BAB II
KAJIAN TEORI

A. Minyak Mentah Indonesia

Minyak merupakan salah satu sumber energi yang sangat dibutuhkan manusia dalam berbagai aspek kehidupan. Sebagai salah satu sumber energi yang sangat dibutuhkan oleh manusia, permintaan minyak pasti akan terus meningkat seiring berjalannya waktu dan peningkatan permintaan minyak tersebut akan berpengaruh pada kenaikan harga minyak mentah itu sendiri. Saat ini penetapan harga minyak mentah dunia didasarkan pada dua kelompok atau standar yang umum yaitu Brent (Brent Crude) dan WTI (West Texas Intermediate). Sedangkan di Indonesia, harga basket minyak mentah dunia digunakan untuk perhitungan besarnya ICP (*Indonesian Crude Price*). ICP (*Indonesian Crude Price)* atau harga minyak mentah Indonesia merupakan basis harga minyak mentah Indonesia di pasar internasional yang digunakan dalam APBN serta indikator perhitungan pajak migas yang diperoleh kontraktor migas. Besarnya ICP (*Indonesian Crude Price)* ditetapkan tiap bulan dengan satuan US/barel oleh Direktorat Jendral Minyak dan Gas serta Departemen Energi dan Sumber Daya Mineral Republik Indonesia.

1. Ketersediaan Minyak Mentah di Indonesia

Minyak bumi merupakan salah satu sumber daya alam yang tidak dapat diperbarui. Sedangkan permintaan masyarakat akan minyak bumi selalu meningkat untuk kebutuhan sehari-hari. Hal tersebut menyebabkan ketersedian semakin hari semakin berkurang. Dirjen Migas Kementrian ESDM, IGN

Wiratmaja Puja menjelaskan bahwa pada tahun 2016 cadangan minyak negara Indonesia terhitung hanya tersisa sebanyak 3,7 milyar barel. Dalam 10 sampai 20 tahun lagi cadangan minyak Indonesia akan habis (Daniel, 2012).

Jika konsumsi minyak mentah terus menerus meningkat tanpa diimbangi dengan ketersediaan maka akan terjadi kelangkaan sumber daya minyak. Dampak ketidakseimbangan antara produksi dan konsumsi yang telah terjadi saat ini salah satunya adalah kenaikan harga minyak yang berfluktuatif.

2. Faktor-faktor yang Mempengaruhi Harga Minyak Mentah di Indonesia

Kestabilan harga minyak mentah di Indonesia sangat diperlukan untuk mendorong pertumbuhan ekonomi Indonesia. Harga minyak mentah Indonesia sangat digunakan oleh pemerintah, kontraktor migas dan investor. Dengan demikian penting bagi pemerintah, kontraktor migas dan investor untuk mengetahui harga minyak mentah di Indonesia di masa yang akan datang.

Penetapan harga minyak mentah di Indonesia dipengaruhi oleh 3 faktor, yaitu sebagai berikut.

1) Faktor fundamental, yang terdiri atas permintaan minyak, pasokan minyak, stok minyak, kapasitas produksi cadangan dunia dan kemampuan kilang dunia.
2) Faktor non fundamental, yang terdiri dari geopolitik, kebijakan pemerintah, cuaca, bencana alam, pemogokan, kerusakan instalasi ranai produksi, pelemahan nilai dollar dan spekulasi.
3) Pengaruh dari kebijakan pasokan *Organization of the Petroleum Exporting Countries* (OPEC).

B. Time Series

Time series adalah serangkaian pengamatan terhadap suatu peristiwa, kejadian, gejala atau peubah yang diambil dari waktu ke waktu, dicatat secara teliti menurut urutan waktu dan kemudian disusun sebagai data statistik (Hanke & Wichern, 2005:58). Data *time series* dapat di analisis menggunakan metode analisis *time series*. Beberapa konsep dasar dalam analisis *time series* adalah plot data, autokorelasi, autokorelasi parsial dan konsep *white noise*.

1. Stasioner

Suatu data *time series* dikatakan stasioner, apabila nilai rata-rata dan variansi dari data tersebut tidak mengalami perubahan (Hanke & Wichern, 2005:67). Data yang stasioner atau tidak stasioner dapat dilihat dari plot data *time series*. Bila fluktuasi data berada di sekitar suatu nilai rata-rata dan variansi yang konstan, maka data tersebut sudah stasioner. Selain itu juga bisa digunakan plot autokorelasi.

2. Autokorelasi *(Autocorrelation)*

Autokorelasi adalah asosiasi atau ketergantungan antara nilai-nilai suatu *time series* yang sama pada periode waktu yang berlainan dan digunakan untuk menentukan koefisien korelasi pada *time series*.

Autokorelasi pada *lag-k* (ρ_k) merupakan suatu korelasi pada data *time series* antara pengamatan Y_t dan Y_{t+k} yang didefinisikan sebagai berikut (Wei, 2006:10):

$$\rho_k = \frac{E[(Y_t-\mu)(Y_{t+k}-\mu)]}{\sqrt{E[(Y_t-\mu)^2]E[(Y_{t+k}-\mu)]^2}} = \frac{Cov(Y_t,Y_{t+k})}{var(Y_t)} = \frac{\gamma_k}{\gamma_0} \qquad (2.1)$$

dengan

μ : rata-rata

γ_k : autokovariansi pada *lag k*

ρ_k : autokorelasi pada *lag k*

t : waktu pengamatan, $t = 1,2,3, \ldots$

Y_t : pengamatan pada saat *t*

Y_{t+k} : pengamatan pada saat $t + k$

$var(Y_t) = var(Y_{t+k}) = \gamma_0$

Nilai-nilai ρ_k pada saat $k = 1,2,3, \ldots$ disebut fungsi autokorelasi (*autocorrelation function*/ACF). Nilai dugaan dari ρ_k yaitu $\hat{\rho}_k$ (atau r_k) diestimasi dengan menggunakan nilai dari autokorelasi sampel yaitu autokorelasi antara pengamatan pada waktu *t* sampai pengamatan pada waktu *t+k*, digunakan rumus sebagai berikut (Tsay, 2010:31) :

$$\hat{\rho}_k = r_k = \frac{\sum_{t=k+1}^{n}(Y_t - \bar{Y})(Y_{t+k} - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2}, \quad k = 1,2,3, \ldots \tag{2.2}$$

dengan

r_k : autokorelasi sampel pada *lag k*

$\bar{Y}$: rata-rata dari pengamatan $\{Y_t\}$

Y_t : pengamatan pada saat *t*

Y_{t+k} : pengamatan pada saat $t + k$

Pengujian signifikansi autokorelasi digunakan untuk mengetahui apakah autokorelasi berbeda signifikan dari nol. Hipotesis yang digunakan adalah:

$H_0: \rho_k = 0$ (autokorelasi pada *lag k* tidak berbeda signifikan dari nol)

$H_1: \rho_k \neq 0$ (autokorelasi pada *lag k* berbeda signifikan dari nol)

Statistik uji yang digunakan adalah:

$$t = \frac{r_k}{SE(r_k)} \text{ dengan } df = n - 1 \quad (2.3)$$

Standar *error* dari koefisien autokorelasi dirumuskan sebagai berikut (Hanke & Wichern, 2005:64):

$$SE(r_k) = \sqrt{\frac{1+2\sum_{i=1}^{k-1} r_i^2}{n}} \quad (2.4)$$

dengan

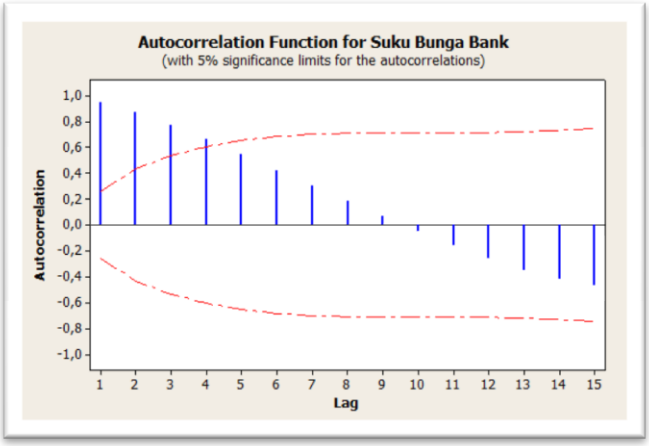
$SE(rk)$	:	standar *error* autokorelasi pada *lag* k
r_k	:	autokorelasi sampel pada *lag* k
n	:	banyak pengamatan

Jika nilai $t_{hitung} > t_{n-1}\left(\frac{\alpha}{2}\right) atau\ t_{hitung} < -t_{n-1}\left(\frac{\alpha}{2}\right)$ maka H_0 ditolak. Artinya autokorelasi dikatakan berbeda secara signifikan dari nol. Sedangkan jika nilai t_{hitung} memenuhi $-t_{n-1}\left(\frac{\alpha}{2}\right) < t_{hitung} < t_{n-1}\left(\frac{\alpha}{2}\right)$ maka autokorelasi dikatakan tidak berbeda secara signifikan dari nol.

Signifikansi autokorelasi dapat ditentukan dengan melihat *correlogram. Correlogram* adalah plot antara *lag k* dengan r_k, dimana $r_k = 0$ adalah pusat selang kepercayaan, sedangkan garis putus-putus merupakan batas atas dan bawah dari selang kepercayaan. Selang kepercayaan tersebut dapat ditentukan dengan menggunakan rumus :

$$0 \pm t_{n-1}\left(\frac{\alpha}{2}\right) \times SE(r_k) \qquad (2.5)$$

Pada gambar 2.1 dapat dilihat bahwa autokorelasi tidak berbeda signifikan.



Gambar 2.1 Plot Autokorelasi Suku Bunga Bank

Gambar 2.1 memperlihatkan bahwa pada data suku bunga bank, autokorelasi pada l*ag* 1 sampai *lag* 4 berbeda signifikan dari nol karena *lag* 1 sampai *lag* 4 melalui selang kepercayaan. Hal ini menunjukan bahwa ada hubungan antar pengamatan.

3. Autokorelasi Parsial *(Partial Autocorrelation/*PACF*)*

Autokorelasi parsial merupakan korelasi antara Y_t dan Y_{t+k} dengan mengasumsikan hubungan antara Y_t dan $Y_{t+1}, Y_{t+2}, \dots, Y_{t+k-1}$ adalah konstan, sehingga diperoleh bentuk korelasi baru yang dinyatakan sebagai

$$corr(Y_t, Y_{t+k} | Y_{t+1}, Y_{t+2}, \dots, Y_{t+k-1})$$

Autokorelasi parsial antara Y_t dan Y_{t+k} akan sama dengan autokorelasi antara $(Y_t - \hat{Y}_t)$ dan $(Y_{t+k} - \hat{Y}_{t+k})$ sehingga (Wei, 2006:13)

$$\rho_k = \frac{E[(Y_t-\mu)(Y_{t+k}-\mu)]}{\sqrt{E[(Y_t-\mu)^2]E[(Y_{t+k}-\mu)]^2}} = \frac{Cov(Y_t,Y_{t+k})}{var(Y_t)} = \frac{\gamma_k}{\gamma_0} \qquad (2.6)$$

$$Y_{t+k} = \emptyset_{k1}Y_{t+k-1} + \emptyset_{k2}Y_{t+k-2} + \cdots + \emptyset_{kk}Y_t + e_{t+k}$$

$$E(Y_{t+k-j}, Y_{t+k}) = \emptyset_{k1}E(Y_{t+k-j}, Y_{t+k-1}) + \emptyset_{k2}E(Y_{t+\_-j}, Y_{t+k-2})$$

$$+ \cdots + \emptyset_{kk}E(Y_{t+k-j}, Y_t) + E(Y_{t+k-j}, e_{t+k})$$

$$\gamma_j = \emptyset_{k1}\gamma_{j-1} + \emptyset_{k2}\gamma_{j-2} + \cdots + \emptyset_{kk}\gamma_{j-k_e} \quad (2.7)$$

$E(Y_{t+k-j}, Y_{t+k-1}) = \gamma_{j-1}$, $E(Y_{t+k-j}, Y_{t+k}) = \gamma_{0,}$ dan $E(Y_{t+k-j}, e_{t+k}) = 0$.

Y_{t+k} merupakan proses stasioner dengan *mean* nol yang diregresikan dengan *k lag* variabel $Y_{t+k-1}, Y_{t+k-2}, \ldots, Y_t$ dimana $\emptyset_{ki}$ merupakan parameter regresi ke-i dan e_{t+k} menyatakan *error* yang tidak berkorelasi dengan Y_{t+k-j} $untuk\ j \geq 1$. Jika kedua ruas dibagi dengan γ_0 diperoleh

$$\rho_j = \emptyset_{k1}\rho_{j-1} + \emptyset_{k2}\rho_{j-2} + \cdots + \emptyset_{kk}\rho_{j-k} \text{ untuk } j = 1,2,3, \ldots, k \quad (2.8)$$

dimana $\rho_{-1} = \rho_1$ dan $\rho_{-(k-1)} = \rho_{k-1}$. Sehingga

$$\rho_1 = \emptyset_{k1}\rho_0 + \emptyset_{k2}\rho_1 + \cdots + \emptyset_{kk}\rho_{k-1}$$

$$\rho_2 = \emptyset_{k1}\rho_1 + \emptyset_{k2}\rho_0 + \cdots + \emptyset_{kk}\rho_{k-2}$$

$$\vdots$$

$$\rho_k = \emptyset_{k1}\rho_{k-1} + \emptyset_{k2}\rho_{k-2} + \cdots + \emptyset_{kk}\rho_0$$

$\rho_1 = \frac{\gamma_1}{\gamma_0}$ merupakan korelasi pertama sehingga diperoleh $\emptyset_{11} = \rho_1$ atau autokorelasi parsial pertama sama dengan autokorelasi pertama. Menurut aturan Cramer diperoleh (Wei, 2006:15)

$$\emptyset_{kk} = \frac{\begin{vmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{k-2} & \rho_1 \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{k-3} & \rho_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \rho_{k-1} & \rho_{k-2} & \rho_{k-3} & \cdots & \rho_1 & \rho_k \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{k-2} & \rho_{k-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{k-3} & \rho_{k-2} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \rho_{k-1} & \rho_{k-2} & \rho_{k-3} & \cdots & \rho_1 & 1 \end{vmatrix}}$$

$\emptyset_{kk}$ merupakan fungsi dari *k* yang disebut fungsi autokorelasi parsial. Pengujian signifikansi autokorelasi parsial digunakan untuk mengetahui apakah autokorelasi parsial berbeda signifikan dari nol. Hipotesis yang digunakan adalah:

$H_0: \emptyset_{kk} = 0$ (autokorelasi parsial pada *lag k* tidak bebeda signifikan dari nol)

$H_1: \emptyset_{kk} \neq 0$ (autokorelasi parsial pada *lag k* berbeda signifikan dari nol)

Statistik uji yang digunakan adalah:

$$t = \frac{\hat{\emptyset}_{kk}}{SE(\hat{\emptyset}_{kk})} \text{ dengan } df = n - 1 \qquad (2.9)$$

Standar *error* dari koefisien autokorelasi dirumuskan sebagai berikut (Wei, 2006:22):

$$SE(\hat{\emptyset}_{kk}) = \sqrt{\frac{1}{n}} \qquad (2.10)$$

dengan

$SE(\hat{\emptyset}_{kk})$: standar *error* autokorelasi parsial pada *lag k*

$\hat{\emptyset}_{kk}$: autokorelasi parsial sampel pada *lag k*

n : banyak pengamatan

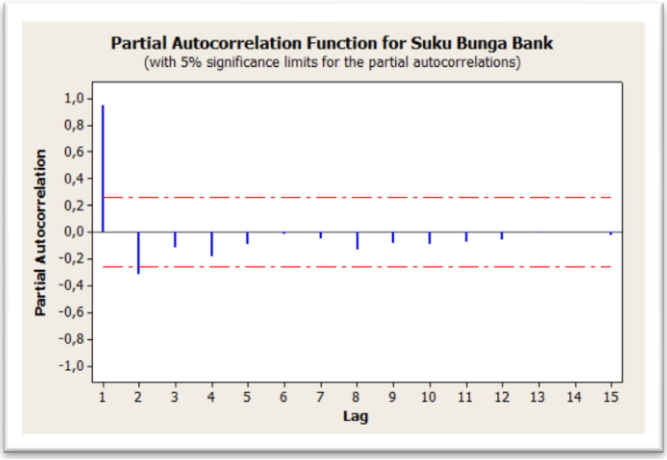
Jika nilai $t_{hitung} > t_{n-1}\left(\frac{\alpha}{2}\right) atau\, t_{hitung} < -t_{n-1}\left(\frac{\alpha}{2}\right)$ maka H_0 ditolak. Artinya autokorelasi parsial dikatakan berbeda secara signifikan dari nol. Sedangkan jika nilai t_{hitung} memenuhi $-t_{n-1}\left(\frac{\alpha}{2}\right) < t_{hitung} < t_{n-1}\left(\frac{\alpha}{2}\right)$ maka autokorelasi parsial dikatakan tidak berbeda secara signifikan dari nol.

Signifikansi autokorelasi dapat ditentukan dengan melihat *correlogram* dengan fungsi autokorelasi parsial. *Correlogram* adalah plot antara *lag k* dengan $\hat{\emptyset}_{kk}$, dimana $\hat{\emptyset}_{kk} = 0$ adalah pusat selang kepercayaan, sedangkan garis putus-

putus merupakan batas atas dan bawah dari selang kepercayaan Selang kepercayaan dapat ditentukan menggunakan rumus :

$$0 \pm t_{n-1}\left(\frac{\alpha}{2}\right) \times SE(\widehat{\emptyset}_{kk}) \qquad (2.11)$$

Berikut pada Gambar 2.2 dapat dilihat bahwa autokorelasi parsial berbeda signifikan.



Gambar 2.2 Plot Autokorelasi Parsial Data Suku Bunga Bank

Gambar 2.2 memperlihatkan bahwa pada data suku bunga bank, autokorelasi parsial pada *lag* ke 1 berbeda signifikan dari nol karena garis biru melewati batas selang kepercayaan. Hal ini menunjukkan bahwa ada hubungan antar pengamatan.

4. White Noise

Sebuah proses $\{Y_t\}$ disebut *white noise* jika merupakan serangkaian variabel acak yang tidak berkorelasi dan berdistribusi tertentu dengan rata-rata tetap $E(Y_t)$ biasanya bernilai 0, variansi konstanta $Var(Y_t) = \sigma^2$ dan $Cov(Y_t, Y_{t+k}) = 0$ untuk semua $k \neq 0$ (Wei, 2006: 16). Proses *white noise* dari suatu *time series* $\{Y_t\}$ adalah stasioner dengan fungsi autokorelasi:

$$\rho_k = \begin{cases} 1 & ,\mathrm{k} = 0 \\ 0 & ,k \neq 0 \end{cases}$$

fungsi autokorelasi parsial,

$$\emptyset_{kk} = \begin{cases} 1 & ,\mathrm{k} = 0 \\ 0 & ,k \neq 0 \end{cases}$$

fungsi autokovarians,

$$\gamma_k = \begin{cases} {\sigma_\alpha}^2 & ,\mathrm{k} = 0 \\ 0 & ,k \neq 0 \end{cases}$$

Suatu proses *white noise* dapat diperoleh dengan melihat plot *ACF* dan *PACF* dengan nilai autokorelasinya tidak melebihi garis signifikansi. Pada proses *white noise,* autokorelasi dan autokorelasi parsial tidak berbeda signifikan dari 0.

C. Logika *Fuzzy*

Logika *Fuzzy* adalah suatu cara yang tepat untuk memetakan suatu ruang *input* ke dalam suatu ruang *output*. Logika *fuzzy* menjadi alternatif dari berbagai sistem yang ada dalam pengambilan keputusan karena logika *fuzzy* mempunyai kelebihan diantaranya konsep logika yang mudah untuk dimengerti, memiliki toleransi terhadap data yang tidak tepat, dan mampu memodelkan fungsi non-linear yang kompleks (Kusumadewi, 2013 : 2).

Perbedaan mendasar dari logika *crisp* dan logika *fuzzy* adalah keanggotaan elemen dalam suatu himpunan. Dalam logika *crisp* suatu elemen mempunyai dua pilihan yaitu bernilai 1 atau 0, sedangkan keanggotaan elemen pada logika *fuzzy* berada di selang [0,1] (Kusumadewi, 2013: 3).

Ada beberapa konsep dasar dalam logika *fuzzy* yaitu himpunan *fuzzy*, fungsi keanggotaan, operator *fuzzy, fuzzifikasi*, dan *defuzzifikasi*.

1. Himpunan *Fuzzy*

Himpunan *universal* (semesta pembicaraan) adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel. Misalkan *S* adalah himpunan *universal* dengan $x \in S$. Suatu himpunan *fuzzy A* dalam *S* didefinisikan sebagai suatu fungsi keanggotaan $\mu_A(x)$ yang memetakan setiap objek di *S* menjadi suatu nilai real dalam interval [0,1]. (Wang, 1997)

Himpunan *fuzzy* merupakan perluasan dari himpunan klasik. Pada himpunan klasik, fungsi keanggotaannya hanya bernilai 0 atau 1. Sedangkan fungsi keanggotaan pada himpunan *fuzzy* ialah fungsi kontinu dalam interval [0,1]. Suatu himpunan *fuzzy A* dalam *S* dapat dinotasikan sebagai pasangan berurutan dari nilai elemen *x* dengan derajat keanggotaannya $\mu_A(x)$. (Wang, 1997)

$$A = \{(x, \mu_A(x)) | x \in S\}$$

Domain himpunan *fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *fuzzy*. Nilai domain dapat berupa bilangan positif maupun negatif.

2. Fungsi Keanggotaan

Fungsi keanggotaan (*membership function*) merupakan *fungsi* yang memetakan nilai variabel ruang input ke dalam nilai keanggotaan yang berada pada *range* [0 1] (Jang dkk, 1997). Fungsi keanggotaan yang dapat dibangun dan digunakan untuk mempresentasikan himpunan *fuzzy* antara lain (Kusumadewi, 2010:20-39).

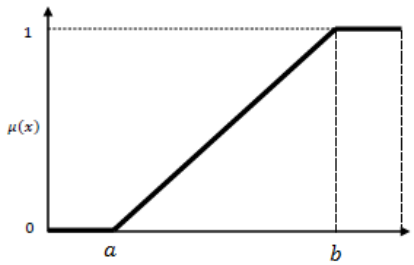
a. Representasi Linear

Pada representasi linear, pemetaan *input* ke derajat angggotanya digambarkan sebagai suatu garis lurus sehingga merupakan bentuk yang paling sederhana. Terdapat 2 keadaan pada himpunan *fuzzy* yang linear, yaitu representasi linear naik dan representasi linear turun.

Representasi linear naik dimulai dari domain yang memiliki derajat keanggotaan nol dan bergerak ke kanan menuju nilai domain yang memiliki derajat keanggotaan satu. Representasi linear naik memiliki fungsi keanggotaan yaitu:

$$\mu(x) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a < x \leq b \\ 1 & x > b \end{cases} \tag{2.12}$$

dengan grafik representasinya seperti pada Gambar 2.3.

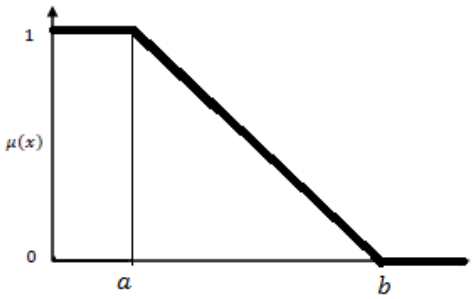


Gambar 2.3 Grafik Representasi Linear Naik

Representasi linear yang kedua yaitu representasi linear turun. Representasi linear turun memiliki fungsi keanggotaan yaitu:

$$\mu(x) = \begin{cases} 0 & x > b \\ \frac{b-x}{b-a} & a < x \leq b \\ 1 & x \leq a \end{cases} \tag{2.13}$$

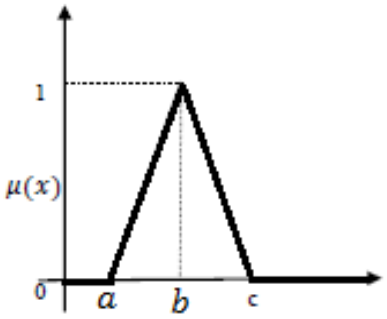
dengan grafik representasi kurva linear turun seperti pada Gambar 2.4.



Gambar 2.4 Grafik Representasi Linear Turun

b. Representasi Kurva Segitiga

Representasi kurva segitiga pada dasarnya terbentuk dari gabungan 2 garis linear, yaitu linear naik dan linear turun. Kurva segitiga hanya memiliki satu nilai x dengan derajat keanggotaan tertinggi, yaitu pada saat x=b. Nilai yang tersebar di persekitaran b memiliki perubahan derajat keanggotaan menurun yang menjauhi 1, seperti pada Gambar 2.5.



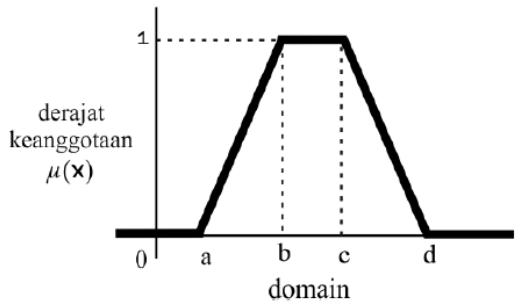
Gambar 2.5 Grafik Representasi Kurva Segitiga

dengan fungsi keanggotaan

$$\mu(x) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a < x \leq b \\ \frac{c-x}{c-b} & b < x \leq c \\ 1 & x \geq c \end{cases} \quad (2.14)$$

c. Representasi Kurva Trapesium

Representasi kurva trapesium merupakan perluasan dari kurva segitiga, yang memiliki lebih dari satu titik yang nilai keanggotaanya 1, seperti pada Gambar 2.6.



Gambar 2.6 Grafik Representasi Kurva Trapesium

dengan fungsi keanggotaan

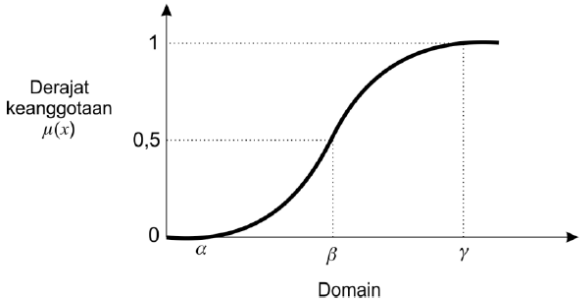
$$\mu(x) = \begin{cases} 0 & x \leq a \; atau \; x \geq b \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \frac{d-x}{d-c} & c \leq x \leq d \end{cases} \tag{2.15}$$

d. Representasi Kurva-*S*

Kurva-*S* atau *sigmoid* terdiri dari kurva pertumbuhan dan penyusutan yang merupakan kurva berbentuk huruf dan digunakan menghubungkan kenaikan dan penurunan permukaan yang tidak linear. Definisi Kurva-*S* menggunakan 3 parameter , yaitu nilai keanggotaan nol (α), nilai keanggotaan satu (γ), dan titik infleksi (β) yaitu titik dengan domain yang memiliki derajat keanggotaan sebesar 0,5.

1) Kurva-*S* untuk pertumbuhan

Kurva-*S* untuk pertumbuhan bergerak dari sisi kiri dengan nilai keanggotaan 0 ke sisi kanan dengan nilai keanggotaan 1, seperti pada Gambar 2.7.



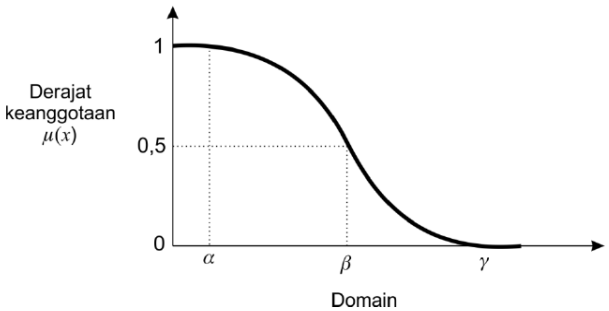
Gambar 2.7 Grafik Representasi Kurva-S Pertumbuhan

dengan fungsi keanggotaan

$$\mu(x) = \begin{cases} 0 & x \leq \alpha \\ 2(\frac{x-\alpha}{\gamma-\alpha})^{2} & \alpha \leq x \leq \beta \\ 1 - 2(\frac{\gamma-x}{\gamma-\alpha})^{2} & \beta \leq x \leq \gamma \\ 1 & x \geq \gamma \end{cases} \quad (2.16)$$

2) Kurva-*S* untuk Penyusutan

Kurva–*S* untuk penyusutan bergerak dari sisi kanan dengan nilai keanggotaan 1 ke sisi kiri dengan nilai keanggotaan 0, seperti pada Gambar 2.8.



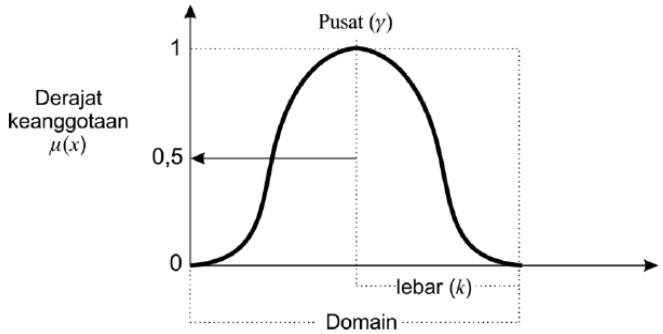
Gambar 2.8 Grafik Representasi Kurva-S untuk penyusutan

dengan fungsi keanggotaan

$$\mu(x) = \begin{cases} 1 & x \leq \alpha \\ 1 - 2(\frac{x-\alpha}{\gamma-\alpha})^2 & \alpha \leq x \leq \beta \\ 2(\frac{\gamma-x}{\gamma-\alpha})^2 & \beta \leq x \leq \gamma \\ 0 & x \geq \gamma \end{cases} \quad (2.17)$$

3) Representasi Kurva Gauss

Kurva gauss menggunakan parameter (γ) untuk menunjukkan nilai domain pada pusat kurva dan (*k*) sebagai lebar kurva seperti yang ditunjukkan pada Gambar 2.9.



Gambar 2.9 Grafik Representasi Kurva Gauss

dengan fungsi keanggotaan

$$\mu(x) = e^{\frac{-(x-y)^2}{2k^2}} \quad (2.18)$$

3. Fuzzifikasi

Fuzzifikasi merupakan suatu proses untuk mengubah *input* yang bernilai *crisp* menjadi derajat keanggotaan yang bernilai *fuzzy*. Pada tahap ini diperoleh nilai derajat keanggotaan masing-masing data pada himpunan *fuzzy* dengan menggunakan fungsi keanggotaan masing-masing himpunan *fuzzy* tersebut (Wang, 1997:7).

4. Defuzzifikasi

Defuzzifikasi merupakan proses akhir untuk mengubah *output* yang bernilai *fuzzy* menjadi suatu nilai *crisp.* Hasil *output* berupa bilangan pada domain himpunan *fuzzy* tertentu, sehingga harus diambil suatu nilai *crisp* tertentu sebagai *output.*

Terdapat beberapa metode defuzzifikasi salah satunya adalah aturan Mamdani (Kusumadewi & Purnomo, 2010).

1) Metode *Mean of Maximum*

Solusi tegas diperoleh dengan cara mengambil nilai rata-rata domain yang memiliki nilai keanggotaan maksimum.

2) Metode *Largest of Maximum*

Solusi tegas diperoleh dengan cara mengambil nilai maksimum domain yang memiliki nilai keanggotaan maksimum.

3) Metode *Smallest of Maximum*

Solusi tegas diperoleh dengan cara mengambil nilai minimum domain yang memiliki nilai keanggotaan maksimum.

D. Neural Network

Dalam memprediksi harga minyak mentah di Indonesia, digunakan neural network untuk melakukan penyelesaian proses peramalan. *Neural network* adalah model yang terinspirasi oleh sistem saraf biologis dan terdiri dari unsur-unsur sederhana yang beroperasi secara paralel (Graupe, 2007). Jaringan pada *neural network* dilatih untuk melakukan fungsi tertentu dengan menyesuaikan nilai-nilai dari koneksi (bobot) antar unsur-unsur. Biasanya, *neural network* disesuaikan,

atau dilatih, sehingga input tertentu menyebabkan output target tertentu (Baele et al, 2010:2). Struktur jaringan *neural network* terdiri dari tiga lapisan sebagai berikut (Siang, 2005: 24) .

1. Lapisan input, bertugas menerima pola *input* dari luar kemudian melewatkan nya ke lapisan tersembunyi.
2. Lapisan tersembunyi, terletak di antara lapis *input* dan lapis *output*, yang dapat terdiri atas beberapa lapis tersembunyi. *Output* dari lapisan tersembunyi tidak dapat diamati secara langsung.
3. Lapisan *output*, merupakan solusi dari pemrosesan selama jaringan dibentuk.

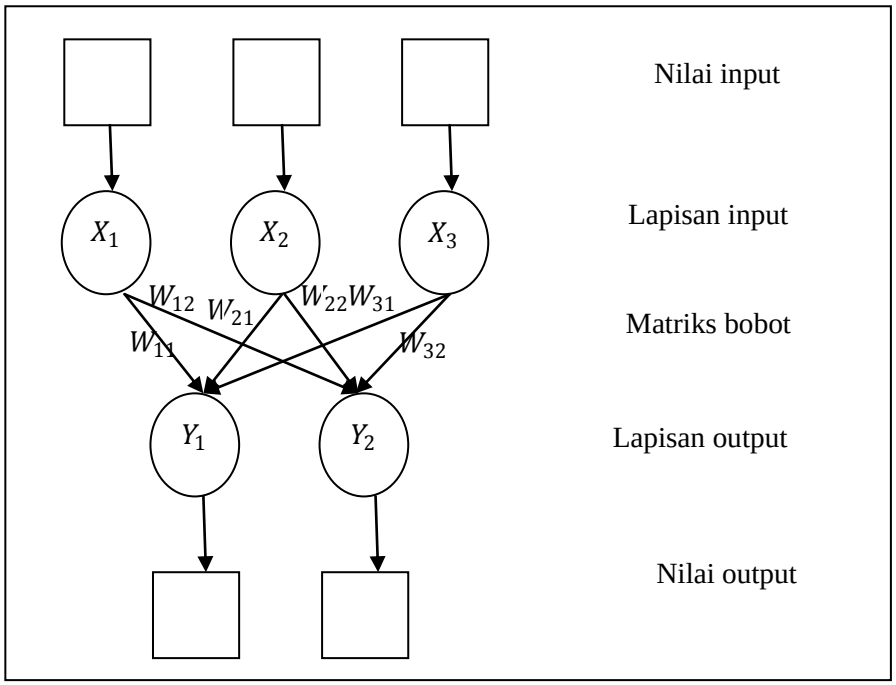
Model NN ditentukan oleh tiga hal yaitu, arsitektur jaringan, algoritma pembelajaran, dan fungsi aktivasi (Fausett, 1994:3). Arsitektur jaringan merupakan pola hubungan yang terjalin antar *neuron*. Algoritma pembelajaran merupakan metode untuk menentukan bobot-bobot.

1. Arsitektur Jaringan

Arsitektur jaringan merupakan salah satu indikator dalam penentuan model NN. Beberapa arsitektur jaringan yang sering dipakai dalam NN antara lain (Siang, 2005: 24):

a. Jaringan Lapisan Tunggal (*Single Layer*)

Jaringan ini merupakan sekumpulan neuron *input* yang dihubungkan langsung dengan sekumpulan *output*nya. Arsitektur jaringan ini ditunjukkan pada Gambar 2.10 Beberapa neuron pada lapisan *input* dan lapisan *output* saling terhubung dan memiliki bobot masing-masing (Kriesel, 2005: 74).

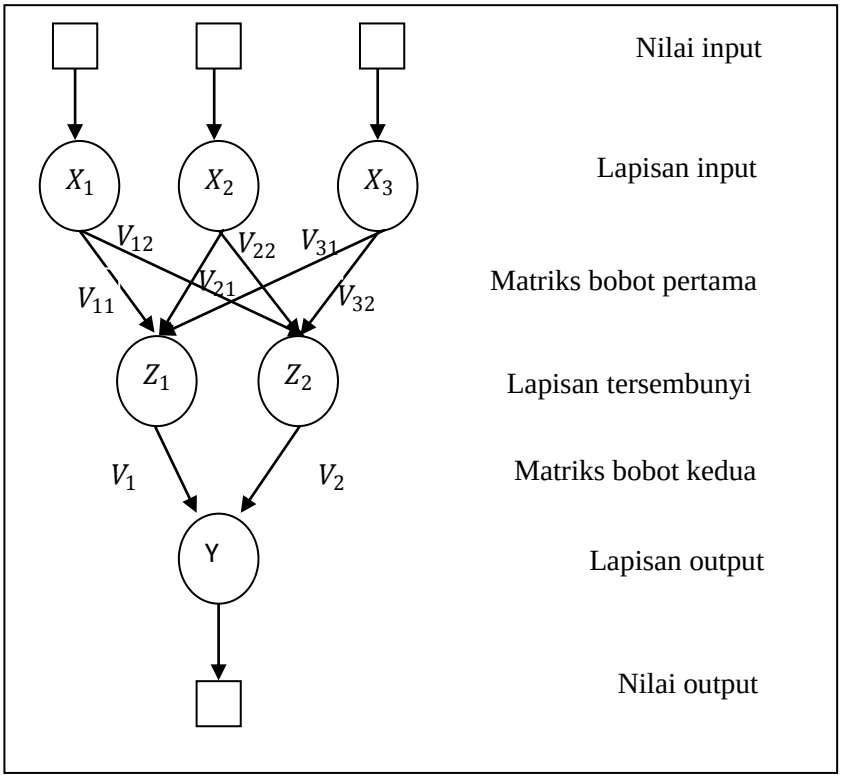


Gambar 2.10 Arsitektur Jaringan Lapisan Tunggal (Single Layer)

Pada gambar tersebut lapisan input memiliki 3 neuron yaitu X_1, X_2 dan X_3 sedangkan pada lapisan output memiliki 2 neuron yaitu Y_1 dan Y_2. Setiap neuron pada lapisan input dan lapisan output saling terhubung oleh bobot yang bersesuaian.

b. Jaringan Lapisan Banyak (*Multi Layer*)

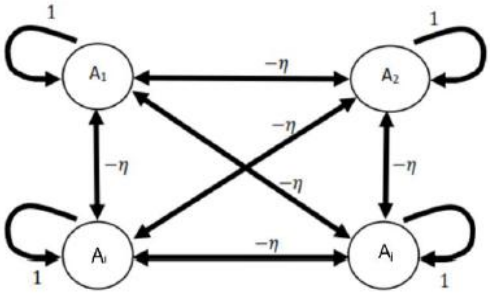
Jaringan lapisan banyak memiliki satu atau lebih lapisan tersembunyi (*hidden layer*) (Fausset, 1994:14), yaitu lapisan yang terletak diantara lapisan *input* dan lapisan *output.* Jaringan ini dapat menyelesaikan permasalahan yang lebih sulit dari lapisan *single layer*. Model jaringan lapisan banyak dapat dilihat pada Gambar 2.11.



Gambar 2.11 Arsitektur Jaringan Lapisan Banyak (Multi Layer)

c. Neural Network dengan Lapisan Kompetititf (Competitive Layer Net)

Neural Network dengan Lapisan Kompetititf memiliki bentuk yang berbeda dari lapisan tunggal dan lapisan multilayer. Antar neuron saling terhubung pada lapisan kompetitif di jaringan ini. Sinyal yang melalui lapisan neuron bergerak dalam dua arah dengan menghasilkan perulangan (loop) pada jaringan secara terus menerus sampai mencapai titik keseimbangan.Gambar 2.12 merupakan salah satu contoh arsitektur *neural network* dengan lapisan kompetititf.



Gambar 2.12 Arsitektur Jaringan Lapisan Kompetitif

2. Fungsi Aktivasi

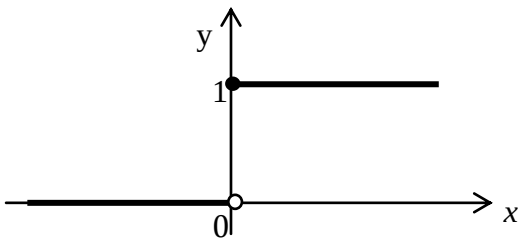
Fungsi aktivasi digunakan untuk mengaktifkan setiap neuron yang ada pada jaringan. Fungsi aktivasi akan menentukan *output* suatu unit (mengubah sinyal *input* menjadi sinyal *output*) yang akan dikirim ke unit lain. Ada beberapa fungsi aktivasi yang sering digunakan dalam jaringan saraf tiruan (Fausett, 1994: 17-19), antara lain sebagai berikut.

a. Fungsi Undak Biner (*Hard Limit*)

Jaringan dengan lapisan tunggal sering menggunakan fungsi undak (*step function*) untuk mengkonversikan *input* dari suatu variabel yang bernilai kontinu ke suatu *output* biner (0 atau 1). Fungsi undak biner (*hard limit*) dengan rumus sebagai berikut:

$$y = f(x) = \begin{cases} 0, & jika\ x < 0 \\ 1, & jika\ x \geq 0 \end{cases} \quad (2.19)$$

Grafik dari fungsi aktivasi undak biner seperti pada Gambar 2.4.4.



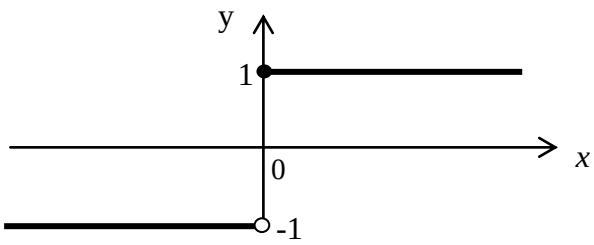
Gambar 2.13 Fungsi Aktivasi Undak Biner (Hard Limit)

Pada MatlabR2013a, sintaks untuk fungsi aktivasi ini adalah *hardlim.*

b. Fungsi Bipolar (*Symetric Hard Limit*)

Fungsi bipolar mirip dengan fungsi undak biner, perbedaannya terletak pada nilai *output* yang dihasilkan. Nilai *output* bipolar berupa 1 dan -1 seperti pada Gambar 2.14. Fungsi bipolar dirumuskan sebagai berikut:

$$y = f(x) = \begin{cases} -1, & jika\ x < 0 \\ 1, & jika\ x \geq 0 \end{cases} \tag{2.20}$$



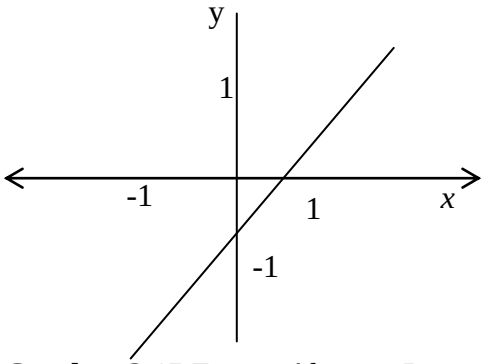
Gambar 2.14 Fungsi Aktivasi Bipolar (Symetric Hard Limit)

Pada MatlabR2013a, sintaks untuk fungsi aktivasi ini adalah *hardlims.*

c. Fungsi Linier

Fungsi linier merupakan suatu fungsi dengan grafiknya berupa garis lurus. Grafik fungsi linier ditunjukkan pada Gambar 2.15. Bentuk umum dari fungsi linier yaitu

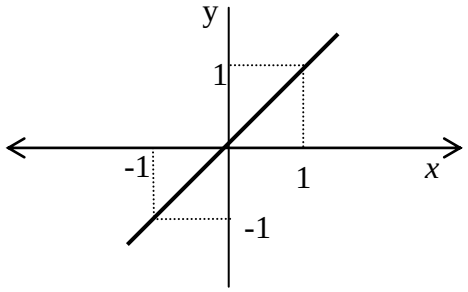
$$y = f(x) = ax + b,\ \ x \in R \tag{2.21}$$



Gambar 2.15 Fungsi Aktivasi Linier

Salah satu bentuk khusus dari fungsi linier adalah fungsi identitas. Fungsi identitas memiliki nilai *output* yang sama dengan nilai *input*nya (Gambar 2.16). Fungsi identitas sering dipakai apabila menginginkan *output* berupa sembarang bilangan riil. Fungsi identitas dirumuskan sebagai berikut:

$$y = f(x) = x, \;\; x \in R \tag{2.22}$$



Gambar 2.16 Fungsi Aktivasi Identitas

Pada MatlabR2013a, sintaks untuk fungsi aktivasi ini adalah *purelin*.

d. Fungsi *Sigmoid Biner*

Fungsi ini digunakan untuk jaringan syaraf yang menggunakan algoritma pembelajaran *backpropagation*. Fungsi *sigmoid biner* memiliki nilai pada interval 0 sampai 1. Oleh karena itu fungsi ini sering digunakan untuk jaringan syaraf yang membutuhkan nilai *output* yang terletak pada interval 0 sampai 1. Namun, fungsi ini juga bisa digunakan oleh jaringan syaraf yang nilai *output* nya 0 atau 1. Fungsi *sigmoid biner* dirumuskan sebagai:

$$y = f(x) = \frac{1}{1+e^{-x}} \tag{2.23}$$

Pada MatlabR2013a, sintaks untuk fungsi aktivasi ini adalah *logsig*.

e. Fungsi *Sigmoid Bipolar*

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

$$y = f(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad (2.24)$$

Pada MatlabR2010a, sintaks untuk fungsi aktivasi ini adalah *tansig.*

3. Metode Pembelajaran *(Learning Method)*

Selain arsitektur, metode pengaturan nilai bobot (*training*) merupakan karakteristik yang penting dalam jaringan *neural network* karena semakin banyak *training* yang dilakukan maka semakin kecil *error* yang dihasilkan di *ouput* layer-nya, dengan demikian semakin kecil juga *error* suatu sistem. Metode pembelajaran dalam *neural network* dibagi menjadi 2 jenis, yaitu metode pembelajaran terawasi (*supervised learning*) dan metode pembelajaran tak terawasi (*unsupervised learning*) (Fausset, 1994:12-15).

a Pembelajaran terawasi *(Supervised learning)*

Pada metode ini, setiap nilai yang diberikan ke dalam jaringan sudah diketahui nilai *output*nya. Selisih antara nilai aktual dengan nilai yang dikehendaki disebut *error*. Nilai *error* digunakan untuk mengoreksi bobot dari jaringan agar jaringan tersebut mampu menghasilkan *output* sesuai dengan data target yang diketahui. Contoh algoritma jaringan yang menggunakan pembelajaran ini yaitu Hebbian, Perceptron, Backpropagation.

b Pembelajaran tak terawasi *(Unsupervised learning)*

Berbeda dengan *supervised learning*, metode ini tidak memerlukan nilai target *output*. Selama proses pembelajaran, nilai bobot disusun dalam suatu range tertentu bergantung dengan nilai *input* yang diberikan. Tujuan dari pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu *range* tertentu, dan sering digunakan untuk mengklasifikasikan suatu pola. Contoh algoritma jaringan yang menggunakan pembelajaran ini adalah Kohenen, LVQ (*Learning Vector Quantization*), Competitive.

E. Algoritma *Backpropagation* pada Jaringan Elman

Algoritma pembelajaran yang digunakan dalam tugas akhir ini adalah algoritma pembelajaran *backpropagation* yang termasuk algoritma terawasi (*supervised learning*). Pembelajaran *backpropagation* terdiri dari 3 fase yaitu perambatan maju (*feedforward*), perambatan mundur (*backpropagation*) dan perubahan bobot (Fausset, 1994: 294-295). Fase pertama adalah fase perambatan maju, pola masukan dihitung maju dari lapisan *input* sampai lapisan *output* menggunakan suatu fungsi aktivasi. Fase selanjutnya adalah fase perambatan mundur, menggunakan *error output* untuk mengubah nilai bobot-bobotnya. Fase terakhir yaitu fase perubahan bobot yang dimaksudkan untuk menurunkan *error* yang terjadi dengan cara memodifikasi bobot. Jaringan yang digunakan dalam tugas akhir ini adalah RNN jaringan Elman maka fase-fase disesuaikan dengan adanya umpan balik dari *output* lapisan tersembunyi ke lapisan *input*.

Langkah awal dalam proses algoritma *backpropagation* yaitu inisialisasi bobot dengan bilangan acak kecil serta ditetapkan maksimum *epoch*, target *error*, dan *learning rate*. Selama nilai *epoch* kurang dari maksimum *epoch* (*epoch* <

maksimum *epoch*) dan MSE lebih besar dari target *error* (MSE > target *error*) dengan target *error* 0,01 maka dilakukan proses *feedforward* dan *backpropagation.*

Langkah pertama dalam proses f*eedforward* yaitu tiap unit *input* ($x_i = 1,2,\dots n$) menerima sinyal dan meneruskannya ke unit selanjutnya, yaitu lapisan tersembunyi. Setelah itu dihitung semua sinyal-sinyal *input* terbobot pada lapisan tersembunyi sebagai berikut:

$$U\_net_k = b_k + \sum_{i=1}^{p} x_i w_{ik} \tag{2.25}$$

dengan

b_k = bobot bias pada neuron ke-*k* pada lapisan tersembunyi dengan $k = 1,2,\dots,m$

x_i = variabel input, $i = 1,2,\dots,n$

w_{ik} = bobot dari lapisan *input* ke–*i* menuju neuron ke-*k* pada lapisan tersembunyi.

Sinyal *output* U_net_k dihitung menggunakan fungsi aktivasi sebagai berikut:

$$U_k = f(U\_net_k) \tag{2.26}$$

Langkah selanjutnya tiap-tiap neuron tambahan ($U_k, k = 1,2,\dots,m$) menerima sinyal dan meneruskan sinyal ke lapisan tersembunyi. Neuron tambahan kemudian mengirimkan kembali sinyal ke neuron lapisan tersembunyi sebagai berikut:

$$s\_net_k = \sum_{i=1}^{q} x_i w_{jk} \tag{2.27}$$

dengan

U_k = sinyal output (telah diaktivasi) yang diterima lapisan tersembunyi dari neuron input $k = 1,2, \dots, m$

w_{jk} = bobot dari neuron tambahan ke-j menuju neuron ke-k pada lapisan tersembunyi, $j = 1,2, \dots, m$

Selanjutnya tiap-tiap neuron pada lapisan tersembunyi ($T_k, k = 1,2, \dots m$) menjumlahkan sinyal-sinyal *input* terbobot yang diperoleh dari lapisan *input* dan neuron tambahan sebagai berikut:

$$T\_net_k = U\_net_k + s\_net_k = b_k + \sum_{i=1}^{p} x_i w_{ik} + \sum_{j=1}^{m} x_i w_{jk} \quad (2.28)$$

dengan

U_net_k = sinyal yang diterima lapisan tersembunyi dari neuron lapisan *input*

s_net_k = sinyal yang diterima lapisan tersembunyi dari neuron lapisan tambahan

Sinyal *output* T_net_k dihitung menggunakan fungsi aktivasi sebagai berikut:

$$T_k = f(T\_net_k) \quad (2.29)$$

Langkah berikutnya sinyal T_k di kirim pada lapisan *output*. Fungsi aktivasi yang digunakan dari lapisan input ke lapisan tersembunyi dalam tugas akhir ini adalah *tansig*, sehingga persamaan (2.5.2) dan Persamaan (2.5.5) menjadi:

$$U_k = \frac{1-e^{-(U_{net\ k})}}{1+e^{-(U_{net\ k})}} \quad (2.30)$$

$$\text{dan} \quad T_k = \frac{1-e^{-(T_{net_k})}}{1+e^{-(T_{net_k})}} \quad (2.31)$$

selanjutnya, tiap neuron *output* (Y_t) menjumlahkan sinyal-sinyal *input* terbobot dari lapisan tersembunyi menggunakan rumus sebagai berikut:

$$Y = b_o + \sum_{i=1}^{p} v_k T_k \quad (2.32)$$

dengan

b_o = bias pada neuron *output,*

v_k = bobot dari neuron ke-*k* pada lapisan tersembunyi yang menuju lapisan *output* dengan *k* =1,2,.., m,

T_k = sinyal *output* yang telah diaktivasi dari lapisan tersembunyi dengan *k* =1,2,.., m.

Sinyal *output y* dihitung menggunakan fungsi aktivasi sebagai berikut:

$$Y_t = f(y) \tag{2.33}$$

Setelah langkah dalam proses *feedforward*, kemudian dilakukan langkah *backpropagation*. Langkah awal dalam proses *backpropagation* yaitu menghitung faktor δ unit *output* berdasarkan *error* di setiap unit output $(Y_t, t = 1,2, \dots, r)$ yang didefinisikan sebagai berikut

$$\delta_k = (t_k - Y_t) f'(y) \tag{2.34}$$

Kemudian dihitung koreksi bobot yang nantinya akan digunakan untuk memperbaiki nilai v_k dan koreksi bias yang nantinya akan digunakan untuk memperbaiki nilai $b_0$dengan rumus berikut :

$$\Delta v_k = \alpha \delta_k Y_t \tag{2.35}$$

$$\Delta b_0 = \alpha \delta_k \tag{2.36}$$

Dimana α adalah laju percepatan

Kemudian δ_k dikirimkan ke neuron-neuron pada lapisan sebelumnya (lapisan tersembunyi). Langkah ini dilakukan sebanyak jumlah neuron pada lapisan tersembunyi.

Selanjutnya tiap-tiap *neuron* tersembunyi $T_k, k = 1,2, \dots, m$ menjumlahkan delta *input*nya dari neuron-neuron yang berada pada lapisan di atasnya sebagai berikut:

$$\delta\_net_k = \sum_{k=1}^{m} \delta_k v_k \quad (2.37)$$

Untuk menghitung informasi *error*, kalikan nilai δ_net_k dengan turunan dari fungsi aktivasinya:

$$\delta_k = \delta_{net_k} f'(U\_net_k) \quad (2.38)$$

Kemudian dihitung koreksi bobot yang nantinya akan digunakan untuk memperbaiki nilai w_{ik} dan w_{jk} dengan rumus sebagai berikut :

$$\Delta w_{ik} = \alpha \delta_k x_i \quad (2.39)$$

$$\Delta w_{jk} = \alpha \delta_k U_k \quad (2.40)$$

Selain itu, menghitung koreksi bias yang nantinya akan digunakan untuk memperbaiki nilai b_k:

$$\Delta b_k = \alpha \delta_k \quad (2.41)$$

Langkah berikutnya tiap-tiap neuron *output* Y_k memperbaiki bias dan bobotnya sebagai berikut :

$$v_k(baru) = v_k(lama) + \Delta v_k \quad (2.42)$$

$$b_0(baru) = b_0(lama) + \Delta b_0 \quad (2.43)$$

Langkah selanjutnya setiap neuron lapisan tersembunyi $T_k, k = 1,2, \dots, m$ memperbaiki bias dan bobotnya sebagai berikut :

$$w_{ik}(baru) = w_{ik}(lama) + \Delta w_{ik} \quad (2.44)$$

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.45)$$

$$b_k(baru) = b_k(lama) + \Delta b_k \quad (2.46)$$

Setelah proses *feedforward* dan *backpropagation* memenuhi kondisi yang ditentukan yaitu *epoch* < maksimum *epoch* dan MSE > target *error*, maka pembelajaran dengan algoritma *backpropagation* berhenti.

Setelah pembelajaran algoritma *backpropagation* berhenti, maka algoritma *backpropagation* tersebut dapat digunakan sebagai pengujian data *testing*. Namun, dalam penentuan *output* jaringan dalam jaringan ini hanya terdiri dari proses *feedforward* yaitu pada langkah (2.25) sampai dengan (2.32). Sebelum melakukan pembelajaran, maka terlebih dahulu mengatur parameter-parameter yang akan digunakan dalam proses pembelajaran.

Penetapan parameter pembelajaran untuk algoritma *backpropagation* pada *Elman* RNN adalah sebagai berikut.

1. Menentukan nilai dari maksimum *epoch*

Maksimum epoch adalah jumlah maksimum *epoch* yang boleh dilakukan selama proses pembelajaran. Iterasi akan dihentikan apabila nilai *epoch* melebihi maksimum *epoch*. Nilai default untuk maksimum *epoch* adalah 10. Perintah pada Matlab R2013a adalah **`net.trainParam.epochs = MaxEpoch`**.

2. Kinerja Tujuan (Target *Error*)

Target *error* adalah target nilai fungsi kinerja. Iterasi akan dihentikan apabila nilai fungsi kinerja kurang atau sama dengan kinerja tujuan (target *error*). Nilai default untuk kinerja tujuan adalah 0. Perintah pada Matlab R2013a adalah **`net.trainParam.goal = TargetError.`**

3. Learning Rate

Learning rate adalah laju pembelajaran. Semakin besar nilai learning rate maka semakin cepat pula langkah pembelajaran. Semakin kecil *learning rate*, maka proses pembelajaran akan sangat lama. Sehingga perlu pemilihan nilai yang tepat untuk *learning rate*. Nilai default learning rate adalah 0,01. Perintah pada Matlab R2013a adalah **`net.trainParam.lr = LearningRate.`**

4. Rasio untung menaikan learning rate

Rasio ini berguna sebagai faktor pengali untuk menaikan *learning rate* apabila *learning rate* terlalu rendah untuk mencapai kekonvergenan. Nilai default untuk rasio kenaikan *learning rate* adalah 1,05. Perintah pada Matlab R2013a adalah **`net.trainParam.lr.inc=IncLearningRate`**.

5. Rasio untuk menurunkan *learning rate*

Rasio ini berguna sebagai faktor pengali untuk menurunkan *learning rate* apabila *learning rate* terlalu tinggi untuk mencapai kekonvergenan. Nilai default untuk rasio penurunan *learning rate* adalah 0,7. perintah pada Matlab R2013a adalah **`net.trainParam.lr.dec=DecLearningRate.`**

6. Jumlah epoch yang akan digunakan kemajuannya

Parameter ini menunjukkan berapa jumlah *epoch* yang berselang yang akan ditunjukkan kemajuannya. Nilai *default* untuk jumlah *epoch* yang akan ditunjukan adalah 25. Perintah pada Matlab R2013a adalah **`net.trainParam.show=EpochShow.`**

7. Maksimum kenaikan kinerja

Maksimum kenaikan kinerja adalah nilai maksimum kenaikan *error* yang diijinkan, antara *error* saat ini dan *error* sebelumnya. Nilai default untuk maksimum kenaikan kinerja adalah 1,04. Perintah pada Matlab R2013a adalah **`net.trainParam.max_perf_inc = MaxPerfInc`.**

8. Momentum

Momentum adalah perubahan bobot yang didasarkan atas arah gradien pola akhir dan pola sebelumnya. Besarnya momentum antara 0 sampai 1. Apabila momentum = 0, maka perubahan bobot hanya akan dipengaruhi oleh gradiennya. Tetapi, apabila nilai momentum = 1, maka perubahan bobot akan sama dengan perubahan bobot sebelumnya. Perintah pada Matlab R2013a adalah **`net.trainParam.mc = Momentum`.**

E. Ukuran Akurasi Ketepatan Model Peramalan

Terdapat banyak macam kriteria akurasi ketepatan model dari sebuah peramalan. Dalam tugas akhir ini digunakan MAPE dan MSE sebagai kriteria kebaikan model. MAPE dan MSE adalah (Hanke dan Wichern, 2005:79)

1. *Mean Absolut Percentage Error* (MAPE)

MAPE merupakan salah satu ukuran ketepatan peramalan. MAPE dapat dihitung dengan rumus

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|e_t|}{Y_t}\ x\ 100\% \qquad (2.47)$$

dengan

MAPE : *Mean absolutpercentage error*

e_t : *eror* peramalan pada waktu ke-*t*

Y_t : nilai aktual data ke-*t*

n : banyaknya pengamatan

2. *Mean Square Error* (MSE)

MSE adalah suatu ketepatan peramalan dengan mengkuadratkan masing-masing *error* untuk masing-masing pengamatan dalam sebuah susunan data dan kemudian memperoleh rata-rata jumlah kuadrat tersebut. *MSE* memberikan bobot yang lebih besar terhadap kesalahan yang besar dibandingkan dengan kesalahan yang kecil, sebab kesalahan dikuadratkan sebelum dijumlahkan. *MSE* dapat dihitung dengan rumus

$$MSE = \sum_{t=1}^{n} \frac{{e_t}^2}{n} \tag{2.48}$$

dengan

MSE : *Mean square error*

e_t : *error* peramalan pada waktu ke-*t*

n : banyaknya pengamatan

Kebaikan suatu model tergantung nilai MAPE dan MSE model tersebut. Model yang baik adalah model yang memiliki MAPE dan MSE yang lebih kecil, artinya model tersebut lebih baik digunakan daripada model yang memberikan MAPE dan MSE besar.

F. Algoritma Genetika

Menurut Wati (2011: 162), algoritma genetika merupakan sebuah metode untuk menyelesaikan masalah optimasi dengan memanfaatkan proses seleksi

alamiah yang dikenal dengan proses evolusi biologis. Dalam proses evolusi hanya individu-individu yang kuat yang mampu bertahan.

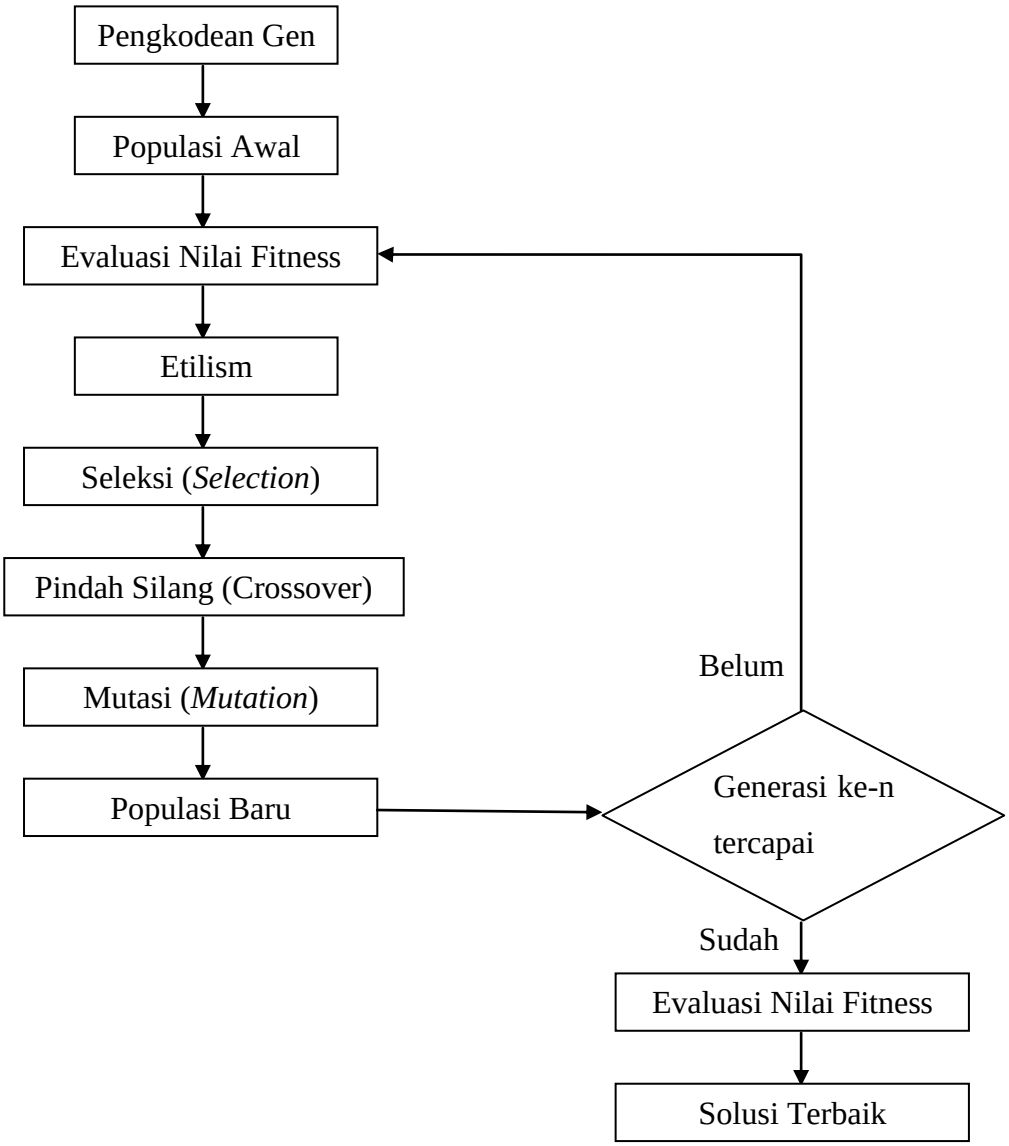
Pada algoritma genetika secara umum untuk semua kasus adalah mendefinisikan individu dan nilai *fitness*, menentukan proses pembangkitan populasi awal, proses seleksi, proses *crossover* dan mutasi gen yang akan digunakan (Ahmad Basuki, 2003: 4).

Algoritma genetika sangat berguna dan efisien untuk masalah dengan karakteristik sebagai berikut (Suyanto, 2005: 3).

a. Ruang masalah sangat besar, kompleks, dan sulit dipahami.
b. Kurang atau bahkan tidak ada pengetahuan yang memadai untuk mempresentasikan masalah ke dalam ruang pencarian yang lebih sempit
c. Tidak tersedianya analisis matematika yang memadai.
d. Ketika metode-metode konvensional sudah tidak mampu menyelesaikan masalah yang dihadapi.
e. Solusi yang diharapkan tidak harus paling optimal, tetapi solusi yang cukup "bagus" atau bisa diterima.

1. Skema Alur Algoritma Genetika

Skema alur algoritma genetika dapat digambarkan sebagai berikut:



Gambar 2.17 Skema algoritma genetika

Pada gambar 2.17 diatas Algoritma Genetika dimulai dengan menentukan nilai-nilai pada gen menggunakan teknik pengkodean. Kumpulan gen akan membentuk kromosom yang kemudian membentuk sebuah individu. Populasi awal dibentuk dari kumpulan beberapa individu. Individu pada populasi akan dievaluasi nilai *fitness*nya yang kemudian dipilih individu dengan nilai *fitness* terbaik. Individu tersebut akan disimpan dan menjalani proses seleksi, *crossover* dan mutasi. Proses tersebut akan menghasilkan individu baru yang digunakan

untuk membentuk populasi baru pada generasi selanjutnya. Langkah-langkah tersebut akan diulang-ulang hingga diperoleh nilai solusi optimal atau setelah tercapai generasi ke-*n*.

2. Komponen Algoritma Genetika

Pada dasarnya, algoritma genetika memiliki enam komponen. Beberapa komponen algoritma genetika diantaranya adalah sebagai berikut.

a. Teknik Pengkodean (Kusumadewi, 2003: 39)

Teknik pengkodean adalah bagaimana mengkodekan gen dari kromosom, dimana gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variabel. Agar dapat diproses melalui algoritma genetika, maka alternatif solusi tersebut harus dikodekan terlebih dahulu kedalam bentuk kromosom. Masing-masing kromosom berisi sejumlah gen yang mengkodekan informasi yang disimpan didalam kromosom.

Gen dapat dipresentasikan dalam bentuk: bit, bilangan real, daftar aturan, elemen permutasi, elemen program atau representasi lainnya yang dapat diimplementasikan untuk operator genetika. Dengan demikian kromosom dapat direpresentasikan dalam beberapa bentuk (Kusumadewi, 2003: 39).

1) String bit : 10011 dst.
2) Array bilangan real : 65.65, -67.98, 77.34 dst.
3) Elemen permutasi : E2, E3, E5 drt.
4) Daftar aturan : R1, R2, R3 dst.
5) Elemen program : pemograman genetika.
6) Struktur lainnya

b. Membangun populasi awal (Kusumadewi, 2003: 42)

Membangkitkan populasi awal adalah proses membangkitkan sejumlah individu secara acak atau melalui prosedur tertentu. Ukuran untuk populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian dilakukan pembangkitan populasi awal.

Teknik dalam pembangkitan populasi awal ini ada beberapa cara, diantaranya adalah *random* generator, pendekatan tertentu, dan permutasi gen. Pembangkitan populasi awal dengan *random* generator adalah melibatkan pembangkitan bilangan acak untuk nilai setiap gen sesuai dengan representasi kromosom yang digunakan. Sedangkan pembangkitan populasi awal dengan pendekatan tertentu adalah memasukan nilai tertentu kedalam gen pada populasi awal yang dibentuk. Nilai tertentu yang digunakan bisa berasal dari bobot pembelajaran *NN*.

c. Fungsi *Fitness* (Ahmad Basuki, 2003: 6)

Algoritma genetika bertujuan mencari individu dengan nilai *fitnes* yang paling tinggi. Evaluasi nilai *fitness* berfungsi untuk mengukur kualitas dari sebuah solusi dan memungkinkan tiap solusi untuk dibandingkan (Michalewicz, 1996: 72). Nilai *fitness* diperoleh dari suatu fungsi yang disesuaikan dengan kasus yang akan diselesaikan. Pada kasus optimasi dengan meminimumkan nilai dari fungsi x, nilai f*itness* dapat diperoleh dengan

$$f = \frac{1}{x} \tag{2.49}$$

Dari persamaan diatas, semakin kecil nilai x maka nilai *fitness* akan semain tinggi. Individu dengan nilai *fitness* tertinggi akan bertahan sampai akhir dalam proses algoritma genetika. Agar nilai *fitness* dapat terdefinisi maka nilai $x \neq 0$ sehingga perlu ditambahkan dengan α, yaitu suatu bilangan yang sangat kecil. (Suyanto, 2005: 10)

$$f = \frac{1}{x+\alpha} \tag{2.50}$$

d. Etilism

Etilitsm merupakan proses untuk mempertahankan individu yang mempunyai nilai *fitness* terbaik. Nilai fitness terbaik perlu dipertahankan karena proses seleksi dilakukan secara acak sehingga tidak ada jaminan bahwa individu dengan *fitness* terbaik akan selalu terpilih. Jika nilai *fitness* terbentuk terpilih, mungkin nilai *fitness* tersebut dapat rusak karena proses pindah silang dan mutasi.

e. Seleksi

Seleksi digunakan untuk memilih individu-individu mana saja yang akan dipilih untuk proses cross-over dan mutasi. Menurut Kusumadewi (2003: 282), terdapat beberapa metode seleksi yaitu seleksi rangking (*rank-based-fitness assigment*), seleksi *roulette wheel* (*roulette wheel selection*), *stochastic universal* sampling, seleksi lokal *(local selection*), seleksi dengan pemotongan (*truncation selection*), dan seleksi dengan turnamen (*tournament selection*). Pada penelitian ini digunakan metode seleksi rangking (*rank-based fitness assigment*).

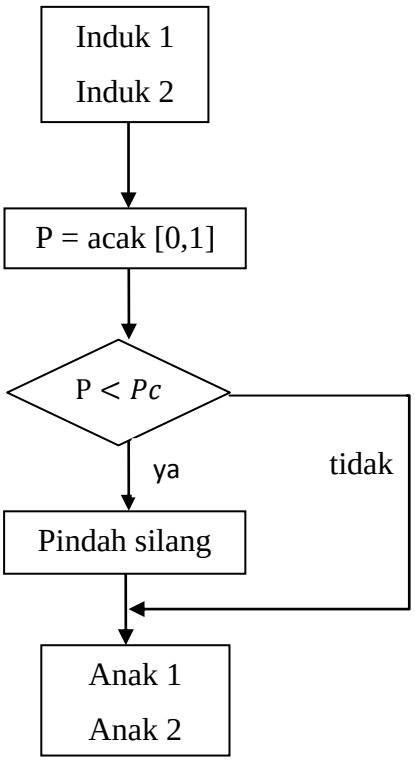
Pada seleksi rangking, individu diurutkan menurut nilai objektifnya. Nilai *fitness* dari tiap-tiap individu hanya tergantung pada posisi individu tersebut dalam urutan. Cara kerja metode seleksi rangking adalah sebagai berikut

1) Dihitung nilai *fitness* dari masing-masing individu (f_i dimana i adalah individu ke-1 s/d ke-n).
2) Nilai fitness diurutkan dari nilai yang terkecil hingga paling besar.
3) Setelah diurutkan, individu terburuk diberi nilai *fitness* baru sebesar 1,individu kedua terburuk diberi nilai 2, dan seterusnya. Individu terbaik diberi nilai *fitness* baru sebesar *n* dimana *n* adalah banyak individu dalam suatu populasi.
4) Dihitung total *fitness* semua individu.
5) Dihitung *probabilitas* masing-masing individu.
6) Dihitung nilai probabilitas kumulatif.
7) Dibandingkan bilangan random antara 0 sampai 1.
8) Dari bilangan random yang dihasilkan, ditentukan individu mana yang terpilih dalam proses seleksi.

f. Pindah Silang (*Cross-over*)

Pindah silang (*Cross-over*) adalah salah satu operator dalam algoritma genetika yang berfungsi untuk membentuk kromosom baru dari dua buah kromosom induk melalui proses penukaran gen yang bersesuaian. Operasi ini hanya dilakukan pada bebarapa individu yang dipilih secara acak berdasarkan nilai Pc *(probability crossover)* yang telah ditentukan. Pc biasanya bernilai antara 0,6 sampai dengan 0,95. Jika pindah silang tidak dilakukan, maka nilai dari induk

akan diwariskan secara langsung pada keturunannya (Michalewicz, 1996: 78). Proses pindah silang dijelaskan pada Gambar 2.18.



Gambar 2.18 Sistematis proses pindah silang

Pada kasus dengan teknik pengkodean bilangan rill, teknik pindah silang yang digunakan adalah pindah silang aritmatika (Syarif, 2014: 39). Pindah silang aritmatika dilakukan dengan menentukan secara acak bilangan random antara 0 sampai 1. Selain itu ditentukan juga 2 bilangan acak *k* untuk menentukan posisi gen yang akan dipindah silangkan. Nilai *k* berada pada rentang [1,*n*] dimana *n* adalah banyaknya gen pada individu. Nilai pada anak diperoleh dengan persamaan sebagai berikut :

$$u'_1(k) = r.u_1(k) + (1-r).u_2(k) \qquad (2.51)$$

$$u'_2(k) = r.u_2(k) + (1-r).u_1(k) \qquad (2.52)$$

dengan

u'_1 : nilai gen pada anak 1

u'_2 : nilai gen pada anak 2

r : nilai acak [0 1]

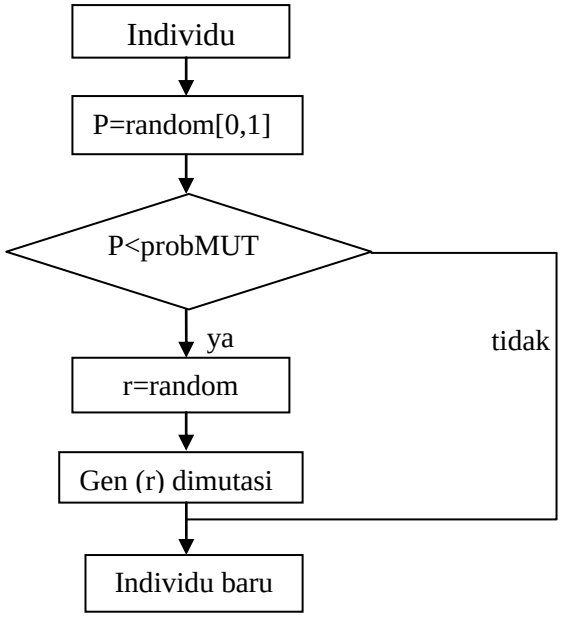
k : posisi gen yang dilakukan pindah silang

u_1 : Nilai gen pada induk 1 yang akan dipindah silangkan

u_2 : Nilai gen pada induk 2 yang akan dipindah silangkan

g. Mutasi

Mutasi meruapakan salah satu operator dalam algoritma genetika pada kromosom dan bertujuan untuk memperoleh kromosom-kromosom baru. Kromosom-kromosom baru ini akan menjadi kandidat solusi pada generasi mendatang dengan nilai *fitness* yang lebih baik dan menuju solusi optimum yang diinginkan. Proses mutasi dijelaskan pada Gambar 2.19 berikut :



Gambar 2.19 Sistematika proses mutasi

h. Pembentukan Populasi Baru

Pembentukan populasi baru akan menghasilkan populasi yang berada dengan populasi awal. Setelah populasi baru terbentuk, dilakukan pengulangan langkah-langkah evaluasi nilai *fitness*, prosess seleksi, proses pindah silang, proses mutasi pada populasi baru untuk membentuk populasi baru selanjutnya. Proses ini akan berakhir hingga diperoleh solusi yang optimal.