BAB II
TINJAUAN PUSTAKA

2.1 Definisi

2.1.1 Saham

Saham adalah tanda penyertaan modal seseorang atau pihak (badan usaha) dalam suatu perseroan terbatas. Seseorang atau pihak (badan usaha) dapat memiliki klaim atas pendapatan dari suatu perusahaan dengan cara membeli saham perusahaan tersebut.

Data saham seringkali mengalami fluktuasi yang tidak menentu. Dalam teori Dow, dijelaskan bahwa pergerakan harga saham dikelompokkan menjadi 3, yaitu:

1. *Primary trend*, yaitu pergerakan harga saham dalam jangka waktu yang lama (tahunan).
2. *Secondary trend*, yaitu pergerakan harga saham yang terjadi selama terjadinya pergerakan harga saham dalam *primary trend*. Biasanya terjadi dalam mingguan atau bulanan.
3. *Minor trend*, yaitu fluktuasi harga saham yang terjadi setiap hari.

2.1.2 Risiko (*Risk*)

Risiko adalah suatu keadaan yang dihadapi seseorang atau perusahaan dimana terdapat kemungkinan yang merugikan. Joel G. Siegel dan Jae K. Sim (2000) mendefinisikan risiko pada 3 hal:

1. Keadaan yang mengarah kepada sekumpulan hasil khusus dimana hasilnya dapat diperoleh dengan kemungkinan yang telah diketahui oleh pengambilan keputusan.
2. Variasi dalam keuntungan penjualan atau variabel keuangan lainnya.
3. Kemungkinan dari sebuah masalah keuangan yang mempengaruhi kinerja operasi perusahaan atau posisi keuangan.

Ditinjau dari segi manusianya, manusia sebagai investor mempunyai kecenderungan didalam rangka menghadapi *risk*. Dalam hal ini ada 3 kecenderungan investor dalam menghadapi *risk*, antara lain :

a. *Risk seeker* yaitu orang yang suka pada risiko. Jika terdapat pilihan investasi dengan risiko yang mempunyai *return* sama, maka *risk seeker* akan lebih memilih investasi yang mempunyai risiko yang lebih besar.

b. *Risk indifference* yaitu orang yang mencari keseimbangan antara *risk* dan *return*. Investor mengharapkan adanya tambahan *return* sesuai dengan tambahan *risk* yang dihadapi.
c. *Risk averter* yaitu orang yang menginginkan tambahan *return* yang lebih besar dengan adanya tambahan *risk* yang dihadapi. *Risk averter* bisa dibilang sebagai orang yang takut pada risiko, investor tipe ini lebih suka menghindari risiko atau memilih investasi yang risikonya lebih kecil.

2.1.3 *Return*

Return atau pengembalian adalah keuntungan yang diperoleh seseorang atau perusahaan dari hasil kebijakan investasi yang dilakukan. Lo, *et al*. (1997) dalam Tsay (2002) mengemukakan bahwa penelitian di bidang ekonomi atau keuangan yang berkaitan dengan saham sering menggunakan data *return*, yang dapat menggambarkan peningkatan atau penurunan harga saham secara langsung. *Return* bertanda negatif terjadi jika harga saham mengalami penurunan dari waktu $t-1$ ke waktu t dan positif jika terjadi peningkatan harga saham dalam selang waktu sama (Surya dan Hariadi, 2002). Pendekatan untuk fluktuasi harga saham adalah penyusun kontinu (*continuously compounded return*) atau *log return*, yaitu:

$$Z_t = Log\frac{X_t}{X_{t-1}} = Log\, X_t - LogX_{t-1} \qquad (2.1)$$

Dalam penelitiannya, Engle (1982) mengemukakan walaupun *return* (Z_t) secara serial tidak berkorelasi, namun terdapat ketergantungan pada *return* kuadrat (Z_t^2). Taylor (1986) juga mengungkapkan adanya ketergantungan pada nilai mutlak *return* $|Z_t|$. Menurut Ding, *et al*. (1993) jika suatu deret merupakan proses *iid* (*independent and identically distributed*), maka transformasi dari deret tersebut juga sebuah proses *iid*, sebagai contoh pada Z_t^2 dan $|Z_t|$. Jika model deret waktu (ARIMA) digunakan untuk memodelkan *return* saham, maka akan menghasilkan ragam sisaan yang tidak konstan sebagai akibat dari karaketeristik *return* saham itu sendiri.

2.1.4 Volatilitas

Perilaku saham dijelaskan oleh dua parameter, yaitu rata-rata dan ragam. Ragam didefinisikan sebagai volatilitas yaitu ukuran ketidakpastian dari suatu data deret waktu keuangan atau risiko yang mungkin dihadapi investor dalam perdagangan bursa (Surya dan

Hariadi, 2002). Semakin besar nilai volatilitas, maka akan semakin besar pula kemungkinan harga saham tersebut akan naik atau turun yang disebut fluktuatif. Berdasarkan fakta tersebut, para pelaku bisnis sektor finansial tertarik untuk melakukan pemodelan dan peramalan volatilitas harga saham yang dikenal fluktuatif.

2.1.5 *Value at Risk*

Value at Risk (*VaR*) didefinisikan sebagai penduga kerugian yang didapatkan oleh investor selama periode tertentu. *VaR* dapat merepresentasikan seberapa besar investor merugi dalam periode waktu investasi *t* dengan tingkat kepercayaan (1-α) (Maruddani, 2009).

Menurut Jorion (2002), *VaR* merupakan kerugian selama waktu tertentu dengan selang kepercayaan waktu tertentu. Secara matematis *VaR* dapat didefinisikan sebagai berikut :

$$VaR = -S \times \left[\hat{Z}_t - Z_\alpha \, \hat{\sigma}_t\right] \qquad (2.2)$$

di mana :

VaR : besarnya resiko
S : besarnya investasi
$\hat{Z}_t$: penduga *return*
$\hat{\sigma}_t$: penduga volatilitas waktu ke-*t*
$\hat{Z}_\alpha$: nilai kritis sebaran normal pada α tertentu. Pada kasus ini α = 5%, sehingga $\hat{Z}_{0.05}$= 1.645.

2.1.6 *Capital Gain*

Capital Gain adalah keuntungan yang diterima karena adanya selisih antara harga jual dan harga beli yang terbentuk dengan adanya aktivitas perdagangan saham di pasar modal (Jogiyanto, 2000).

$$Capital\ Gain = \frac{P_t - (P_{t-1})}{P_{t-1}} \qquad (2.3)$$

di mana P_t adalah harga saham penutupan. Apabila P_t bernilai lebih tinggi daripada P_{t-1} maka investor mengalami keuntungan (*gain*). Namun jika sebaliknya, investor akan mengalami kerugian (*loss*).

2.2 Data Deret Waktu

Data deret waktu (Z_t) adalah hasil pengamatan yang diperoleh pada waktu (*t*) yang berbeda dengan selang waktu yang sama, di mana hasil pengamatan antar selang waktu sama diasumsikan saling berhubungan (Box dan Jenkins, 1994). Analisis deret waktu pada dasarnya digunakan untuk melakukan analisis data yang memper-

timbangkan pengaruh waktu. Tujuan dari analisis deret waktu adalah untuk meramalkan Z_t di luar selang *t* berdasarkan nilai Z_t di dalam selang *t* (Cryer dan Chan, 2008).

2.2.1 Kestasioneran Deret Waktu

Salah satu asumsi penting dalam analisis deret waktu adalah stasioneritas. Menurut Makridakis (1999) data stasioner terjadi jika tidak ada pertambahan atau pengurangan nilai Z_t secara nyata saat *t* bertambah. Data deret waktu dikatakan stasioner jika Z_t membentuk pola horizontal pada selang waktu *t*, sehingga fluktuasi Z_t berada di sekitar nilai rata-rata $\bar{Z}$ yang konstan. Terdapat dua macam kestasioneritasan data:

1. Stasioner terhadap ragam.

Ragam dikatakan stasioner apabila fluktuasi Z_t konstan sepanjang waktu. Sifat heteroskedastis data dicerminkan dari ragam yang tidak stasioner yang ditunjukkan oleh nilai λ dari Transformasi Box-Cox dengan bentuk transformasi :

$$Z_t(\lambda) = \frac{Z_t^{\lambda}-1}{\lambda} \qquad (2.4)$$

di mana:

$Z_t(\lambda)$: data transformasi
Z_t : pengamatan pada waktu ke-*t*
λ : parameter transformasi

Jika nilai λ ± 1, maka data dikatakan telah stasioner terhadap ragam atau dapat dikatakan ragam bersifat homoskedastis (Cryer, 1986).

2. Stasioner terhadap rata-rata

Data dikatakan stasioner terhadap rata-rata apabila pada plot autokorelasi dari keseluruhan data, 95% data masuk dalam selang $0 \pm \frac{2}{\sqrt{n}}$. Apabila data tidak stasioner terhadap rata-rata, maka perlu dilakukan *differencing* yaitu deret asli diganti dengan deret selisih hingga mencapai kestasioneran yang dinotasikan dengan *d* (Hanke dkk, 2003). Menurut Tsay (2002) uji stasioneritas yang sering digunakan adalah *Augmented Dicky-Fuller* (ADF). Misalkan terdapat model deret waktu sebagai berikut:

$$Z_t = \phi Z_{t-1} + \varepsilon_t \qquad (2.5)$$

jika pada persamaan (2.2) pada masing-masing ruas dikurangi dengan Z_{t-1}, maka persamaan tersebut akan menjadi :

$$Z_t - Z_{t-1} = \phi Z_{t-1} - Z_{t-1} + \varepsilon_t$$
$$\Delta Z_t = (\phi - 1) Z_{t-1} + \varepsilon_t$$
$$\Delta Z_t = \phi^* Z_{t-1} + \varepsilon_t \quad (2.6)$$

dimana ε_t adalah sisaan dan untuk menguji hipotesis :

$H_0 : \phi^* = 0$ (data tidak stasioner)
$H_1 : \phi^* < 0$ (data stasioner)

Statistik uji yang digunakan adalah :

$$\tau = \frac{\hat{\phi}^*}{SE(\hat{\phi}^*)} \quad (2.7)$$

$$SE(\hat{\phi}^*) = \frac{\sigma_{(\hat{\phi}^*)}}{\sqrt{n}}$$

di mana :

$\hat{\phi}^*$ = nilai duga parameter *Augmented Dicky-Fuller*
$SE(\hat{\phi}^*)$ = standar *error* $\hat{\phi}^*$
$\sigma_{(\hat{\phi}^*)}$ = standar deviasi dari $\hat{\phi}^*$

Selanjutnya dilakukan perbandingan $\tau \sim \tau_{(\alpha,n)}$ berdasarkan tabel ADF dengan kriteria uji sebagai berikut :

a. Jika $\tau > \tau_{(\alpha,n)}$, maka H_0 diterima sehingga stasioneritas tidak terpenuhi.
b. Jika $\tau < \tau_{(\alpha,n)}$, maka H_0 ditolak sehingga stasioneritas terpenuhi.

Menurut Hanke, et all (2003), *data* tidak stasioner terhadap rata-rata dapat ditangani agar menjadi stasioner melalui *differencing*, yaitu:

$$\nabla^d Z_t = \nabla^{d-1} Z_t - \nabla^{d-1} Z_{t-1} \quad (2.8)$$

di mana *d* adalah derajat pembedaaan (d = 1,2,3,…) dan $\nabla^0 Z_t = Z_t$. Pembedaan dilakukan hingga plot $\nabla^0 Z_t$ terhadap *t* tidak menunjukkan pola trend atau uji ADF pada $\nabla^0 Z_t$ menolak H_0.

2.2.2 Fungsi Autokorelasi (ACF)

Autokorelasi menyatakan bagaimana keterkaitan nilai Z_t dengan Z_{t-k} , dimana k = 1, 2, … merupakan waktu keterlambatan atau *lag* (Cryer, 1986). Fungsi autokorelasi ρ_k dapat diduga dengan r_k :

$$r_k = \frac{\sum_{t=k+1}^{n} (Z_t - \bar{Z})(Z_{t-k} - \bar{Z})}{\sum_{t=1}^{n} (Z_t - \bar{Z})^2} \quad (2.9)$$

$$\bar{Z} = \frac{\sum_{t=1}^{n} Z_t}{n}$$

di mana :

r_k = fungsi autokorelasi

Z_t = respons pada waktu t

Z_{t-k} = respons pada waktu t-k

$\bar{Z}$ = rata-rata respons

n = ukuran contoh

2.2.3 Fungsi Autokorelasi Parsial (PACF)

Autokorelasi parsial digunakan untuk mengukur tingkat keeratan antara Z_t dan Z_{t-k} (Makridakis, 1999). Menurut Cryer (1986), pendugaan dari PACF merupakan koefisien autokorelasi dari persamaan *Yule-Walker* untuk $j = 1,2, \ldots, k$:

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

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

$$\vdots$$

$$\rho_k = \phi_{k1}\rho_{k-1} + \phi_{k2}\rho_{k-2} + \cdots + \phi_{kk}\rho_0 \qquad (2.10)$$

Sehingga pendugaan dari PACF adalah sebagai berikut :

$$\phi_{kk} = \frac{\rho_k - \sum_{j=1}^{k-1} \phi_{k-1,j}\rho_{k-1}}{1 - \sum_{j=1}^{k-1} \phi_{k-1,j}\rho_j} \qquad (2.11)$$

dengan : $\phi_{kj} = \phi_{k-1,j} - \phi_{kk}\phi_{k-1,k-j}$ untuk j = 1, 2, …, k-1, di mana :

ϕ_{kk} = koefisien autokorelasi parsial pada *lag k*

ρ_k = koefisien autokorelasi pada *lag k* yang diduga dengan r_k

ρ_j = koefisien autokorelasi pada *lag j* yang diduga dengan r_j

ρ_{k-j} = koefisien autokorelasi pada *lag k-j* yang diduga dengan r_{k-j}.

2.3 ARIMA(*Autoregressive Integrated Moving Average*)

Model ARIMA merupakan model yang sering dipakai dalam melakukan peramalan dengan asumsi stasioneritas terpenuhi. Cryer (1986) merumuskan beberapa model umum ARIMA sebagai berikut:

1. Model ARIMA(p,d,q)

$$W_t = \nabla^d Z_t$$

$$W_t = \alpha + \phi_1 W_{t-1} + \ldots + \phi_p W_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \ldots - \theta_q \varepsilon_{t-q} \qquad (2.12)$$

2. Model ARMA(p,q)

$$Z_t = \alpha + \phi_1 Z_{t-1} + \ldots + \phi_p Z_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \ldots - \theta_q \varepsilon_{t-q} \quad (2.13)$$

3. Model AR(p)

$$Z_t = \alpha + \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + \ldots + \phi_p Z_{t-p} + \varepsilon_t \quad (2.14)$$

4. Model MA(q)

$$Z_t = \alpha + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q} \quad (2.15)$$

dimana:

ϕ = parameter *autoregressive*
θ = parameter *moving average*
p = derajat *autoregressive*
d = derajat *difference*
q = derajat *moving average*
ε_t = galat acak (*white noise*)

Menurut Wei (1990), prosedur pemodelan ARIMA adalah sebagai berikut:

1. Plot data deret waktu

Dilakukan untuk mengetahui apakah data tersebut mengandung tren, musiman, atau ragam tidak konstan.

2. Identifikasi karakteristik dari ACF dan PACF

Orde p dan q model ARIMA (p,d,q) diidentifikasi berdasarkan cirri-ciri ACF dan PACF yang ditunjukkan pada Tabel 2.1 berikut :

Tabel 2.1 Karakterisrik ACF dan PACF

Proses	ACF	PACF
AR(p)	Pola turun eksponesial	Beda nyata pada lag 1 sampai lag ke-p
MA(q)	Beda nyata pada lag 1 sampai lag ke-q	Pola turun eksponesial
ARMA(p,q)	Pola turun setelah lag ($q-p$)	Pola turun setelah lag ($p-q$)

3. Pendugaan parameter

Metode yang sering digunakan untuk melakukan pendugaan parameter pada model ARIMA adalah *Maximum Likelihood*. Misalkan terdapat model ARMA (p,q) sebagai berikut :

$$\hat{Z}_t = \phi_1 \hat{Z}_{t-1} + \cdots + \phi_p \hat{Z}_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \cdots - \theta_q \varepsilon_{t-q} \quad (2.16)$$

dimana $\hat{Z}_t = Z_t - \mu$ dan $e_t \sim iidN(0, \sigma_\varepsilon^2)$, parameter-parameter yang diduga yaitu $\phi = (\phi_1, \ldots, \phi_p)$, $\theta = (\theta_1, \ldots, \theta_p)$, $\mu = E(Z_t)$, dan $\sigma_\varepsilon^2 = E(\varepsilon_t^2)$. Fungsi kepekatan peluang dari $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$ adalah :

$$f(\varepsilon|\phi, \theta, \mu, \sigma_\varepsilon^2) = (2\pi\sigma_\varepsilon^2)^{-n/2} exp(-\frac{1}{2\sigma_\varepsilon^2} \sum_{t=1}^{n} \varepsilon_t^2) \quad (2.17)$$

dari persamaan 2.16 dapat ditulis dalam bentuk persamaan :

$$\varepsilon_t = \theta_1 \varepsilon_{t-1} + \cdots + \theta_p \varepsilon_{t-q} + \hat{Z}_t - \phi_1 \hat{Z}_{t-1} - \cdots - \phi_p \hat{Z}_{t-p} \quad (2.18)$$

dengan parameter $\phi, \mu, \sigma_\varepsilon^2$ maka fungsi *log-likelihood* adalah :

$$\ell(\phi, \mu, \theta, \sigma_\varepsilon^2) = -\frac{(n-p)}{2} ln 2\pi\sigma_\varepsilon^2 - \frac{S_*(\phi,\mu,\theta)}{2\sigma_\varepsilon^2} \quad (2.19)$$

di mana

$$S_*(\phi, \mu, \theta) = \sum_{t=1}^{n} \varepsilon_t^2(\phi, \mu, \theta|Z)$$

merupakan fungsi jumlah kuadrat bersyarat. Penduga *maximum likelihood* bersyarat dari persamaan 2.19 yaitu $\hat{\phi}$, $\hat{\theta}$, dan $\hat{\mu}$.

4. Uji signifikansi parameter

Tahap ini dilakukan untuk menguji keberartian parameter model ARIMA. Hipotesis yang mendasari uji ini:
H_0 : parameter model tidak nyata
H_1 : parameter model nyata
Anggap model AR(1), dan jika H_0 benar maka:

$$\frac{\hat{\phi}_1}{Se(\hat{\phi}_1)} \sim t_{(n-1)} \quad (2.20)$$

di mana:
$\hat{\phi}_1$ = penduga parameter ARIMA(1,0,0)
$Se(\hat{\phi}_1)$ = salah baku statistik $\hat{\phi}_1$

5. Pemilihan model terbaik

Menurut Enders (2004), AIC (*Akaike Information Criterion*) terkecil digunakan untuk memilih model terbaik dari beberapa model sementara yang layak. Perhitungan nilai AIC dilakukan dengan rumus :

$$AIC = n \ln(\hat{\sigma}_\varepsilon^2) + 2m \quad (2.21)$$

di mana :
n = banyaknya pengamatan
$\hat{\sigma}_{\varepsilon}^2$ = penduga ragam sisaan
m = banyaknya parameter dalam model.

2.4 Pengujian Adanya Efek GARCH

Salah satu cara untuk mengidentifikasi adanya proses GARCH yaitu menggunakan pengujian *Lagrange Multiplier* (LM). Engle (1982) menyebutkan pengujian GARCH dengan LM dilakukan dengan meregresikan kuadrat sisaan ke-*t* terhadap 1 hingga *q lag* kuadrat sisaan yaitu :

$$\varepsilon_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \cdots + \alpha_q \varepsilon_{t-q}^2 + \varepsilon_t ; \quad t = q+1, \ldots, T \qquad (2.22)$$

Tahapan pengujian uji *Lagrange Multiplier* adalah :

1. Hipotesis
   H_0 : tidak terdapat proses GARCH
   H_1 : terdapat proses GARCH
2. Statistik uji
   $$\text{LM} = TR^2 \qquad (2.23)$$
   di mana:
   T = jumlah observasi
   R^2 = koefisien determinasi dari regresi antara kuadrat sisaanke-*t* dengan kuadrat sisaan ke *t-b.*
3. Daerah penolakan
   $$TR^2 > \chi^2_{\alpha,b} \qquad (2.24)$$
   Jika $TR^2 > \chi^2_{\alpha/2,b}$, maka H_0 ditolak yang berarti dalam kuadrat sisaan tersebut terdapat proses GARCH.

2.5 Model ARCH (*Autoregressive Conditional Heteroscedastic*)

Model ARCH merupakan suatu teknik pemodelan data untuk mengatasi tidak terpenuhinya asumsi ragam stasioner. Secara umum ARCH berorde p (ARCH (p)) digunakan untuk memodelkan ragam bersyarat (σ_t^2) pada waktu t berdasarkan kuadrat sisaan pada waktu $t-1$ hingga $t-p$. Misalkan terdapat model rataan:

$$Z_t = \mu_t + \varepsilon_t$$

Tsay (2002) menyatakan bahwa μ_t merupakan nilai harapan Z_t bersyarat F_{t-1}, dimana $F_{t-1} = \{Z_{t-1}, Z_{t-2}, \dots, Z_2, Z_1\}$ dan ε_t merupakan komponen random dari model (sering disebut sebagai proses *white noise*), di mana $E(\varepsilon_t) = 0$ dan bersifat tidak berkorelasi dengan waktu lampau atau waktu yang akan datang. Engle (1982) menguraikan nilai ε_t sebagai berikut:

$$\varepsilon_t = \epsilon_t \sigma_t , \qquad \varepsilon_t | F_{t-1} \sim iidN(0, {\sigma_t}^2) \qquad (2.25)$$
$$\epsilon_t \sim iidN(0,1)$$

di mana σ_t merupakan akar dari σ_t^2 dan ϵ_t adalah proses i.i.d. (*independent* dan *identically distributed*) yang seringkali diasumsikan berdistribusi normal standard N(0,1).

Berdasarkan persamaan 2.25, maka diperoleh ragam bersyarat bagi ε_t (Cryer dan Chan, 2008):

$$Var(\varepsilon_t | F_{t-1}) = E(\varepsilon_t^2 | F_{t-1}) = E(\epsilon_t^2 \sigma_t^2 | F_{t-1}) = \sigma_t^2 E(\epsilon_t^2 | F_{t-1}) = \sigma_t^2 \qquad (2.26)$$

dan ragam bersyarat yang mendefinisikan model ARCH dengan orde p yaitu:

$$\pm_t^2 = \sim_0 + \sum_{i=1}^{p} \sim_i £_{t-i}^2 \qquad (2.27)$$

α_0 dan α_i merupakan parameter ARCH, $\alpha_0 > 0$, $\alpha_i \geq 0$, $p > 0$ dan $i = 1,2, \dots, p$.

2.6 Model GARCH (*Generalized* ARCH)

Model GARCH (*Generalized Autoregressive Conditional Heteroscedasticity*) merupakan pengembangan model ARCH yang diperkenalkan oleh Bollerslev (1986). GARCH merupakan model sederhana dengan jumlah parameter lebih sedikit dibandingkan model ARCH berorde tinggi. Secara umum GARCH dengan orde p dan q ditulis GARCH (p,q), di mana ragam bersyarat waktu t dimodelkan sebagai fungsi dari kuadrat sisaan pada waktu $t-1$ hingga $t-p$ dan ragam bersyarat pada waktu $t-1$ hingga $t-q$. Misalkan terdapat model rataan:

$$Z_t = \mu_t + \varepsilon_t$$

di mana μ_t merupakan nilai harapan Z_t bersyarat F_{t-1}, dimana $F_{t-1} = \{Z_{t-1}, Z_{t-2}, \dots, Z_2, Z_1\}$ dan ε_t merupakan komponen random dari model

(sering disebut sebagai proses *white noise*), di mana $E(\varepsilon_t) = 0$ dan bersifat tidak berkorelasi dengan waktu lampau atau waktu yang akan datang. Jika Z_t dimodelkan ARIMA, maka :

$$\mu_t = E(Z_t | F_{t-1}) = \theta_0 + \sum_{i=1}^{p} \phi_i Z_{t-i} - \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} \qquad (2.28)$$

di mana:

Z_t = respons pada waktu t

F_{t-1} = seluruh himpunan informasi pada waktu 1 hingga $t-1$

μ_t = nilai harapan Z_t bersyarat F_{t-1}

ε_t = sisaan ARIMA waktu t

Berdasarkan uraian ε_t pada persamaan 2.25 dan ragam bersyarat pada 2.27, maka diperoleh ragam bersyarat dari sisaan model GARCH berorde p dan q yaitu (Cryer dan Chan, 2008):

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \qquad (2.29)$$

di mana α_0, α_i dan β_j adalah parameter GARCH, $\alpha_0 > 0$, $\alpha_i \geq 0$, $\beta_j \geq 0$, $p > 0$, $q \geq 0$ untuk $i = 1,2,\ldots,p$ dan $j = 1,2,\ldots,q$.

Model pada persamaan (2.29) memberikan informasi bahwa ragam sisaan bersyarat dipengaruhi oleh kuadrat sisaan pada *p* periode yang lalu dan ragam sisaan pada *q* periode yang lalu. Keunggulan model GARCH bila dibandingkan dengan model ARCH yaitu pada data yang sama memberikan kemudahan dalam mengestimasi parameter.

2.6.1 Model GARCH (1,1)

GARCH (1,1) merupakan model GARCH yang mana ragam bersyaratnya dipengaruhi oleh kuadrat sisaan dan ragam sisaan dari satu periode sebelumnya. Lo (2003) menjelaskan bahwa model GARCH (1,1) merupakan model yang paling sering digunakan untuk meramalkan data saham. Model GARCH (1,1) dinyatakan sebagai:

$$Z_t = \mu + \varepsilon_t, \quad \varepsilon_t \sim N(0, \sigma^2)$$

$$\sigma_t^2 = \alpha_0 + \beta_1 \sigma_{t-1}^2 + \alpha_1 \varepsilon_{t-1}^2 \qquad (2.30)$$

Menurut Hariadi (2004), model GARCH merupakan pengembangan dari model ARCH. Saat dilakukan pencocokan (*fitting*) untuk model ARCH, diperlukan derajat yang tinggi (*high order*) untuk mendapatkan model yang tepat. Peningkatan derajat ARCH tidak serta

merta meningkatkan ketepatan model ARCH (Hariadi, 2004). Oleh karena itu model GARCH (1,1) merupakan jawaban dari peningkatan derajat ARCH (*p*).

2.6.2 Pendugaan Parameter GARCH (1,1)

Pendugaan parameter untuk model GARCH (1,1) menggunakan metode *Maximum Likelihood*. Jika sisaan menyebar normal dengan rata-rata nol dan ragam σ_t^2 atau $\varepsilon_t \sim N(0, \sigma_t^2)$, maka fungsi kepekatan peluang untuk sisaan dapat dituliskan:

$$f(\varepsilon_1, \ldots, \varepsilon_n|\eta) = f(\varepsilon_n|F_{n-1}) f(\varepsilon_{n-1}|F_{n-2}) \ldots f(\varepsilon_{p+1}|F_p) f(\varepsilon_1, \ldots, \varepsilon_p|\eta)$$

$$= \prod_{t=p+1}^{n} \frac{1}{\sqrt{2\pi\sigma_t^2}} \exp\left[-\frac{\varepsilon_t^2}{2\sigma_t^2}\right] xf(\varepsilon_1, \ldots, \varepsilon_p|\eta)$$

di mana $\eta = (\alpha_0, \alpha_1, \ldots, \alpha_p)$ dan $f(\varepsilon_1, \ldots, \varepsilon_p|\eta)$ adalah fungsi kepekatan peluang bersama dari $\varepsilon_1, \ldots, \varepsilon_p$ dengan fungsi *likelihood* bersyarat:

$$f(\varepsilon_{p+1}, \ldots, \varepsilon_n | \eta, \varepsilon_1, \ldots, \varepsilon_p) = \prod_{t=p+1}^{n} \left\{\frac{1}{\sqrt{2\pi\sigma_t^2}}\right\} \exp\left\{-\frac{\varepsilon_t^2}{2\sigma_t^2}\right\}$$

Fungsi *log likelihood* bersyarat adalah:

$$\lambda(\varepsilon_{p+1}, \ldots, \varepsilon_n | \eta, \varepsilon_1, \ldots, \varepsilon_p) = \sum_{t=p+1}^{n} \left[-\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(\sigma_t^2) - \frac{1}{2}\frac{\varepsilon_t^2}{\sigma_t^2}\right]$$

karena $\ln(2\pi)$ tidak melibatkan parameter, fungsi *log likelihood* bersyarat menjadi:

$$\lambda(\varepsilon_{p+1}, \ldots, \varepsilon_n | \eta, \varepsilon_1, \ldots, \varepsilon_p) = -\sum_{t=p+1}^{n} \left[\frac{1}{2}\ln(\sigma_t^2) - \frac{1}{2}\frac{\varepsilon_t^2}{\sigma_t^2}\right] \quad (2.31)$$

di mana $\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \cdots + \alpha_p \varepsilon_{t-p}^2$

Jika sisaan menyebar normal dengan rata-rata nol dan ragam σ_t^2 , maka fungsi log *likelihood* untuk L adalah sebagai berikut (Enders, 2004):

$$\ln(L) = l = -\frac{T}{2}\ln(2\pi) - \frac{1}{2}\sum_{t=1}^{T} ln\,\sigma_t^2 - \frac{1}{2}\sum_{t=1}^{T}\left(\frac{\varepsilon_t^2}{\sigma_t^2}\right) \quad (2.32)$$

Pendugaan parameter GARCH dilakukan dengan memaksimumkan fungsi log *likelihood* pada persamaan (2.32).

2.7 *Artificial Neural Network* (ANN)

2.7.1 Definisi ANN

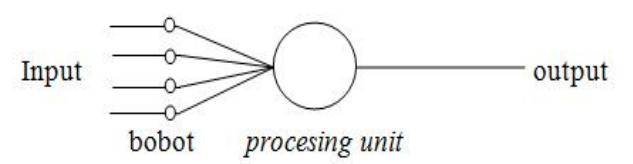
Artificial Neural Network (ANN) disebut juga Jaringan Syaraf Tiruan (JST) merupakan salah satu sistem pemrosesan informasi yang didesain dengan menirukan cara kerja otak manusia dalam menyelesaikan suatu masalah dengan melakukan proses belajar melalui perubahan bobot. ANN melakukan pengenalan pola data masa lalu untuk dipelajari sehingga ANN mampu memberikan keputusan terhadap data yang belum pernah dipelajari (Kusumadewi, 2008).

ANN merupakan suatu model matematis yang berupa sistem pengolahan informasi yang mengadopsi kinerja jaringan syaraf biologis. ANN dapat dipertimbangkan sebagai teknik pengolahan data yang mampu memetakan informasi arus *input* berupa deret waktu dan *output* berupa peramalan *item* berikutnya (Wardani, 2006).

ANN dibentuk dengan asumsi sebagai berikut :

1. Pemrosesan informasi pada elemen sederhana (neuron).
2. Sinyal dikirim melalui penghubung diantara neuron.
3. Penghubung diantara neuron memiliki bobot.
4. Dalam menentukan *output* (keluaran), setiap neuron menggunakan fungsi aktivasi yang dikenakan pada setiap *input* (masukan) yang diterima.

Neuron adalah bagian dasar dalam pemrosesan suatu jaringan. Pada ANN, neuron (*node*) akan dikumpulkan dalam lapisan-lapisan yang disebut lapisan neuron (*neuron layer*).



Gambar 2.1 Bentuk Dasar Neuron (Lesmana, 2009)

Neuron-neuron dalam satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya, sehingga nantinya akan terbentuk lapisan *input* (*input layer*), lapisan tersembunyi (*hidden layer*), dan lapisan *output* (*output layer*). Menurut Lesmana (2009), sebuah neuron (*node*) terdiri dari beberapa bagian :

1. *Input*

   Input merupakan bagian dari sistem yang digunakan untuk memberikan masukan pada sistem yang digunakan untuk proses pembelajaran dan proses pengenalan objek.

2. Bobot

Bobot merupakan beban yang diberikan pada penghubung yang berfungsi untuk meningkatkan serta menurunkan pengaruh pada suatu neuron terhadap *input* yang masuk agar nantinya dihasilkan *output* yang sesuai dengan target pembelajaran.

3. *Processing unit*

Processing unit merupakan tempat terjadinya proses komputasi atau pengenalan terhadap objek berdasarkan pengetahuan yang diperoleh dari *input* dan juga dari bobot yang sudah ditentukan sebelumnya.

4. *Output*

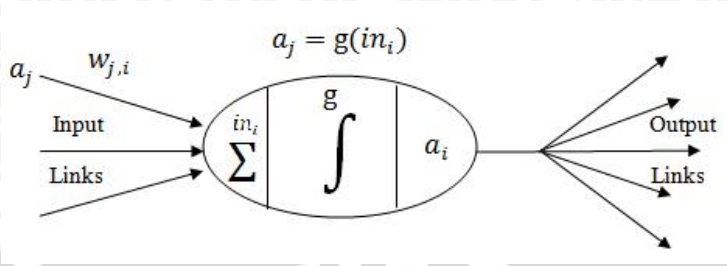
Output merupakan bagian yang memberikan hasil keluaran atau solusi pemecahan masalah dari proses pembelajaran yang berlangsung terhadap *input*.

Secara umum ANN memiliki beberapa komponen, yaitu neuron, lapisan, fungsi aktivasi, dan bobot. Pemodelan ANN dilihat pada bentuk jaringan yang terdiri dari jumlah *neuron* pada lapisan *input*, jumlah *neuron* pada lapisan tersembunyi (*hidden layer*) dan jumlah *neuron* pada lapisan *output*, serta fungsi aktivasi yang digunakan (Suhartono, 2007).

2.7.2 Pemodelan ANN

Pemodelan di dalam ANN terbagi menjadi dua yaitu proses *training* dan *testing*. Proses *training* merupakan proses pembelajaran dari sistem yang mengatur *input*, serta pemetaannya pada *output* sampai didapatkan model yang sesuai. Proses ini terjadi ketika mengatur bobot dan bias. Sedangkan proses *testing* adalah proses pengujian ketelitian dari model yang didapatkan setelah proses *training*. Menurut pengalaman Yao dan Tan (2001), pembagian data *testing* dan *training* masing-masing adalah 20% dan 80%.

Penggambaran struktur ANN seperti pada Gambar 2.3. Sinyal masuk *a* dikalikan dengan masing-masing bobot yang bersesuaian *w*. Kemudian seluruh hasil perkalian dijumlahkan dan keluaran yang dihasilkan diteruskan ke dalam fungsi aktivasi untuk mendapatkan tingkat derajat keluarannya *F(a,w).*



Gambar 2.2 Model Tiruan Sebuah ANN (Kusumadewi, 2008)

di mana:

a_j = nilai aktivasi dari unit j

a_i = nilai aktivasi dari unit i

$w_{j,i}$ = bobot dari unit j ke i

in_i = penjumlahan bobot dan masukan ke unit i

g = fungsi aktivasi

Menurut Suhartono (2007), secara umum model ANN bekerja dengan menerima suatu vektor dari *input* x dan kemudian menghitung suatu respon atau *output* ŷ(x) dengan memproses (*propagating*) x melalui elemen-elemen proses yang saling terkait. Elemen-elemen proses tersusun dalam beberapa lapisan dan data *input* x mengalir dari satu lapisan ke lapisan berikutnya secara berurutan. Dalam tiap-tiap lapisan, *input-input* ditransformasi ke dalam lapisan secara nonlinier oleh elemen-elemen proses dan kemudian diproses maju ke lapisan berikutnya. Akhirnya, nilai-nilai *output* ŷ yang berupa nilai skalar atau vektor, dihitung pada lapisan *output* dengan persamaan sebagai berikut:

$$\hat{y}_{(k)} = f^o\left[\left[\sum_{j=1}^{q} v_j^o f_j^h\left(\sum_{j=1}^{p} w_{j,i}^h x_{i(k)} + b_j^h\right) + b^o\right]\right] \qquad (2.33)$$

di mana:

$\hat{y}_{(k)}$ = nilai dugaan dari peubah *output*

k = indeks pasangan data *input*-target $(x_{i(k)}, \hat{y}_{(k)})$, k = 1,2,...,*n*, dimana *n* merupakan jumlah pola

f^o = fungsi aktivasi pada *neuron* di lapisan *output*

v_j^o = bobot dari *neuron* ke-*j* di lapisan tersembunyi menuju neuron pada lapisan *output*

f_j^h = fungsi aktivasi di *neuron* ke-*j* pada lapisan tersembunyi

$w_{j,i}^h$ = bobot dari *input* ke-i yang menuju neuron ke-*j* pada lapisan tersembunyi, (*j*=1,2,...,*q*)

$x_{i(k)}$ = peubah *input* ke-*i*, (*i* = 1,2,...,*p*) dimana *p* merupakan jumlah *input*

b^h = bias pada *neuron* di lapisan tersembunyi

b^o = bias pada *neuron* di lapisan *output*.

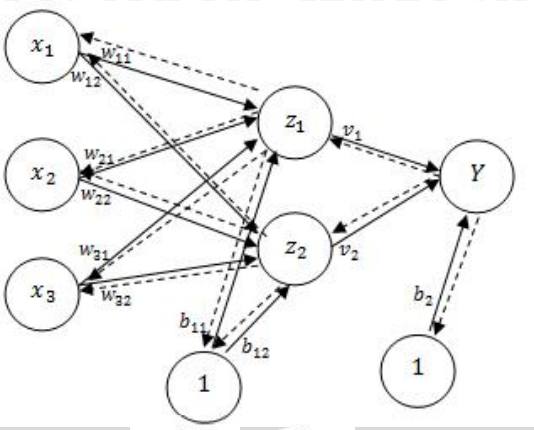
2.7.3 Arsitektur ANN

Sebuah ANN minimal tersusun atas lapisan *input* dan lapisan *output*. Dalam beberapa tipe jaringan, diantara lapisan *input* dan lapisan *output* terdapat lapisan tersembunyi (*hidden layer*). Lapisan *input* merupakan aktifitas unit-unit lapisan *input* yang menunjukkan informasi dasar yang digunakan dalam ANN. *Hidden layer* merupakan aktifitas setiap unit-unit *hidden layer* yang ditentukan oleh aktifitas dari unit-unit *input* dan bobot dari koneksi antara unit-unit *input* dan unit-unit *hidden layer*. Sedangkan lapisan *output* merupakan karakteristik dari unit-unit *output* tergantung dari aktifitas unit-unit *hidden layer* dan bobot antara unit-unit *hidden layer* dan unit-unit *output*.

Hal ini berarti bahwa semua neuron pada lapisan *input* akan berhubungan ke semua *neuron* dalam *hidden layer* yang selanjutnya setiap *neuron* dalam *hidden layer* akan dihubungkan ke semua *neuron* di lapisan *output*. Pada setiap lapisan biasanya *neuron* mempunyai fungsi aktivasi serta pola hubungan ke *neuron* lain yang sama.

Menurut Siang (2005) berdasarkan jumlah lapisannya ANN dibagi menjadi tiga, yaitu *single layer network, multi layer network, dan reccurent network*. Tipe *multi layer network* merupakan tipe yang paling banyak digunakan dan telah menghasilkan hasil yang cukup baik dalam pemodelan dan peramalan. Oleh karena itu, arsitektur yang digunakan dalam penelitian ini adalah *Multi Layer Network* atau *Feedforward Neural Network (FFNN)*.

Jaringan dengan banyak lapisan memiliki satu atau lebih lapisan yang terletak diantara lapisan *input* dan lapisan *output*, seperti terlihat pada Gambar 2.4 berikut:



Gambar 2.3 Jaringan Syaraf dengan Banyak Lapisan (Warsito, 2007)

Arsitektur jaringan pada Gambar 2.3 tersusun atas 3 unit *input* pada lapisan *input* yaitu x_1, x_2 , dan x_3 ; 1 *hidden layer* dengan 2 unit *hidden* yaitu z_1 dan z_2 ; dan 1 unit *output* pada lapisan *output* yaitu y, di mana:

w_{ij} = bobot yang menghubungkan neuron *input* ke-*i* ke neuron ke-*j* pada *hidden layer*

v_j = bobot yang menghubungkan z_1 dan z_2 dengan neuron pada lapisan *output*

b_1 = bobot bias yang menuju ke neuron ke-*j* pada *hidden layer*

b_2 = bobot bias yang menghubungkan *hidden layer* dengan lapisan *output*.

Yao dan Tan (2001) menyatakan bahwa pada pemilihan arsitektur jaringan, semakin kompleks tipe jaringan yang digunakan dan semakin banyak *hidden layer* dan *unit hidden* yang digunakan tidak menjamin peramalan yang dihasilkan selalu lebih baik. Oleh karena itu, tipe jaringan yang digunakan adalah tipe yang sesederhana mungkin namun dengan hasil yang seefisien mungkin.

2.8 Metode Pembelajaran ANN

2.8.1 Jenis Pembelajaran ANN

Prosedur yang digunakan ANN dalam mencari pengaturan bobot yang tepat mengacu pada paradigma belajarnya (*learning paradigm*). Paradigma belajar ini dikenal sebagai algoritma belajar dalam ANN. Berdasarkan strategi pelatihan, paradigma belajar ANN dapat diklasifikasikan menjadi dua:

1. Pembelajaran Terawasi (*Supervised Learning*)
   Metode pembelajaran pada jaringan syaraf disebut terawasi jika *output* yang diharapkan telah diketahui sebelumnya. ANN yang akan mempelajari dan menentukan pola *input* dan *output*. Proses ini berhenti jika selisih antara *output* dari ANN dan *output* yang sebenarnya sudah mencapai titik konvergen. Algoritma *supervised learning* antara lain (Setiawan, 2010):
   - Algoritma pembelajaran Hebb, biasanya digunakan sebagai dasar untuk algoritma pembelajaran yang lebih kompleks.
   - Algoritma pembelajaran delta rule, Adaline, Madaline.
   - Backpropagation (BP), terdiri dari algoritma BP standar (*gradient descent* dan *gradient descent* dengan momentum), algoritma BP dengan perbaikan menggunakan teknik *heuristic* (*resilent backpropagation*, *gradient descent* dengan *learning rate adaptif*, *gradient descent* dengan momentum dan *learning rate adaptif*), BP dengan perbaikan menggunakan optimasi numerik (algoritma-algoritma dengan *conjugate gradient*, algoritma *Quasi Newton* dan algoritma *Levenberg-Marquardt*).
2. Pembelajaran Tak Terawasi (*Unsupervised Learning*)
   Metode pembelajaran tak terawasi tidak memerlukan target *output*. Pada metode ini, tidak dapat ditentukan hasil seperti apakah yang diharapkan selama proses pembelajaran. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini sangat cocok untuk pengelompokan (*cluster*) pola. Contoh *unsupervised learning* yaitu *self organizing map*.

2.8.2 Fungsi Aktivasi

Dalam penelitian Galang (2010) dijelaskan bahwa sebelum melakukan *training*, data *input* dan target perlu dilakukan normalisasi terlebih dahulu agar dapat masuk ke dalam selang fungsi aktivasi. Normalisasi dilakukan dengan rumus:

$$X_i = \frac{x - \bar{x}}{s} \qquad (2.34)$$

di mana:

X_i : nilai data normalisasi
x : data aktual
$\bar{x}$: rata-rata data aktual
s : standard deviasi data aktual

Fungsi aktivasi adalah fungsi matematis di dalam neuron yang mengolah *input* yang masuk di dalam neuron. Fungsi aktivasi digunakan untuk memformulasikan *output* dari setiap neuron. Beberapa fungsi aktivasi yang sering digunakan pada arsitektur *feedforward neural network* yaitu:

1. Fungsi Linier (Identitas)

   Fungsi linier digunakan pada permasalahan dimana *output* yang dihasilkan adalah penjumlahan dari *input* terboboti. Pada MATLAB, fungsi ini dikenal dengan nama *purelin*. Fungsi linier dirumuskan sebagai berikut:

$$f(x) = \sum x_i v_{ij} \qquad (2.35)$$

   di mana:

   f (x) = fungsi aktivasi
   x = *input* ke i
   v_{ij} = bobot *input* ke-*i* pada node ke-*j*

2. Fungsi *Sigmoid*

   a. Sigmoid biner

   Fungsi ini digunakan untuk jaringan syaraf yang dilatih dengan menggunakan metode *backpropagation*. Fungsi *sigmoid biner* memiliki nilai pada rentang 0 sampai 1. Pada MATLAB fungsi ini dikenal dengan nama *logsig*. Fungsi *sigmoid biner* dirumuskan sebagai berikut:

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2.36)$$

   b. Sigmoid bipolar

   Fungsi *sigmoid bipolar* hampir sama dengan fungsi *sigmoid biner*, hanya saja *output* dari fungsi ini memiliki range antara 1 sampai -1. Pada MATLAB fungsi ini dikenal dengan nama *tansig*.

$$f(x) = \frac{1-e^{-x}}{1+e^{-x}} \qquad (2.37)$$

2.8.3 Inisialisasi Bobot

Inisialisasi nilai bobot awal sangat mempengaruhi jaringan saraf dalam mencapai minimum global terhadap nilai *error*, serta cepat tidaknya proses pelatihan menuju konvergen. Apabila nilai bobot awal terlalu besar, maka *input* ke setiap *hidden layer* atau lapisan *output* akan jatuh pada daerah di mana turunan fungsi sigmoidnya sangat kecil. Sebaliknya, apabila nilai bobot awal terlalu kecil, maka *input* ke setiap

hidden layer atau lapisan *output* akan sangat kecil, yang menyebabkan proses pelatihan akan berjalan sangat lambat (Warsito, 2007).

Nilai bobot akan bertambah, jika informasi yang diberikan oleh *neuron* yang bersangkutan tersampaikan, sebaliknya jika informasi tidak disampaikan oleh suatu *neuron* ke *neuron* yang lain, maka nilai bobot yang menghubungkan keduanya akan dikurangi. Pada saat pelatihan dilakukan pada *input* yang berbeda, maka nilai bobot akan diubah secara dinamis hingga mencapai suatu nilai yang cukup seimbang. Apabila nilai ini telah tercapai mengindikasikan bahwa tiap-tiap *input* telah berhubungan dengan *output* yang diharapkan (Warsito, 2007).

Pelatihan *backpropagation* dilakukan dalam rangka pengaturan bobot, sehingga pada akhir pelatihan akan diperoleh bobot-bobot yang optimal. Hal ini dapat dipenuhi dengan melakukan proses pelatihan dengan menyesuaikan parameter-parameter atau bobot dan bias koneksi sehingga untuk suatu *input* tertentu, jaringan dapat menghasilkan *output* yang sesuai dengan target. Proses penyesuaian dilakukan dengan memberikan sebuah *training set* (berupa suatu himpunan yang terdiri dari pasangan *input* dan *output*) pada jaringan. *Training set* tersebut dievaluasi dengan suatu algoritma pelatihan tertentu. Selama proses pelatihan berlangsung, *training set* akan dievaluasi berkali-kali secara iteratif untuk meminimumkan fungsi kuadrat *error*nya. Fungsi ini akan mengambil kuadrat *error* yang terjadi antara *output* dan target, dengan metode ini kesalahan lokal tiap sel dapat dilihat sebagai bagian yang terkontribusi dalam menghasilkan kesalahan total pada lapisan *output*. Apabila kesalahan pada lapisan *output* dapat dipropagasikan kembali masuk ke *hidden layer*, maka kesalahan lokal sel-sel pada lapisan tersebut dapat dihitung untuk mendapatkan perubahan bobot.

2.9 Algoritma Pelatihan *Backpropagation*

Pelatihan *backpropagation* menggunakan metode pencarian titik minimum untuk mencari bobot dengan *error* minimum dengan menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Pelatihan backpropagation meliputi 3 fase yaitu fase maju, fase mundur dan fase perubahan bobot.

Fase I : Propagasi Maju

Selama propagasi maju, sinyal *input* (x_i) dipropagasikan ke *hidden layer*. Sinyal-sinyal *input* terboboti yang masuk ke tiap-tiap unit pada *hidden layer* (z_j , j=1,2,..,p), ditentukan dengan persamaan:

$$z\_in_j = b1_j + \sum_{i=1}^{n} x_i w_{ij} \tag{2.38}$$

di mana $b1_j$ merupakan bias pada *hidden layer*. Selanjutnya sinyal *output* dari *hidden layer* ditentukan menggunakan fungsi aktivasi, yang dalam penelitian ini menggunakan sigmoid bipolar, dengan persamaan:

$$z\_out_j = f(z\_in_j) = \frac{1-e^{-(b1_j+\sum_{i=1}^{n} x_i w_{ij})}}{1+e^{-(b1_j+\sum_{i=1}^{n} x_i w_{ij})}} \tag{2.39}$$

Selanjutnya z_j dipropagasikan maju ke lapisan *output* dengan persamaan:

$$y\_in_j = b2 + \sum_{i=1}^{p} z_j v_j \tag{2.40}$$

di mana $b2$ merupakan bias pada unit *output*. Selanjutnya sinyal *output* ditentukan dengan fungsi aktivasi, yaitu fungsi identitas, dengan persamaan:

$$y\_out_j = f(y\_in) = y\_in = b2 + \sum_{i=1}^{p} z_j v_j \tag{2.41}$$

Output (y_out) dibandingkan dengan target (t_k) yang harus dicapai. Selisih $(t_j - y\_out)$ adalah *error* (δ_j). Jika *error* lebih kecil dari batas toleransi yang ditentukan, maka *epoch* dihentikan. Tapi jika *error* masih lebih besar dari batas toleransi, maka bobot setiap garis dalam jaringan akan dimodifikasi untuk mengurangi kesalahan yang terjadi.

Fase II : Propagasi Mundur

Kesalahan (*error*) yang terjadi dipropagasikan mundur, dimulai dari garis yang berhubungan langsung dengan unit pada lapisan *output*. Tiap-tiap unit *output* $(y_j, j = 1,2,3,\dots,p)$ menerima target pola yang berhubungan dengan pola *input*, *error* dihitung dengan persamaan :

$$\delta_j = (t_j - y\_out) f'(y\_in_j) \tag{2.42}$$

Kemudian koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai w_{ij}) dihitung dengan persamaan :

$$\Delta w_{ij} = \eta \delta_j z_j \quad (2.43)$$

Koreksi bias dihitung untuk memperbaharui nilai w_{ij}, dan mengirimkan δ_j ke unit-unit pada *hidden layer*. Selanjutnya tiap-tiap unit tersembunyi $(z_j, j = 1,2,3, \dots, p)$ menjumlahkan delta *input* :

$$\delta\_in_j = \sum \delta_j w_{ij} \quad (2.44)$$

Nilai (δ_in_j) dikalikan dengan turunan dari fungsi aktivasinya untuk menghitung *error*

$$\delta_j = \delta\_in_j \, f'(z\_in_j) \quad (2.45)$$

Koreksi bobot dihitung untuk memperbaharui v_j

$$\Delta v_j = \eta \delta_j z_j \quad (2.46)$$

Fase III : Perubahan Bobot

Modifikasi bobot dan bias untuk menurunkan *error* yang terjadi.

- Perubahan bobot dan bias pada garis yang menuju unit tersembunyi

$$w_{ij}(baru) = w_{ij}(lama) + \Delta w_{ij}$$

- Perubahan bobot dan bias pada garis yang menuju unit *output*.

$$v_j(baru) = v_j(lama) + \Delta v_j$$

Ketiga fase diulang-ulang hingga kondisi penghentian dipenuhi. Umumnya kondisi penghentian yang sering dipakai adalah jumlah *epoch* atau *error*. *Epoch* akan dihentikan jika jumlah *epoch* yang dilakukan sudah melebihi jumlah maksimum *epoch* yang ditetapkan, atau jika *error* yang terjadi sudah lebih kecil dari batas toleransi yang diijinkan.

2.9.1 Algoritma Quasi Newton

Metode Quasi Newton adalah salah satu metode untuk meminimumkan suatu fungsi. Metode ini menggunakan pendekatan terhadap matriks Hessian atau terhadap invers matriks Hessian tanpa harus menghitung langsung nilai dari turunan kedua yang dianggap terlalu rumit (Fallgren dalam Warsito, 2007). Warsito dan Sri (2007) dalam penelitiannya menyebutkan bahwa algoritma *Quasi Newton* memberikan hasil yang lebih baik dibandingkan algoritma *Levenberg Marquardt* dalam rangka meminimumkan kesalahan yang terjadi.

Berdasarkan Gambar 2.3, misalkan w_t adalah vektor bobot yang mengandung $w_{ij}(t)$, $b2(t)$, $v_j(t)$ dan $b1_j(t)$ konsep dasar metode newton adalah:

$$w_{t+1} = w_t - H_{t^{-1}} g_t \qquad (2.47)$$

di mana:

w = vektor bobot dan bias koneksi untuk masing-masing $w_{ij}(t)$, $b2(t)$, $v_j(t)$ dan $b1_j(t)$

$\boldsymbol{g_t}$ = vektor gradien untuk masing-masing $g_{j(t)}$, $g_{b2(t)}$, $g_{ij(t)}$, $g_{b1j(t)}$

H = matriks Hessian

Salah satu algoritma perubahan bobot dengan metode newton adalah algoritma BFGS yang diperkenalkan oleh Broyden, Fletcher, Goldfarb dan Shanno (BFGS). Algoritma pelatihan dengan metode *Quasi Newton* adalah sebagai berikut:

Langkah 0 :

a. Inisialisasi bobot awal dengan bilangan acak kecil
b. Inisialisasi *epoch* 0, MSE ≠ 0
c. Inisialisasi H_0 yang merupakan matrik definit posotif simetrik.
d. Tetapkan maksimum *epoch* dan target *error*.

Langkah 1 :

Jika kondisi penghentian belum terpenuhi (*epoch* < maksimum *epoch* atau MSE > target *error*), lakukan langkah berikutnya.

Langkah 2 :

Unit *output* Y menerima target pola yang berhubungan dengan pola *input* pelatihan. Kesalahan pada unit *output* didefinisikan sebagai:

$$e = t - y$$

di mana:

e = kesalahan pada unit *output*
t = keluaran yang diinginkan (acuan/target)
y = keluaran aktual

Fungsi jumlah kuadrat *error* didefinisikan dengan:

$$E = \frac{1}{2}(t - y)^2$$

Misalkan w_t adalah vektor bobot yang mengandung $w_j(t)$, $b2(t)$, $v_{ij}(t)$,

dan $b1_j(t)$. Gradien fungsi kinerja terhadap nilai bobot dan bias koneksi pada waktu ke-t didefinisikan dengan:

$$g_{j(t)} = -\delta_k(t).z_j(t)$$
$$g_{b2(t)} = -\delta_k(t)$$
$$g_{ij(t)} = -\gamma_j(t)x_i$$
$$g_{b1j(t)} = -\gamma_j(t)$$

g_j merupakan vektor gradien untuk masing-masing $g_{j(t)}$, $g_{b2(t)}$, $g_{ij(t)}$ dan $g_{b1j(t)}$. Jika $g_t = 0$ maka algoritma berhenti, jika tidak maka hitung:

$$\boldsymbol{d_t} = -\boldsymbol{H_t}.\boldsymbol{g_t} \qquad (2.48)$$

Langkah 3 :
Tempatkan nilai α_t dengan menggunakan fungsi *line search*. Tujuannya adalah untuk meminimumkan *error* yang akan terjadi

$$\alpha_t = \arg min_\alpha [f(\boldsymbol{w_t} + \alpha_t \boldsymbol{d_t})]$$

Perubahan vektor bobot dan bias yang terjadi adalah :

$$\boldsymbol{w_{t+1}} = \boldsymbol{w_t} + \alpha_t.\boldsymbol{d_t} \qquad (2.49)$$

Langkah 4 :
Hitung perubahan bobot dan bias dengan persamaan berikut :

$$\Delta\boldsymbol{w_t} = \alpha_t.\boldsymbol{d_t} \qquad (2.50)$$

Sedangkan perubahan arah pencarian adalah sebagai berikut :

$$\Delta\boldsymbol{g_t} = \boldsymbol{g_{t+1}} - \boldsymbol{g_t}$$

Maka didapatkan persamaan :

$$\boldsymbol{H_{t+1}} = \boldsymbol{H_t} + \left(1 + \frac{\Delta g_t{}^T.H_t.\Delta g_t}{\Delta g_t{}^T.\Delta w_t}\right).\frac{\Delta w_t.\Delta w_t{}^T}{\Delta w_t{}^T.\Delta g_t} - \frac{H_t.\Delta g_t.\Delta w_t{}^T + \left(H_t.\Delta g_t.\Delta w_t{}^T\right)^T}{\Delta g_t{}^T.\Delta w_t} \qquad (2.51)$$

Langkah 5 :
epoch = *epoch* + 1.
Proses berhenti jika nilai *epoch* = *epoch* + 1, jika belum mencapai nilai tersebut maka perhitungan dilakukan kembali dari langkah 2.

2.10 Pemodelan Neuro-GARCH

Tahapan memodelkan Neuro-GARCH meliputi hal-hal berikut:

1. Diketahui sejumlah data *return* (Z_t) deret waktu $Z_1, Z_2, \ldots, Z_t$ selanjutnya digunakan untuk memperkirakan nilai Z_{t+1}.
2. Tahapan pemodelan GARCH adalah sebagai berikut:
   a. Data *return* dimodelkan menggunakan ARIMA untuk mengetahui kestasioneran ragamnya.
   b. Jika ragam data tidak stasioner, maka data dimodelkan menggunakan model GARCH.
   c. Setelah model GARCH terbentuk, penelitian dilanjutkan dengan memodelkan kuadrat sisaan dan ragam sisaan model GARCH sebagai *input* dari ANN.
3. Proses pemodelan Neuro-GARCH adalah sebagai berikut:
   Menentukan *input* dan target dari model GARCH (1,1). Misalkan model yang terbentuk yaitu GARCH (1,1), maka *input* dan target untuk model Neuro-GARCH yaitu

$$Z_t = \phi Z_{t-1} + \varepsilon_t \,, \quad \varepsilon_t \sim N(0, \sigma_t^2)$$
$$\sigma_t^2 = \alpha_0 + \beta_1 \sigma_{t-1}^2 + \alpha_1 \varepsilon_{t-1}^2$$

   maka *input* untuk data *return* yaitu Z_{t-1} dan ε_t dengan target yaitu Z_t. Sedangkan *input* untuk data volatilitas yaitu σ_{t-1}^2 dan ε_{t-1}^2, dengan target yaitu σ_t^2.
3. Proses analisis menggunakan ANN selanjutnya akan dijelaskan pada subbab 3.2.3.
4. Didapatkan model Neuro-GARCH sebagai berikut:

$$\hat{Z}_t = f^0\left(\sum_{j=1}^{q} v_j^o \, f_j^h\left(\sum_{i=1}^{p} w_{j,i}^h \, Z_{t-1} + w_{j,i}^h \, \varepsilon_t + b_j^h\right) + b^o\right) \tag{2.52}$$

$$\hat{\sigma}_t^2 = f^0\left(\sum_{j=1}^{q} v_j^o \, f_j^h\left(\sum_{i=1}^{p} w_{j,i}^h \, \sigma_{t-1}^2 + w_{j,i}^h \, \varepsilon_{t-1}^2 + b_j^h\right) + b^o\right) \tag{2.53}$$

di mana:

$\hat{Z}_t$ = nilai dugaan dari *mean* model
$\hat{\sigma}_t^2$ = nilai dugaan dari *varian* model
f^o = fungsi aktivasi pada *neuron* di lapisan *output*
v_j^o = bobot dari *neuron* ke-*j* di lapisan tersembunyi menuju

neuron pada lapisan *output*

f_j^h = fungsi aktivasi di *neuron* ke-*j* pada lapisan tersembunyi

$w_{j,i}^h$ = bobot dari *input* ke-i yang menuju neuron ke-*j* pada lapisan tersembunyi, (*j*=1,2,...,*q*)

Z_{t-1} = *input* pertama untuk model rataan GARCH

ε_t = *input* kedua untuk model rataan GARCH

σ_{t-1}^2 = *input* pertama untuk model ragam GARCH

ε_{t-1}^2 = *input* kedua untuk model ragam GARCH

b^h = bias pada *neuron* di lapisan tersembunyi

b^o = bias pada *neuron* di lapisan *output*.

5. Pemilihan model Neuro-GARCH terbaik berdasarkan nilai MAD dan MSE *testing* terkecil.

2.11 Kriteria Pemilihan Model Terbaik

2.11.1 *Mean Absolute Deviation* (MAD)

MAD merupakan ukuran pertama kesalahan peramalan keseluruhan untuk sebuah model. Nilai ini dihitung dengan mengambil jumlah nilai absolut dari tiap kesalahan peramalan dibagi dengan jumlah periode data.

$$MAD = \frac{1}{n}\sum_{i=1}^{n}\left|£_t\right| \qquad (2.54)$$

di mana :

ε_t = $Z_t - \hat{Z}_t$

Z_t = nilai aktual pada periode ke-t

$\hat{Z}_t$ = nilai peramalan pada periode ke-t

n = banyaknya data

2.11.2 *Mean Square Error* (MSE)

Makridakis, dkk (1999) menyatakan bahwa model cukup layak digunakan sebagai peramalan jika memiliki kriteria pemilihan model yang tinggi yang ditunjukkan oleh MSE terkecil. Patterson (1996) menyatakan penggunaan MSE untuk berbagai aplikasi ANN dalam pemodelan dan peramalan deret waktu sudah cukup memenuhi dan paling populer digunakan. Semakin kecil nilai MSE, maka model semakin baik.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}£_t^{\,2} \qquad (2.55)$$