BAB II

TINJAUAN PUSTAKA

2.1 Time Series

Time series atau runtun waktu merupakan suatu pengamatan terhadap variabel dari waktu lampau dan dicatat secara beruntut sesuai urutan waktu dengan periode waktu yang tetap (Hanke dan Winchern, 2004). Pada umumnya pencatatan ini dilakukan dalam periode tertentu misalnya harian, bulanan, tahunan dan sebagainya, metode time series juga merupakan metode peramalan dengan menggunakan analisa hubungan anatara variabel yang akan diperkirakan dengan variabel waktu. Peramalan suatu data times series perlu memperhatikan tipe atau pola data. Terdapat empat macam pola data time series yaitu horizontal, musiman, siklis dan trend (Makridakis *et al*, 1995). Pola horizontal terjadi bilamana nilai data fluktuasi disekitar nilai rata-rata yang konstan. Pola musiman terjadi bilamana suatu deret dipengaruhi oleh faktor musiman (misalnya kuartal tahun tertentu, bulanan, atau hari-hari pada minggu tertentu). Pola siklis terjadi bilamana datanya dipengaruhi oleh fluktuasi jangka panjang. Sedangkan pola *trend* terjadi bilamana terdapat kenaikan atau penurunan sekuler jangka panjang dalam data.

2.2 *Neural Network* (NN)

Jaringan syaraf tiruan (JST) atau sering dikenal dengan istilah *neural network* adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan syaraf biologi manusia (Siang, 2005). *Neural network* telah diaplikasikan dalam berbagai bidang diantaranya *pattern recognition, medical*

diagnostic, signal processing, dan peramalan. Pada dasarnya, *neural network* merupakan kumpulan dari elemen-elemen pemroses yang saling berhubungan, yang disebut dengan *unit-unit* atau syaraf-syaraf (Suhartono, 2007). *Neural network* bekerja berdasarkan pola yang terbentuk pada *input*nya. Setiap *neuron* dihubungkan dengan *neuron* lainnya dengan suatu *connection link*, yang direpresentasikan dengan *weight* atau bobot. Metode untuk menentukan nilai *weight* disebut dengan *training*, learning, atau *algoritma*. Setiap *neuron* menggunakan fungsi aktivasi pada net *input* untuk menentukan prediksi *output*.

Neural network termasuk dalam salah satu bentuk *time series* nonlinier dan memiliki bentuk fungsional yang fleksibel sehingga neural network tidak dapat diintrepretasikan atau tidak memiliki asumsi-asumsi yang harus dipenuhi. Neuron-neuron dalam neural network disusun dalam grup, yang disebut dengan layer (lapis). Secara garis besar pada *Neural Network* (NN) memiliki dua tahapan dalam system pemrosesan informasi, yaitu:

a) Tahap pelatihan (*training*)

Tahapan ini dimulai dengan memasukan data latih ke dalam jaringan (warsito, 2009). Dengan menggunakan data latih, jaringan akan mengubah-ubah bobot yang menjadi penghubung antar node. Pada setiap evaluasi dilakukan evaluasi terhadap *output* jaringan. Tahapan ini berlangsung pada beberapa iterasi dan berhenti setelah menemukan bobot yang sesuai dimana nilai *error* yang diinginkan telah tercapai atau jumlah iterasi telah mencapai nilai maksimal yang ditetapkan, selanjutnya bobot ini akan menjadi dasar pengetahuan pada tahapan pengenalan.

b) Tahap pengujian (*testing*)

Pengujian dilakukan dengan memasukan suatu pola yang belum pernah dilatih sebelumnya (data uji) dengan menggunakan bobot hasil tahap pelatihan. Diharapkan bobot-bobot hasil pelatihan yang telah menghasilkan error minimal juga akan menghasil error minimal pada tahap pengujian (Warsito, 2009).

Data training digunakan dalam proses pembentukan arsitektur terbaik dalam proses pelatihan, sedangkan data testing digunakan dalam pengujian keakuratan dari arsitektur yang telah terbentuk. Menurut (Hota, 2013) pilihan ukuran untuk data training dan testing yang dapat digunakan adalah sebagai berikut:

Tabel 2.1 Pembagian Data input, target dan pola

Data Input	**Data *Target***	**Pola**
42	12	13

2.3 Kompenan *Neural Network*

Neural network memilik karakteristik yang sama dengan karakteristik syaraf manusia, berdasarkan karakteristiknya neural network dibangun dengan komponen-komponen yang sama dengan manusia, diantaranya:

1. Neuron/Node

Bertugas memproses semua informasi yang diterima, sama dengan fungsi neuron pada otak manusia, dimana semua proses perhitungan dilakukan disini (Kusumdewi, 2004).

2. Input

Informasi yang diproses dalam neuron input berasal dari lingkungan ataupun dari node lain (Pandjaitan, 2007). *Neural network* hanya dapat memproses

data masukan berupa data numerik sehingga apabila ada masalah yang melibatkan data kualitatif seperti grafik, gambar, sinyal atau suara harus ditranformasikan terlebih dahulu kedalam data numerik.

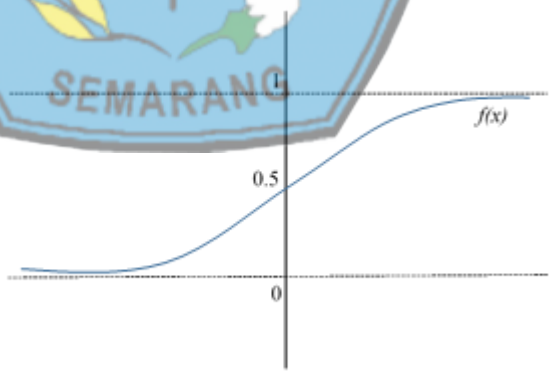
3. Fungsi aktivasi

Fungsi aktivasi yang akan menentukan apakah sinyal dari input neuron akan diteruskan atau tidak (Siang, 2005). Fungsi aktivasi jaringan syaraf tiruan yang dilatih menggunakan optimasi *levenberg marquardt* adalah fungsi sigmoid biner. Fungsi sigmoid biner mempunyai interval range 0 sampai 1, sehingga fungsi ini digunakan untuk jaringan syaraf tiruan yang memiliki nilai output yang terletak pada interval 0 sampai 1. Fungsi sigmoid biner dirumuskan sebagai :

$$y = f\ x = \frac{1}{1+e^{-x}} \qquad (2.1)$$

dengan,

$$y'\ x = (x)\ [1 - f\ (x)]$$



Gambar 2.1 Fungsi Aktivasi Sigmoid Biner

Fungsi sigmoid memiliki nilai maksimum = 1. Maka untuk pola yang targetnya > 1, pola *input* dan *output* harus terlebih dahulu ditransformasi sehingga semua polanya memiliki range yang sama seperti fungsi sigmoid yang dipakai.

4. Bias dan threshold

Bias dapat dipandang sebagai input yang bernilai 1. Bias berfungsi untuk mengubah nilai threshold menjadi 0. Sedangkan threshold berperan sebagai penimbang dalam suatu hubungan dari sebuah unit tertentu (Siang, 2005).

5. Bobot

Bobot merupakan suatu nilai yang menunjukan kekuatan antar node (Siang, 2005). Jika nilai bobot diatur berbeda, maka output yang dihasilkan akan berbeda. Bobot-bobot yang optimal akan memungkinkan system menterjemahkan data masukan secara benar dan menghasilkan keluaran yang diinginkan.

6. Output

Menurut (Pandjaitan, 2007) Suatu nilai yang dihasilkan dari fungsi aktivasi, yang bisa berupa output dari jaringan atau menjadi input bagi node lain disebut sebagai output. Data output merupakan data numerik.

2.4 Artifical *Neural Network*

Arsitektur jaringan dalam NN merupakan susunan dari neuron-neuron dalam lapisan input, hidden dan output yang terhubung dengan bobot, fungsi aktivasi dan fungsi pembelajaran. Arsitektur ini merupakan salah satu karakteristik penting yang membedakan *neural network*. Secara umum ada tiga lapis yang membentuk *neural network*:

1) Lapis input

Unit-unit di lapisan input disebut unit-unit input. Unit-unit input tersebut menerima pola inputan dari luar yang menggambarkan suatu permasalahan. banyak node atau neuron dalam lapis input tergantung pada banyaknya input dalam model

dan setiap input menentukan satu neuron.

2) Lapis tersembunyi (*hidden layer*)

Unit-unit dalam lapisan tersembunyi disebut unit-unit tersembunyi, di mana outputnya tidak dapat diamati secara langsung. Lapis tersembunyi terletak di antara lapis input dan lapis output, yang dapat terdiri atas beberapa lapis tersembunyi.

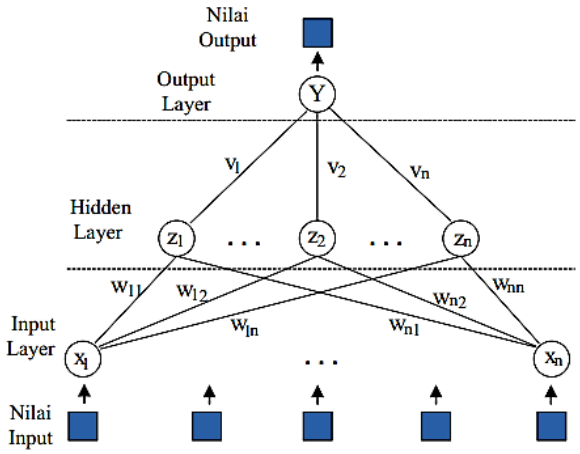
3) Lapis output

Unit-unit dalam lapisan output disebut unit-unit output. Output dari lapisan ini merupakan solusi *Neural Network* terhadap suatu permasalahan. Setelah melalui proses training, network merespon input baru untuk menghasilkan output yang merupakan hasil peramalan.

Berdasarkan jumlah layer yang dimiliki, neural network dibedakan menjadi:

a. Jaringan lapisan tunggal (Single Layer)

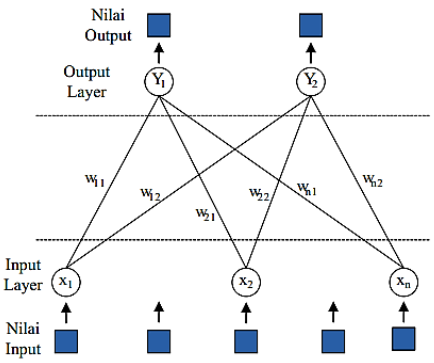
Jaringan lapisan tunggal atau single layer hanya memiliki dua lapisan yakni lapisan input dan lapisan output, dimana lapisan input berperan dalam menerima sinyal data input sedangkan lapisan output berperan sebagai media dalam memberikan hasil output. Layer input disusun oleh beberapa neuron yang dihubungkan oleh bobot menuju layer output dalam satu alur maju dan tidak sebaliknya. Itulah sebabnya arsitektur ini disebut sebagai arsitektur umpan maju (*feedforward*). Walaupun arsitektur ini terdiri dari dua layer, namun arsitektur ini dikategorikan sebagai arsitektur layer tunggal karena layer output secara tunggal melakukan proses komputasi tanpa melibatkan layer lain diantara layer input dan output. Arsitektur dari jaringan single layer digambarkan sebagai berikut:



Gambar 2.2 *Single Layer Network* (Siregar, 2013)

b. Jaringan multi lapis (*Multi Layer*)

Jaringan jenis ini yaitu lapisan layar jamak mempunyai ciri-ciri tertentu ialah memiliki tiga jenis layer yaitu input layer, output layer, dan juga layar tersembunyi atau disebut juga hidden layer. Jaringan yang memiliki beberapa lapisan ini dapat membantu menyelesaikan suatu permasalahan yang lebih rumit daripada menggunakan jaringan yang mempunyai layar tunggal. Tetapi, proses untuk melatih jaringan ini membutuhkan waktu yang lebih lama. Contoh pada algoritma neural network dengan menggunakan suatu metode jenis ini ialah recurrent. Pada gambar 2.3 dapat diamati arsitektur jaringan layar jamak yang mempunyai n buah layar input (X_1, X_2, …, Xn), n buah layar terssembunyi (Z_1, Z_2, …, Zn) dan sebuah layar output (Y) dengan bobot yang menghubungkan layar input dengan layar tersembunyi (W_{11},W_{12}, W_1n, Wn_1, Wn_2, Wnn) dan bobot yang menghubungkan layar tersembunyi dengan layar output (V_1, V_2, …, Vn).



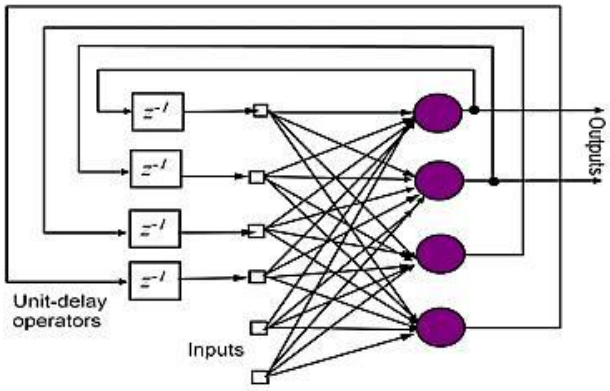
Gambar 2.3 *Multi Layer Network* (Siregar, 2013)

Dalam penelitian ini, digunakan jaringan *multi layer* dengan model *recurrent neural network* (RNN). RNN adalah suatu jaringan *multi layer* yang memiliki paling sedikit satu umpan balik berulang.

Pada pemodelan RNN untuk data *time series*, *input* model adalah jumlah penumpang pesawat terbang dan ouputnya adalah peramalan jumlah pesawat terbang (Huda, A. M. 2009).

c. *Recurrent Network*

Recurrent network memiliki minimal jumlah paling sedikitnya terdapat satu *feedback loop* didalam *recurrent*. *Feedback* loop mempengaruhi kemampuan belajar dan kinerja jaringan (Udin, 2016).



Gambar 2.4 Struktur *Recurrent Network* (Siregar, 2013)

Metode pelatihan jaringan merupakan suatu proses atau prosedur pelatihan jaringan yang merupakan urutan langkah-langkah algoritma untuk memodifikasi nilai-nilai bobot dan bias dengan tujuan agar suatu jaringan mendapat nilai-nilai bobot dan bias yang sesuai sehingga dapat menghasilkan output jaringan yang diinginkan. Jika kesalahan pada output jaringan sangat kecil, maka dapat dikatakan telah diperoleh nilai-nilai bobot dan bias yang sesuai dan jaringan tersebut telah mencapai jaringan yang baik. Dalam penelitian ini digunakan nilai MAPE (*Mean Absolute Persentage Error*) untuk mengukur kesalahan nilai dugaan model yang dinyatakan dalam bentuk rata-rata persentase absolute residual. Nilai MAPE dapat dirumuskan sebagai berikut:

$$MAPE=\frac{\sum APE}{n} \quad (2.2)$$

dimana,

n : banyaknya data yang diprediksi

APE : *absolute percentage error*

Interpretasi dari perhitungan MAPE menurut (Chen *et al*, 2007) adalah sebagai berikut:

1. nilai MAPE < 10% sangat baik untuk melakukan prediksi
2. nilai MAPE berada diantara 10% - 20% menghasilkan nilai prediksi yang baik
3. nilai MAPE berada diantara 20% - 50% menghasilkan prediksi yang proporsional/wajar artinya masih dapat digunakan dalam memprediksi
4. Nilai MAPE > 50% tidak dapat digunakan dalam prediksi

2.5 Metode Pelatihan

Jaringan neural network mempunyai karakteristik yang penting selain arsitektur, yakni metode pengaturan nilai bobot / Training (Fausset, 1994). Metode pelatihan pada *neural network* dibagi menjadi dua jenis, yakni:

1. Pelatihan Terawasi

Pelatihan ini dilakukan dengan adanya urutan vektor pelatihan, atau pola masing-masing terkait dengan target *output*, kemudian bobot disesuaikan untuk *algoritma* pembelajaran.

2. Pelatihan Tak Terawasi

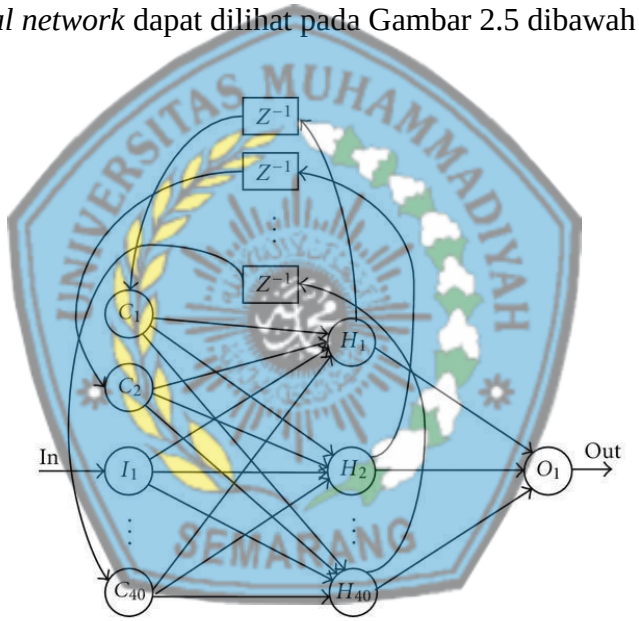
Pada pelatihan ini *neural network* mengatur segala kinerja dirinya sendiri, mulai dari masukan target hingga menggunakan data *training* untuk melakukan pembelajaran .

Pada penelitian ini, yang digunakan sebagai metode untuk mengatur nilai bobot dalam *levenberg marquardt* adalah metode pelatihan terawasi (Kusumadewi, 2004).

2.6 *Recurrent Neural Network*

Recurrent Neural Network ialah jenis neural network yang memiliki kemampuan *feedback* kembali ke neuron itu sendiri ataupun ke neuron yang lainnya, sehingga aliran informasi dari masukan mempunyai arah jamak (Aribowo dalam Ammar, 2014). RNN ialah suatu jaringan yang memiliki paling sedikit satu umpan balik berulang. *Feedback loop* dapat mempengaruhi kemampuan belajar dan kinerja suatu jaringan. RNN mempunyai kemampuan penggambaran yang sangat bagus dan dapat mengatasi kelemahan *feedforward* (Soelaiman & Rifa'i dalam

Ammar, 2014). Output dari RNN tidak bergantung hanya pada input pada waktu itu saja, namun juga bergantung pada kondisi input neural network untuk waktu masa lampau. Kondisi seperti ini dimaksudkan agar jaringan dapat menampung beberapa kejadian pada masa lampau yang diikuti pada proses perhitungan. Hal ini penting untuk masalah yang cukup rumit, dan tanggapan keluaran neural network berkaitan dengan variasi waktu (*time-varying*), sehingga *neural network* memiliki kepekaan terhadap waktu dengan memori kondisi lampau (Ammar, 2014). Struktur *recurrent neural network* dapat dilihat pada Gambar 2.5 dibawah ini.



Gambar 2.5 Struktur *recurrent neural network* (Jaroslaw S., 2011)

Output dari *recurrent neural network* dapat dilakukan perhitungan dengan menggunakan persamaan seperti berikut ini:

$$y(k) = x(k+1) - Ax(k) - V_0^1 - \sigma[W_1^0 x(k)] - V_2^0 \theta[W_1^0 x(k)]u(k) \qquad (2.3)$$

Keterangan:

$x(k) \in n$ ialah *state vector* pada internal

$A \in \Re$ *nxn* ialah matrik yang mempunyai nilai tetap

$x(k) \in \Re$ n ialah *state vector*

$u(k) \in \Re$ m ialah *input* dari vektor, dengan nilai $u(k)$ dan $x(k)$ diketahui

$mxnk$ k W $W \in \Re$ 1, 2, ialah suatu bobot yang terdapat pada lapisan tersembunyi

nxm k k V $V \in \Re$ 1, 2, ialah suatu bobot yang terdapat pada lapisan keluaran

$x(k+1)$ ialah nilai pada *state vector* yang merupakan hasil perkiraan pada saat iterasi ke $k+1$

Fungsi aktivasi pada jaringan yang akan digunakan ialah seperti berikut:

$$T\ m\ [\cdots]|\sigma = \sigma\sigma, \sigma\epsilon\ \mathcal{R}\ m\ dan\ \vartheta\epsilon\ \mathcal{R}\ mxm \quad (2.4)$$

Nilai σ dan φ dihitung dengan menggunakan fungsi sigmoid (Soelaiman dan Rifa'i dalam Ammar, 2014).

2.7 *Levenberg Marquardt*

Levenberg marquardt ialah algoritma yang dikembangkan dari algoritma error pada metode backpropagation. Pada algoritma jenis ini digunakan untuk menangani beberapa kelemahan algoritma error pada metode backpropagation dengan menggunakan optimisasi numerik standar ialah dengan penggunaan pendekatan jacobian matrix. Tujuan dari *levenberg marquardt* adalah meminimalkan total error. (Susanti, 2014). Algoritma pada *levenberg marquardt* ialah salah satu cara untuk menangani kelemahan dari kinerja algoritma backpropagation, dikarenakan algoritma levenberg marquardt dibuat sedemikian rupa agar dapat meminimalisasi jumlah kuadrat error. Algoritma jenis ini dapat memperbarui weight dengan memperkecil nilai error dari selisih *new weight* dan *old weight*. Tetapi, pada saat yang bersamaan algoritma ini akan mempertahankan step *size* agar tidak menjadi besar. (Kurniawan, 2012).

$$W_{new} = W_{wold} - (Z^T\ Z)^{-1} Z^T \epsilon(W_{old}) \quad (2.5)$$

Keterangan:

W_{new} ialah bobot yang baru

W_{old} ialah bobot yang lama

$\in$ ialah vector dari nilai error

Z ialah matrik dari hasil selisih error terhadap bobot.

Algoritma levenberg marquardt mempunyai jumlah nilai iterasi yang lebih bagus jika dibandingkan dengan algoritma pada optimasi backpropagation lainnya. Tetapi, algoritma ini membutuhkan memori yang lebih banyak dibandingkan algoritme sejenisnya (Kurniawan, 2012).

Beberapa struktur algoritma levenberg marquardt (Oktaorora, 2011):

1. Menginisialisasi bias dan juga bobot dengan menggunakan bilangan random, *epoch* yang maksimum, dan juga nilai minimal pada goal (*performance* dihitung menggunakan MSE).
2. Menentukan beberapa parameter, seperti di bawah ini:
   a. Parameter pada *levenberg marquardt* dengan nilai yang harus lebih besar dari 0.
   b. Parameter dari faktor nilai Beta (β) pada algoritma ini digunakan menjadi sebuah parameter yang akan dibagi atau dikalikan dengan menggunakan parameter *levenberg marquardt.*
3. Menghitung maju (*feedforward*) pada layar tersembunyi dan layar keluaran seperti pada langkah-langkah algoritma error pada metode *backpropagation*.
4. Menghitung nilai dari pada MSE.

5. Menghitung nilai error dan juga nilai total error pada jaringan dengan rumus menghitung error:

$$e_r = t_r - y_r \quad (2.6)$$

Keterangan :

r ialah masukan ke-r

Rumus berikut digunakan untuk menghitung nilai dari total error:

$$e = [e_1 e_2 e_3 \cdots e_n)^T \quad (2.7)$$

Keterangan :

e ialah nilai vektor pada kesalahan yang mempunyai ukuran Nx1 dan terdiri atas en r = 1, 2, 3, 4, ..., n

6. Menghitung matriks Jacobian J(x)

Dengan x ialah matriks yang mempunyai nilai bias dan bobot yang terdapat pada jaringan.

X = [v11, v12, ..., vij; v01, v02, ..., v0j ; w11, w12 ..., wjk ; w01, w02 ..., w0K] (2.8)

Matriks Jacobian mempunyai rumus turunan pertama dari nilai error suatu jaringan terhadap nilai dari bias dan bobotnya. Rumus yang digunakan untuk menghitung nilai matriks jacobian ialah seperti berikut:

$$J = \left[\frac{\vartheta_{er}}{\vartheta_w}\right] \quad (2.9)$$

7. Setelah nilai dari J(x) didapat, selanjutnya dapat melakukan perhitungan nilai bias dan perubahan koreksi bobot dengan menggunakan rumus di bawah ini:

$$\Delta x = [J(x)^t](x) + \mu I]^{-1} * Gradien \quad (2.10)$$

8. Setelah nilai Δx didapat, selanjutnya ialah melakukan koreksi nilai bobot dengan menggunakan rumus seperti yang ada pada algoritma nilai error pada metode *backpropagation.*
9. Menghitung maju (*feedforward*) menggunakan nilai bias dan bobot yang baru.
10. Menghitung nilai MSE jaringan menggunakan bias dan juga bobot yang baru. Kemudian tes pada kondisi saat berhenti. Jika nilai iterasi terus berjalan, maka akan menghasilkan dua kondisi yang memungkinkan seperti berikut:
    a. Jika nilai MSE naik maka μ x β
    b. Jika MSE turun $\frac{\mu}{\beta}$
8. Terakhir dengan melakukan pengulangan kembali dari langkah ke 5 sampai dengan langkah ke 8.