviside. Fungsi ini dirumuskan sebagai:

$$y = \begin{cases} 0, & \text{jika } x < \theta \\ 1, & \text{jika } x \geq \theta \end{cases} \quad \text{.....(2.7)}$$

(Elista, 2008)

Untuk kasus bilangan bipolar, maka nilai 0 diganti menjadi -1 sehingga persamaannya menjadi:

$$y = \begin{cases} 1, & \text{jika } x \geq \theta \\ -1, & \text{jika } x < \theta \end{cases} \quad \text{.....(2.8)}$$

(Elista, 2008)

b. Fungsi Linear (identitas)

Fungsi linear memiliki nilai output yang sama dengan nilai inputnya.

$F(x) = x$ .....(2.9)

(Siang, 2009)

Digunakan jika keluaran yang dihasilkan oleh JST merupakan sembarang bilangan riil (bukan hanya pada range [0,1] atau [1,-1]).

c. Fungsi Sigmoid Biner

Fungsi ini digunakan untuk JST yang dilatih dengan menggunakan metode *backpropagation*. Fungsi sigmoid biner memiliki nilai pada range 0 sampai 1. Oleh karena itu, fungsi ini sering digunakan untuk jaringan saraf yang membutuhkan nilai output yang terletak pada interval 0 sampai 1. Namun, fungsi ini bisa juga digunakan oleh jaringan saraf yang nilai outputnya 0 atau 1. Fungsi sigmoid biner dirumuskan sebagai:

$$y = f(x) = \frac{1}{1+e^{-x}} \text{ dengan } f'(x) = f(x)\,[1-f(x)] \quad \text{.....(2.10)}$$

(Siang, 2009)

d. Fungsi Sigmoid Bipolar

Fungsi sigmoid bipolar hampir sama dengan fungsi sigmoid biner, hanya saja output dari fungsi ini memiliki range antara -1 sampai 1. Fungsi ini dirumuskan sebagai berikut.

$$y = f(x) = \frac{1-e^{-x}}{1+e^{-x}} \text{ ; dengan } f'(x) = \frac{1}{2}\,[1+f(x)]\,[1-f(x)] \quad \text{.....(2.11)}$$

(Siang, 2009)

### 2.6.5 Bias dan *Threshold*

Terkadang dalam jaringan ditambahkan sebuah unit masukan yang nilainya selalu sama dengan 1 (Siang, 2009). Unit yang sedemikian itu disebut Bias. Bias berfungsi untuk mengubah nilai *threshold* menjadi sama dengan 0 (bukan θ).

### 2.6.6 Metode Pelatihan/Pembelajaran

Berdasarkan cara memodifikasi bobot, ada beberapa macam pelatihan JST, yaitu: (Siregar, 2009)

a. *Supervised Learning* (pembelajaran terawasi)

Pada metode ini, setiap pola yang diberikan ke dalam JST telah diketahui outputnya. Selisih antara pola output aktual (output yang dihasilkan) dengan pola output yang dikehendaki (output target) yang disebut error digunakan

untuk mengoreksi bobot JST sehingga mampu menghasilkan output sedekat mungkin dengan pola target yang telah diketahui oleh JST. Contoh algoritma yang menggunakan metode ini adalah: Hebbian, Perceptron, ADALINE, Boltzman, Hopfield, *Backpropagation*.

b. *Unsupervised Learning* (pembelajaran tak terawasi)

Pada metode ini, tidak memerlukan target output. Selama proses pembelajaran, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai output yang diberikan. Tujuan pembelajaran ini untuk mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya cocok untuk klasifikasi pola. Contoh algoritma yang menggunakan metode ini yaitu: Competitive, Hebbian, Kohonen, LVQ (Learning Vektor Quantization), Neocognitron.

c. *Hybrid Learning* (pembelajaran hibrida)

Merupakan kombinasi dari metode pembelajaran supervised dan unsupervised learning. Sebagian dari bobot-bobotnya ditentukan melalui pembelajaran terawasi dan sebagian lainnya melalui pembelajaran tak terawasi. Contoh algoritma yang menggunakan metode ini yaitu algoritma RBF.

## 2.7 *Backpropagation*

*Backpropagation* pertama kali dirumuskan oleh Mc. Culloch dan W. H. Pitts tahun 1940. Kemudian dikembangkan oleh Rumelhart, Hinton, dan William tahun 1986 (Anugerah, 2007). *Backpropagation* merupakan tipe jaringan saraf tiruan yang menggunakan metode pembelajaran terbimbing (*supervised learning*). Pada *supervised learning* terdapat pasangan data input dan output yang dipakai

untuk melatih JST hingga diperoleh bobot (*weight*) yang diinginkan. bobot adalah sambungan antar lapisan dalam JST. Algoritma ini memiliki proses pelatihan yang didasarkan pada interkoneksi yang sederhana, yaitu apabila keluaran memberikan hasil yang salah, maka bobot dikoreksi agar galat dapat diperkecil dan tanggapan JST selanjutnya diharapkan dapat mendekati nilai yang benar. *Backpropagation* juga berkemampuan untuk memperbaiki bobot pada lapis tersembunyi (*hidden layer*).

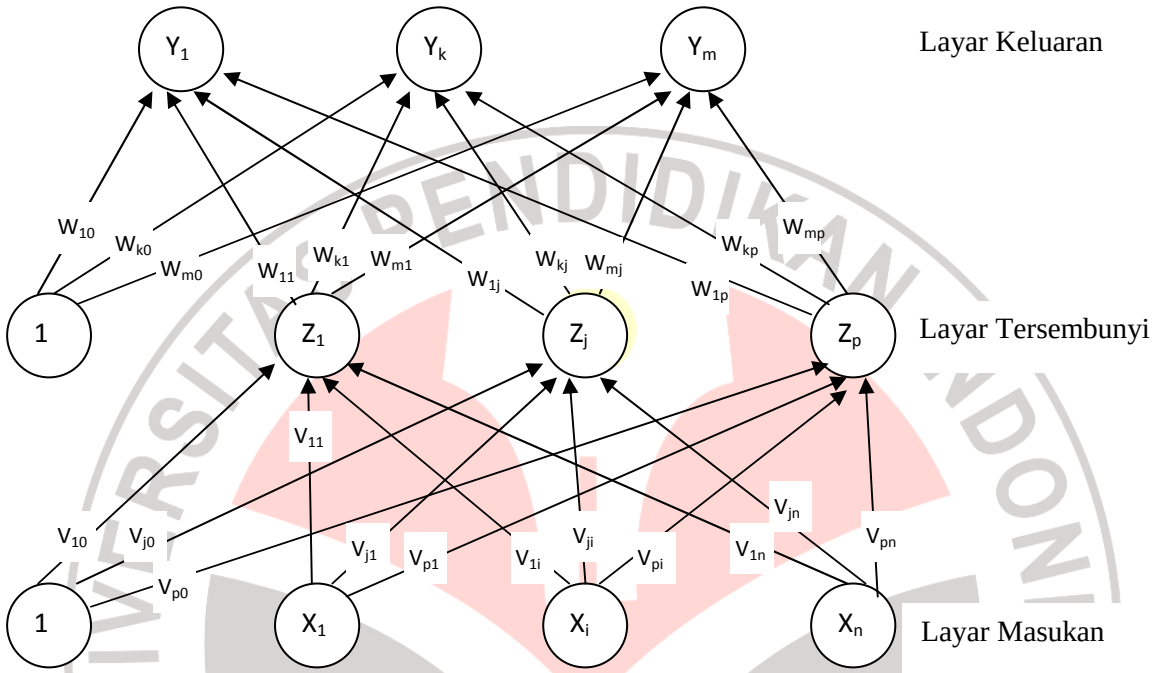
*Backpropagation* termasuk model JST dengan layar jamak. Seperti halnya model JST lainnya, *backpropagation* melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukan yang serupa (tapi tidak sama) dengan pola yang dipakai selama pelatihan.

### 2.7.1 Arsitektur *Backpropagation*

Arsitektur *backpropagation* terdiri dari tiga layar, yaitu layar masukan, layar tersembunyi, dan layar keluaran. Masing-masing layar terdiri dari beberapa unit. Untuk layar tersembunyi, dapat juga terdiri dari beberapa layar. Di bawah ini merupakan arsitektur *backpropagation* dengan *n* buah masukan (ditambah sebuah bias), sebuah layar tersembunyi yang terdiri dari *p* unit (ditambah sebuah bias), serta *m* buah unit keluaran.

$V_{ji}$ merupakan bobot garis dari unit masukan $X_i$ ke unit layar tersembunyi $Z_j$ ($V_{j0}$ merupakan bobot garis yang menghubungkan bias di unit masukan ke unit

layar tersembunyi $Z_j$). $W_{kj}$ merupakan bobot dari unit layar tersembunyi $Z_j$ ke unit keluaran $Y_k$ ($W_{k0}$ merupakan bobot dari bias di layar tersembunyi ke unit keluaran $Z_k$).




Gambar 2.6 Arsitektur *Backpropagation* (Siang, 2009)

### 2.7.2 Fungsi Aktivasi *Backpropagation*

Dalam *backpropagation*, fungsi aktivasi yang dipakai harus memenuhi beberapa syarat sebagai berikut. (Siang, 2009)

a. Kontinu.

b. Terdiferensial dengan mudah.

c. 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) dan fungsi sigmoid bipolar dengan range (-1,1).

Fungsi sigmoid memiliki nilai maksimum 1. Untuk pola yang targetnya lebih dari 1, pola masukan dan keluaran harus terlebih dahulu ditransformasi sehingga semua polanya memiliki range yang sama seperti fungsi sigmoid yang dipakai. Untuk lebih jelasnya, fungsi aktivasi dapat dilihat pada poin 2.5.4.

### 2.7.3 Pelatihan *Backpropagation*

Pelatihan dilakukan dalam rangka perhitungan bobot sehingga pada akhir pelatihan akan diperoleh bobot-bobot yang baik. Selama proses pelatihan, bobot-bobot diatur secara iteratif untuk meminimumkan *error* (kesalahan) yang terjadi. *Error* (kesalahan) dihitung berdasarkan rata-rata kuadrat kesalahan atau *mean square error* (MSE).

Pelatihan *backpropagation* meliputi tiga fase, sebagai berikut. (Siang, 2009)

a. Propagasi maju

   Pola masukan dihitung maju mulai dari layar masukan hingga layar keluaran menggunakan fungsi aktivasi yang ditentukan.

b. Propagasi mundur

   Selisih antara keluaran jaringan dengan target yang diinginkan merupakan kesalahan yang terjadi. Kesalahan yang terjadi itu dipropagasi mundur. Dimulai dari garis yang berhubungan langsung dengan unit-unit di layar keluaran.

c. Perubahan bobot

Modifikasi bobot untuk menurunkan kesalahan yang terjadi. Ketiga fase tersebut diulang-ulang terus hingga kondisi penghentian dipenuhi.

Algoritma pelatihan untuk jaringan *backpropagation* dengan satu layar tersembunyi (dengan fungsi aktivasi sigmoid biner) adalah sebagai berikut. (Siang, 2009)

0) Inisialisasi semua bobot dengan bilangan acak kecil.

1) Jika kondisi penghentian belum dipenuhi, lakukan langkah 2-8.

2) Untuk setiap pasang data pelatihan, lakukan langkah 3-8.

**Langkah 3-5 merupakan fase 1**

3) Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi di atasnya.

4) Hitung semua keluaran di unit tersembunyi $z_j$ (j = 1, 2,..., p).

$$z\_net_j = v_{j0} + \sum_{i=1}^{n} x_i \, v_{ji}$$

$$z_j = f\left(z\_net_j\right) = \frac{1}{1 + e^{-z\_net_j}}$$

5) Hitung semua keluaran jaringan di unit keluaran $y_k$ (k = 1, 2,..,m).

$$y\_net_k = w_{k0} + \sum_{j=1}^{p} z_j \, w_{kj}$$

$$y_k = f\left(y\_net_k\right) = \frac{1}{1 + e^{-y\_net_k}}$$

**Langkah 6-7 merupakan fase 2**

6) Hitung faktor $\delta$ unit keluaran berdasarkan kesalahan di setiap unit keluaran $y_k$

(k = 1, 2,..., m).

$\delta_k = (t_k - y_k)\, f'(y\_net_k) = (t_k - y_k)\, y_k(1 - y_k)$

dengan:

$t_k$ = target keluaran

$\delta_k$ merupakan unit kesalahan yang akan dipakai dalam perubahan bobot lapisan di bawahnya.

Hitung perubahan bobot wkj dengan laju pemahaman α.

$\Delta w_{kj} = \alpha\, \delta_k\, z_j$ dengan k = 1,2,..,m ; j = 0,1,..,p

7) Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi zj (j = 1, 2, ..., p).

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k\, w_{kj}$$

Faktor δ unit tersembunyi.

$\delta_j = \delta\_net_j\, f'(y\_net_j) = \delta\_net_j\, z_j(1 - z_j)$

Hitung suku perubahan bobot $v_{ji}$.

$\Delta v_{ji} = \alpha\, \delta_j\, x_i$ dengan j = 1,2,..,p ; i = 0,1,..,n

**Fase 3**

8) Hitung semua perubahan bobot. Perubahan bobot garis yang menuju ke unit keluaran, yaitu:

$w_{kj}$ (baru) = $w_{kj}$ (lama) + $\Delta w_{kj}$, (k=1,2,…,m ; j=0,1,…,p)

Perubahan bobot garis yang menuju ke unit tersembunyi, yaitu:

$v_{ji}$ (baru) = $v_{ji}$ (lama) + $\Delta v_{ji}$, (j=1,2,…,p ; i=0,1,…,n)

Variabel α merupakan laju pemahaman yang menentukan kecepatan iterasi. Nilai α terletak antara 0 dan 1 ($0 \leq \alpha \leq 1$). Semakin besar harga α, semakin sedikit iterasi yang dipakai. Akan tetapi jika harga α terlalu besar, maka akan merusak pola yang sudah benar sehingga pemahaman menjadi lambat. Satu siklus pelatihan yang melibatkan semua pola disebut epoch.

### 2.7.4 Analisis Kesalahan

Algoritma *backpropagation* adalah algoritma penurunan gradient yang menggunakan MSE (*mean square error*) untuk mengetahui nilai kesalahan bobot. Kesalahan dihitung setelah fase propagasi maju dimana jaringan menghasilkan nilai keluaran yang akan dibandingkan dengan nilai target.

Selain MSE, pengukuran kesalahan yang lain dapat menggunakan RMSE (*root mean square error*) yang diperoleh dari normalisasi MSE seperti pada persamaan 2.12 berikut ini. (Yoo, 2001)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(d_k - o_k)^2} \quad \text{.....(2.12)}$$

Dimana *n* adalah banyaknya pola pelatihan, $d_k$ adalah nilai target, dan $o_k$ nilai keluaran jaringan. RMSE dapat dipakai sebagai pengukur kesalahan pada *backpropagation* yang pelatihannya sedang berlangsung sampai nilai RMSE kurang dari nilai toleransi yang telah ditentukan.

Selain RMSE, untuk mengetahui tingkat akurasi ramalan, digunakan *Mean Absolute Percentage Error* (MAPE). (Anugerah, 2007)

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_t - y'_t}{y_t} \text{ x } 100\%\right| \quad \text{.....(2.13)}$$

dengan,

$y_t$ = nilai aktual pada waktu t

$y'_t$ = nilai ramalan pada waktu t

$n$ = banyaknya ramalan

### 2.7.5 Epoch Maksimum dan Batas Nilai Toleransi

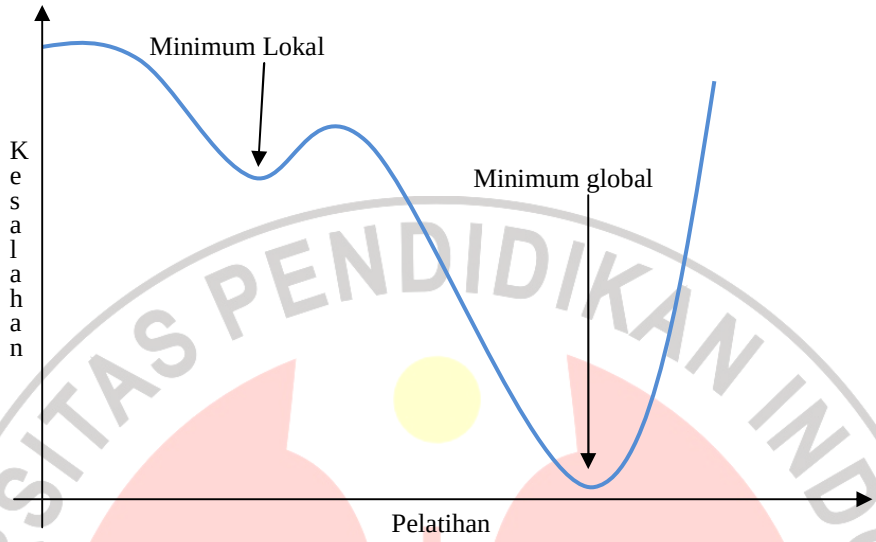
Dalam proses pelatihan jaringan, untuk mendapatkan bobot yang tepat diperlukan banyak epoch. Apabila jumlah epoch terlalu sedikit, jaringan tidak dapat mengenali pola. Tetapi apabila epoch terlalu banyak, maka akan terjadi *over-fitting* atau dengan kata lain jaringan hanya dapat mengenali pola data pelatihan saja, tidak untuk pola baru (Ajoy, 2005).

Untuk menghindarinya, maka selain harus menentukan jumlah *epoch* maksimum, nilai toleransi pun ditentukan. Nilai toleransi merupakan batas nilai minimum kesalahan sebagai penanda berakhirnya pelatihan. Untuk menghitung kesalahan jaringan pada saat pelatihan, digunakan RMSE. Penjelasan mengenai RMSE dapat dilihat pada poin 2.7.4.

### 2.7.6 Optimalisasi *Backpropagation*

Pada dasarnya, *backpropagation* merupakan algoritma penurunan gradient yang setiap perubahan bobot akan semakin mendekati ke kesalahan minimum.

Pada *backpropagation*, kesalahan minimum yang diraihnya dapat berupa minimum lokal dan global seperti yang tergambar pada gambar 2.7 berikut.




Gambar 2.7 Kurva Kesalahan Bobot (Rojas, 1996)

Kesalahan minimum perlu dihindari karena bukan merupakan solusi dari bobot yang tepat. Untuk menghindari tercapainya lokal minimum, dapat dilakukan optimalisasi. Optimalisasi *backpropagation* dapat dilakukan dengan dua cara, yaitu: (Yoo, 1999)

a. Metode numerik

   Dalam menghitung perubahan bobot, di teknik ini dilakukan dengan cara matematis, contohnya dengan menggunakan algoritma Newton dan Quasi-Newton.

b. Teknik heuristik

   Teknik ini dapat dilakukan dengan memodifikasi variabel laju pembelajaran. Ketika laju pembelajaran kecil, penurunan kesalahan bobot menjadi lambat. Tetapi ketika laju pembelajaran besar, penurunan kesalahan bobot menjadi

cepat dan dapat menyebabkan terlampauinya bobot yang tepat. Hal ini dapat diatasi dengan penyesuaian laju pembelajaran, dalam artian laju pembelajaran dapat berubah-ubah selama pembelajaran. Pada penelitian ini, akan dilakukan teknik heuristik untuk mengoptimalkan jaringan.

### 2.7.6.1 Pemilihan Bobot dan Bias Awal

Bobot awal akan mempengaruhi apakah jaringan mencapai titik minimum lokal atau global dan seberapa cepat konvergensinya. Bobot yang menghasilkan nilai turunan aktivasi yang kecil sedapat mungkin dihindari karena akan menyebabkan perubahan bobot yang semakin kecil. Demikian pula nilai bobot awal tidak boleh terlalu besar karena nilai turunan fungsi aktivasinya menjadi sangat kecil juga. Oleh karena itu dalam standar *backpropagation*, bobot dan bias diisi dengan bilangan acak kecil. (Siang, 2009)

Nguyen dan Widrow (1990) mengusulkan cara membuat inisialisasi bobot dan bias ke unit tersembunyi sehingga menghasilkan iterasi lebih cepat. Algoritma inisialisasi Nguyen dan Widrow adalah sebagai berikut: (Siang, 2009)

1) Inisialisasi semua bobot ($V_{ji}$ lama) dengan bilangan acak dalam interval [-0,5; 0,5]
2) Hitung $\|V_j\| = \sqrt{V^2{}_{j1} + V^2{}_{j2} + \cdots + V^2{}_{jn}}$
3) Hitung faktor skala ($\beta$) = $0{,}7\sqrt[n]{p}$ ; dengan n = jmlh simpul masukan dan p = jmlh simpul tersembunyi
4) Bobot yang dipakai sebagai inisialisasi = $V_{ji} = \frac{\beta\, V_{ji}\,(lama)}{\|V_j\|}$

5) Bias yang dipakai sebagai inisialisasi = $V_{j0}$ = bilangan acak antara $-\beta$ dan $\beta$

### 2.7.6.2 Penambahan Momentum

Pada standar *backpropagation*, perubahan bobot didasarkan pada gradient yang terjadi untuk pola yang dimasukkan saat itu. Modifikasi yang dapat dilakukan adalah mengubah bobot yang didasarkan pada arah gradient pola terakhir dan pola sebelumnya (disebut momentum) yang dimasukkan. Jadi tidak hanya pola masukan terakhir saja yang diperhitungkan. (Siang, 2009)

Penambahan momentum dimaksudkan untuk menghindari perubahan bobot yang mencolok akibat adanya data yang berbeda dengan yang lainnya. Apabila beberapa data terakhir yang diberikan ke jaringan memiliki pola serupa, maka perubahan bobot dilakukan secara cepat. Namun apabila data terakhir yang dimasukkan memiliki pola yang berbeda dengan pola sebelumnya, maka perubahan dilakukan secara lambat.

Dengan penambahan momentum, bobot baru pada waktu ke (t+1) didasarkan atas bobot pada waktu t dan (t-1). Di sini harus ditambahkan dua variabel baru yang mencatat besarnya momentum untuk dua iterasi terakhir. Jika $\mu$ adalah konstan ($0 \leq \mu \leq 1$) yang menyatakan variabel momentum maka bobot baru dihitung berdasarkan persamaan:

$$w_{kj}(t+1) = w_{kj}(t) + \alpha\, \delta_k\, z_j + \mu(w_{kj}(t) - w_{kj}(t-1)) \quad \text{.....(2.14)}$$

dan

$$v_{ji}(t+1) = v_{ji}(t) + \alpha\, \delta_j\, x_i + \mu(v_{ji}(t) - v_{ji}(t-1)) \quad \text{.....(2.15)}$$

### 2.7.6.3 *Dynamic Adaptation of Learning Rate* dan *Momentum*

Dalam mengubah bobot, jika *learning rate* terlalu besar, perubahan bobot menjadi besar juga. Dan kemungkinan besar proses training menjadi cepat. Tetapi hal itu tidak bisa menjamin karena *oscillation* (ketidakstabilan) bisa tercapai. Agar menghindari hal tersebut, maka *learning rate* harus dimaksimalkan untuk mempercepat konvergensi dalam suatu rentang untuk mencegah *oscillation*.

*Oscillation* ditandai dengan ketidaksesuaian fluktuasi dari penurunan dan penaikan melalui pengukuran kesalahan RMSE. Dan bisa juga dideteksi dengan suatu interval yang ditetapkan pada *dynamic adaptation* dari *learning rate* dan momentum.

Dalam menganalisis kesalahan kurva, kemungkinan besar konvergen (berkumpul) di global minimum ketika kesalahan menurun atau menaik di interval yang telah ditentukan. Oleh karena itu, ketika frekuensi kesalahan menurun di bawah nilai minimum atau loncat di atas nilai maksimum yang telah ditentukan di analis kesalahan, *learning rate* dan momentum dimodifikasi. Dengan catatan, *learning rate* dan momentum diinisialisasi terlebih dahulu. Berikut ini adalah algoritma *dynamic adaptation of learning rate* dan *momentum*. (Yoo, 2001)

```
delta_error = average_rms t-1 – average_rms t ;
if delta_error < 0,0 then
        oscillation := oscillation + 1;
else if delta_error >= 0,0 then
        oscillation := oscillation – 1;
end if;
```

```
if  reference_interval = true then
        if (oscillation > max_freq) || (oscillation < min_freq) then
                learning_rate = (init_learning_rate * (interval_size - oscillation)) /
                                interval_size;
                momentum = (init_momentum * ((1,0 – init_learning_rate)) +
                                learning_rate);
                oscillation = 0;
        end if;
end if;
```

Untuk menerapkan algoritma ini, *learning rate* dan momentum harus diinisialisasi terlebih dahulu. Di samping itu, interval (reference_interval) untuk analisis kesalahan dan frekuensi penurunan kesalahan (min_freq, max_freq) untuk mendeteksi *oscillation* harus didefinisikan.