# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Indeks harga saham gabungan (IHSG) merupakan salah satu indeks pasar saham yang digunakan oleh bursa efek Indonesia (BEI). IHSG menunjukan pergerakan harga saham secara umum yang tercatat di bursa efek. IHSG dapat digunakan untuk menilai situasi pasar secara umum atau mengukur apakah harga saham mengalami kenaikan atau penurunan. Indeks ini diperkenalkan pertama kali pada tanggal 1 April 1983. Dasar perhitungan IHSG adalah jumlah nilai pasar dari total saham yang tercatat pada tanggal 10 Agustus 1982. Jumlah nilai pasar sendiri merupakan total perkalian setiap saham tercatat (kecuali untuk perusahaan yang berada dalam program restrukturisasi) dengan harga di BEI pada hari tersebut. Perhitungan indeks merepresentasikan pergerakan harga saham di bursa yang terjadi melalui sistem perdagangan lelang (Anonim, 2008).

Harga-harga saham bergerak dalam hitungan detik dan menit, maka nilai indeks pun bergerak turun naik dalam hitungan waktu yang cepat pula. Untuk mengantisipasi perubahan harga saham yang nantinya akan berdampak pada perubahan IHSG tersebut maka diperlukan analisis saham. Terdapat dua pendekatan yang sering dilakukan untuk menganalisis harga saham, yaitu analisis fundamental dan analisis teknikal. Analisis fundamental pada dasarnya adalah melakukan analisis historis atas kekuatan keuangan, proses ini sering juga disebut sebagai analisis perusahaan. Sementara itu analisis teknikal lebih menitikberatkan pergerakan harga saham yang terjadi di bursa pada kurun waktu tertentu. Dalam analisis teknikal pendekatan dilakukan dengan menggunakan metode-metode peramalan, (Sekuritas, 2006). Data IHSG memiliki tingkat fluktuatif yang besar sehingga data ini cenderung mempunyai perubahan variansi yang tinggi. Salah satu metode yang sesuai untuk meramalkan data jenis ini adalah *generalized autoregressive conditional heteroscedasticity* (GARCH).

*Generalized autoregressive conditional heterocedasticity* (GARCH) merupakan perkembangan dari model *autoregressive conditional*



*heterocedasticity* (ARCH). Model ARCH kali pertama diperkenalkan oleh Engle pada tahun 1982 dengan memperkenalkan konsep *conditional heterocedastic*, sebuah konsep tentang ketidak-konstanan variansi dari data acak, dan perubahan variansi ini dipengaruhi oleh data acak sebelumnya yang tersusun dalam urutan waktu. Dalam model GARCH, perubahan variansinya, selain dipengaruhi oleh beberapa data acak sebelumnya, juga dipengaruhi oleh sejumlah variansi dari data acak sebelumnya. Model ini dianggap cukup sesuai dalam memodelkan data dengan perubahan variansi yang cukup tinggi, (Surya dan Hariyadi, 2003). Selain GARCH, metode lain yang juga dapat diterapkan adalah jaringan syaraf tiruan dengan *backpropagation.*

Jaringan saraf tiruan (JST) merupakan salah satu sistem pemrosesan informasi yang didesain dengan menirukan cara kerja otak manusia dalam menyelesaikan suatu masalah melalui proses belajar dengan perubahan bobot sinapsisnya. JST mengidentifikasi pola data dari IHSG dengan metode pendekatan pembelajaran atau pelatihan yaitu untuk menentukan bobot penghubung antar simpul yang optimum. Algoritma pelatihan *backpropagation* pertama kali dirumuskan oleh Werbos dan dipopulerkan oleh Rumelhart dan McClelland untuk

dipakai pada JST. Algoritma ini termasuk metode pelatihan *supervised* (terbimbing) dan didesain untuk operasi pada jaringan *feed forward* multi lapis. Metode *backpropagation* ini telah banyak diaplikasikan secara luas, diantaranya di bidang *financial*, pengenalan pola tulisan, pengenalan pola suara, sistem kendali, pengolahan citra medika dan masih banyak lagi, (Martiana, 2007).
Metode pembelajaran *backpropagation* yang diterapkan dalam skripsi ini ada tiga, yaitu algoritma *gradient descent* dengan momentum (GDM), algoritma *gradient descent* dengan *adaptive learning rate* dan momentum (GDX), dan algoritma *levenberg-marquadt* (LM). Untuk fungsi aktivasi yang digunakan terbatas pada fungsi aktivasi identitas (*pureline*) dan *sigmoid biner (logsig)*.
Hasil peramalan dari kedua metode tersebut (GARCH dan JST*backpropagation*) akan dibandingkan menggunakan kaidah nilai *mean absolute percentage error* (MAPE) dan *mean square error* (MSE). Tujuannya adalah untuk mengetahui hasil ramalan IHSG dengan metode mana yang memiliki

keakuratan ramalan yang lebih baik (lebih mendekati nilai sebenarnya). Data IHSG yang digunakan dalam skripsi ini adalah data IHSG saat penutupan harian pada *trading days* (hari kerja) yang diambil dari bursa efek Indonesia (BEI) sejak 2 Januari 2004 sampai 31 Desember 2008.

### 1.2 Perumusan Masalah

Berdasarkan latar belakang masalah, disusun perumusan permasalahan sebagai berikut
a. bagaimanakah model GARCH yang sesuai untuk meramalkan data harga saham IHSG
b. bagaimanakah struktur jaringan syaraf tiruan *backpropagation* yang baik untuk meramalkan data IHSG
c. seberapa besar keakuratan peramalan IHSG yang dihasilkan oleh model GARCH jika dibandingkan dengan hasil peramalan menggunakan jaringan syaraf tiruan *backpropagation.*

### 1.3 Tujuan

Tujuan dari skripsi ini adalah
1. menentukan model GARCH yang sesuai dengan data IHSG
2. memperoleh struktur jaringan syaraf tiruan *backpropagation* untuk meramalkan data IHSG
3. mendapatkan hasil peramalan data IHSG untuk 1 bulan ke depan yaitu bulan Januari 2009 (19 langkah peramalan) baik menggunakan GARCH maupun jaringan syaraf tiruan**.**

### 1.4 Manfaat

Manfaat dari skripsi ini adalah
1. hasil peramalan IHSG yang diperoleh dapat menjadi wacana dalam mengindikasikan kondisi pasar saham di bursa pada periode selanjutnya
2. menambah wawasan dan pengetahuan mengenai jenis teknik peramalan indeks keuangan dan langkah-langkah yang diperlukan.


## BAB II
## LANDASAN TEORI

### 2.1. Log *Return* dan Fluktuasi Harga

Dalam analisis *financial time series* (data runtun waktu keuangan), yang menjadi pusat perhatian adalah fluktuasi harga yang terjadi. Pada dasarnya

fluktuasi harga merupakan variabel yang menunjukkan naik turunnya harga sebagai dampak dari mekanisme pasar yang ada. Secara umum, fluktuasi harga dapat didefinisikan sebagai perubahan harga terhadap waktu *t* ____ atau dapat dituliskan sebagai
__ , dengan _ adalah harga penutupan IHSG harian pada waktu *t*.
Lebih jauh lagi, pendekatan untuk fluktuasi harga adalah perubahan relatif atau *return* yang sering didefinisikan sebagai return penyusun kontinu (*continously compounded return*) atau log *return*, yaitu

Besaran inilah yang saat ini sering digunakan dalam berbagai analisis lanjut dari *financial time series* yang pada praktiknya jumlahnya sangat besar. Dengan menggunakan parameter ini tentu akan lebih mudah untuk menganalisisnya. Selain itu log *return* sendiri juga bermanfaat untuk menjadikan data stasioner terhadap rata-rata, (Tsay, 2002).

**2.2. Volatilitas**

Volatilitas merupakan sebuah terminologi kepekaan (sensitifitas) sebuah data runtun waktu keuangan. Biasanya besaran ini dinyatakan sebagai standar deviasi dari laju perubahan data runtun waktu keuangan. Pendek kata, volatilitas merupakan ukuran dari ketidakpastian dari data runtun waktu keuangan atau resiko yang mungkin dihadapi investor dalam perdagangan di bursa, (Yohanes dan Situngkir, 2003). Secara sederhana Castiglione (2001) menyatakan volatilitas sebagai harga mutlak dari nilai *return* __ .
5

**2.3. Fungsi Autokorelasi *(Autocorrelation Function*, ACF)**

Kovariansi antara runtun waktu saat _ yaitu _, dengan runtun waktu saat _ _ _ yaitu ___, didefinisikan sebagai
,
yang diestimasi oleh
dengan _ adalah nilai *lag*. Autokorelasi merupakan korelasi antara observasi saat _, _, dengan observasi saat _ _ _, ___, yang terpisah oleh *lag* _ dan didefinisikan sebagai
Autokorelasi diestimasi oleh

_ 0 _ _ _1__ _____ _ $_ _ %_&__'___
dengan _ adalah observasi dari suatu runtun waktu pada waktu _ dan _2 adalah rata-rata dari deret runtun waktu. Himpunan bersama dari (_, 3_()4 __ _ _ %_ &5, untuk berbagai *lag* _ disebut fungsi autokorelasi.
Menurut Pankartz (1983), jika suatu runtun waktu dengan rata-rata stasioner maka estimasi nilai dari ACF turun secara cepat mendekati nol dengan semakin bertambahnya *lag*, tetapi jika rata-ratanya tidak stasioner maka estimasi nilai dari ACF turun secara perlahan mendekati nol.
Uji untuk mengetahui apakah beberapa autokorelasi dalam runtun waktu _ adalah adalah signifikan dapat dilakukan menggunakan uji *Ljung-Box*.
Hipotesis dalam uji *Ljung-Box* adalah
(i). H0: ( _ (% __& _ (6 _ $_(tidak terdapat autokorelasi dalam data runtun waktu),
H1: (7 8 $ untuk paling tidak sebuah 7____7 _ _%_ &__ 6 (terdapat autokorelasi dalam dalam data runtun waktu),
(ii). menghitung statistik uji *Ljung-Box*
9 _ "_" _ %_:
(.)
_
"
; _ _
)0
6
dengan " adalah jumlah data, _ adalah nilai *lag* dan 6 adalah jumlah *lag* maksimum yang ingin diuji,
(iii). statistik uji di atas berdistribusi *Chi-Squared* dengan derajat bebas 6,
(iv). H0 ditolak jika 9 < =;
_.

## 2.4. Fungsi Autokorelasi Parsial (*Partial Autocorrelation Function*, PACF)

Autokorelasi parsial pada *lag* _ dapat dipandang sebagai korelasi antara observasi _ dan ___ setelah menghilangkan hubungan linear dari ___ ___%_&_____ (Wei, 1990). Autokorelasi parsial dari 3_5 pada *lag* _ didefinisikan sebagai
>__ ___ ?
@
@
@
A
_
_( _ _(%_ __& (__% (
(
B _
_
( ____& (__C (%
_____ _____ _____ B
(__ (__% (__C & ( ____(_D
E
E
E
F
? @ @ @ A
_
_( _ _____(%_ _& (__% (__
(
B _
_ _____( _& (__C (__%
____ ____ _____ B

(__ (__%(__C & ( _____ D
E
E
E
F
___ (2.1)

Himpunan bersama dari >__, 3_>))4 ___ _ _ %_ &5, disebut sebagai fungsi autokorelasi parsial. Autokorelasi parsial diestimasi oleh >G __ dengan setiap (H pada persamaan (2.1) diganti oleh autokorelasi sampel, (IH, yang bersesuaian.

**2.5. Model Runtun Waktu dengan Heteroskedastisitas**

Model runtun waktu dapat dibagi berdasarkan ketidaksamaan variansi dari residual satu pengamatan ke pengamatan lainnya. Model dengan kesamaan variansi (variansi konstan) disebut homoskedastisitas. Sebaliknya jika variansinya tidak sama menjadi model runtun waktu dengan heterokedastisitas. Pada skripsi ini, digunakan salah satu model runtun waktu dengan heteroskedastisitas yaitu *Generalized Autoregressive Conditional Model* (GARCH).

**2.5.1. Model *Generalized Autoregressive Conditional Heterocedasticity (GARCH)***

Model GARCH diperkenalkan pertama kali oleh *Bollerslev (1986)*, dimana model ini adalah perkembangan dari model ARCH. Perbedaannya adalah 7
dalam model GARCH, perubahan variansinya, selain dipengaruhi oleh beberapa data acak sebelumnya, juga dipengaruhi oleh sejumlah variansi dari data acak sebelumnya.

Diberikan 3J_5 merupakan sekumpulan data deret waktu keuangan dengan data sejarah K_ , dan 3L_5 adalah proses white noise yang berdistribusi normal sehingga proses J_ didefinisikan sebagai

J_ _ L_ __M_
_.

Proses J_ disebut sebagai GARCH(1,1) jika

J_ _N_K_ _O_P__$_ M_
_ _
M_
_ __Q-_ __Q_J_
_ ___R_M_
_ ,

dengan syarat Q-_ _> 0, Q _ 0, R S , _Q _ R_ T_1 .

**Model GARCH(*p,q*)**

Untuk orde yang lebih tinggi yaitu *p* dan *q* maka model GARCH(1,1) menjadi GARCH(*p,q*) sebagai berikut

M_
_ ___Q-_ __Q_J_
_ __&_ Q;_J_ ;
_ __R_M_
_ __&_ RU_M_U
_
__Q-_ __# QV_J_ V
_ _ # RW_M_ W
_ U
W 0
;V

0
dengan syarat
Q-_> 0, Qv _ 0, Rw _ 0, untuk 7 = _&_ 6 ; H _ _&_ X
dan # _ YZ[__;_U_
v0 Qv _ Rw_ T .
Syarat Q-_> 0, Qv_ \ 0 dan Rw _ 0 dibutuhkan untuk memastikan proses memiliki variansi bersyarat positif. Sedang syarat # _ YZ[__;_U_
v0 Qv _ Rw_ T
menyatakan bahwa J_ memiliki variansi tidak bersyarat yang berhingga.

**2.5.2. Pengujian Eksistensi Model**

Meskipun dalam *financial time series* sering mengandung efek ARCH dan GARCH, namun tidak menutup kemungkinan efek tersebut tidak ada atau tidak signifikan pada runtun waktu tertentu. Oleh karena itu, perlu dilakukan pemeriksaan eksistensi efek ARCH dan GARCH dalam data runtun waktu. Menurut Warsito dkk. (2006), pemeriksaan eksistensi model ARCH ada dua cara yaitu
8
1. memeriksa fungsi yang mengandung efek ARCH. Residunya tidak berkorelasi tetapi kuadrat residunya menunjukkan adanya korelasi yang ditandai dengan nilai-nilai ACF yang signifikan tidak sama dengan nol. Langkah-langkah yang dilakukan adalah
a. mengestimasi _3_
_5 menggunakan model ARMA atau model regresi biasa sehingga diperoleh model terbaik. Dari model tersebut diperoleh kuadrat residu (J_
_) dan dihitung variansi sampel dari residu (M._) yang didefinisikan sebagai
M._ _
# J_
_ /
_0
"

b. menghitung dan membuat plot ACF sampel dari kuadrat residu, nilai ACF tersebut dihitung dengan rumus
(.)

_ _
# _J_
_ _ M._ __J_ )
_ _ M._ _ /
_ 0)*
# _J_
_ _ M._ _ /
_ 0
c. untuk ukuran sampel yang besar, deviasi standar () dapat didekati dengan "!_. Nilai () yang signifikan tidak sama dengan nol mengidentifikasi adanya efek ARCH. Statistik *Ljung-Box* dapat digunakan untuk menguji signifikasi dari koefisien. Bentuk hipotesis-nya adalah
(i). H0 : ( _ (_ _ ] _ (; _ $ (J_
_ tidak memiliki autokorelasi atau

tidak ada efek ARCH)
H0 : paling sedikit satu ( 8 $, _ _ _%_&_ 6 (J_
_ memiliki
autokorelasi atau terdapat efek ARCH)
(ii). statistik uji 9 dari *Ljung-Box* adalah
9 _ "_" _ %_ # _(.)
; _!_" _ ___
)0_
(iii). H0 ditolak jika 9 < ^;
_.
*2.* Uji *Lagrange Multiplier*
Langkah-langkah dalam uji ini adalah
a. menentukan persamaan yang paling sesuai untuk data runtun waktu, dari persamaan tersebut diperoleh kuadrat residu (J_
_)
b. meregresikan J_
_ pada konstata dan 6 *lag*-nya sendiri
9
J_
_ _ Q- _ QJ_
_ __&__ Q;J_ ;
_ _
c. menguji hipotesis dengan
(i). H0 : Q _ Q_ __& _ Q; (tidak ada efek ARCH sampai *lag* 6).
H1 : paling sedikit terdapat satu Q) 8 $, _ _ _%_&_ 6
(ii). menggunakan asumsi normalitas, statisitik uji yang digunakan adalah
_` _ ab_
dengan a adalah banyaknya residu dan b_ adalah ukuran kecocokan data dengan model
(iii). H0 ditolak jika _` < ^;
_ .

**2.5.3. Estimasi Parameter**

Dalam pembahasan estimasi parameter _0, _1, dan _1 dalam model GARCH(p,q) ini, dimisalkan model regresi dari GARCH sedemikian hingga
c_ __ _- ____d_ __e_____ _ _ %_&_ "
J_ __L_ fM_
_
M_
_ __Q-_ __Q_J_
_ __&_ Q;_J_ ;
_ __R_M_
_ __&_ RU_M_U
_ .
Sehingga terdapat vektor parameter gh sebagai berikut
gh _ __-_ __ Q-_ Q_&_ Qi_ R_&__ Rj_k _ l_.k_ mnop_
dengan
mn __ q
Q-

Q
R
r _____s,t_____. __ u
_-
_
v_
Untuk mengestimasi parameter dari variansi _0, _1, dan _1 terlebih dahulu dicari gradien dari fungsi likelihood. Menggunakan asumsi normalitas, fungsi densitas probabilitas dari J_N_K_ adalah
w_J_N_M_
_ – –

_%xM_
_
_yz{ _|_
J_
_
%M_
_} _
Misalkan $L_T$ menyatakan fungsi likelihood untuk observasi ke-$T$ dan ukuran sampel dinyatakan dari $t$ sampai dengan $T$ maka
10
~_ _ __ _w_J_N_M_
_ –
– –

%
|__ %x _ __ M_
_ __
J_
_
%M_
_}_
Turunan terhadap parameter mn adalah
€/
mn __
€/
M_
_ __
M_
_
mn
__|__

%M_
_ __
J_
_
%_M_
_}_

M_
_
mn
_ __

%M_
_ _
M_
_
mn __
J_
_
%_M_
_ _ __
M_
_
mn
__

%M_
_ _
M_
_
mn _
J_
_
M_
_ _ _ .
__|

%M_
_}_._ G
_ __ _ _

Sehingga dari metode Newton-Raphson yang telah dimodifikasi dapat diperoleh bentuk iterasi sebagai berikut

mG
v* __ mG
v _ u# _
_ _.
_..
. ._ .
_. .
_..
. ._ .
__
.
k_
/_
0 v

#
_..

. ._ .
__
__
/_
0 _ 2.2
Persamaan 2.2 dapat disajikan sebagai matriks
mG
V* __mG V _ _.k._ .k.
dengan
. _ .

B _
/___
. _

.

~
Q-
~
Q
&
~
Qi
~
R
&
~
Rj
~_
Q-
B
~_
Q
B
&
~_
Qi
B
~_
R
B
&
~_
Rj
B
~/
Q-
~/
Q
&
~/
Qi

~/
R
&
~/
Rj.
. . . . .
_

.
.
.M
% ._..
.
M
% _ .. ]
.
.M
% _.-..
._ .-..
.
M
% _ .. .
.M
% _M$
% _ .-..
.
M
% _ .. ]
.
.M
% _Mt_
% _ .-..
.
M
% _ ..
.
.M%
% . ..
.
M%
% _ ..
B
]
.
.M%
% _.-..
._ .-..
.
M%
% _ ..
B
.
.M%
% _M
% _ .-..
.
M%
% _ ..
B
]
.
.M%
% _Mt_%
% _ .-..
.
M%
% _ ..
B

B
.
.M"
% ._..
.
M"
% _ .. ]
.
.M"
% l._..
.p ._..
.
M"
% _ .. .
.M"
% _M"_
% _ ._..
.
M"
% _ .. ]
.
.M"
% lMt__"_ _
% p ._..
.
M"
% _ ..
.
. . . . .
dan
11
_ __ .
B
.
Bollerslev (1986) mengusulkan nilai awal dari variansi _M-
_ dan
kuadrat residu _J-
_ agar diestimasi menggunakan a_ _ .
/.# M
_
_ . Nilai awal Q-
dan Q dapat diperoleh dengan meregresikan J_
_ dengan J_
_ , untuk Rdiberi nilai
awal nol (Warsito dkk., 2006).

**2.5.4. Pemilihan Model Terbaik**

Menurut Kutlar dan Dönek (2008), model heteroskedastisitas terbaik dapat dipilih berdasarkan nilai *akaike info criterion* (AIC). AIC dirumuskan sebagai

.. _ _ _%_ .
~
"
. _ % .
_
"
.

dengan ~ adalah fungsi log *likelihood*, _ adalah jumlah parameter yang diestimasi dan " adalah jumlah observasi.

## 2.6. Jaringan Syaraf Tiruan

Jaringan syaraf tiruan merupakan turunan ilmu dari kecerdasan buatan atau *artificial intelegence.* Pada sub bab ini akan dijelaskan terkait konsep dasar, definisi, komponen, arsitektur, bias dan fungsi aktivasinya.

### 2.5.1. Konsep Dasar dan Definisi Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) merupakan suatu sistem pengolahan informasi secara komputasi yang mampu menirukan jaringan *neuron* (syaraf) biologis manusia. Jaringan ini dikembangkan berdasarkan asumsi sebagai berikut

a. jaringan ini tersusun atas elemen-elemen pemroses yang disebut *neuron* tiruan

b. *neuron-neuron* tiruan tersebut saling berhubungan dalam satu jalinan koneksi

c. pada setiap jalinan koneksi yang dilakukan akan membentuk suatu bobot tertentu yang mengalihkan sinyal yang ditransmisikan



d. menentukan fungsi aktivasi pada setiap *neuron*nya untuk menentukan sinyal keluaran *neuron* tersebut

e. memiliki kemampuan belajar dan memperbaiki diri melalui suatu proses pelatihan.

Ciri elemen pemroses jaringan syaraf tiruan yang berasal dari sifat *neuron* biologis adalah sebagai berikut

a. elemen pemroses menerima banyak sinyal masukan, yang mana pemrosesan informasinya dilakukan secara lokal tatapi dapat mempengaruhi kendali proses secara keseluruhan

b. sinyal dapat dimodifikasi dengan bobot pada sinapsis penerima, kekuatan dari sinapsis diperoleh dari pengalaman

c. elemen-elemen pemrosesan menjumlahkan masukan bobot

d. dengan masukan yang cukup besar, *neuron* mentransmisikan sinyal keluaran

e. keluaran dari suatu *neuron* dapat menuju ke banyak *neuron* lain (*neuron* cabang).

Selain itu, karakteristik *neuron* tiruan mirip dengan *neuron* biologis yang toleran terhadap adanya kesalahan yang terjadi pada masukan dan pada kerusakan sistem. Hal ini disebabkan adanya proses pelatihan yang dilakukan secara kontinu untuk jenis kasus kesalahan yang besar pada saat perancangan jaringan, (Martiana, 2007).

Menurut *Hecht-Nielsend (1988)*, "Jaringan syaraf tiruan adalah suatu struktur pemroses informasi yang terdistribusi dan bekerja secara paralel, yang terdiri atas elemen pemroses (yang memiliki memori lokal dan beroperasi dengan informasi lokal) yang diinterkoneksi bersama dengan alur sinyal searah yang disebut koneksi. Setiap elemen pemroses memiliki koneksi keluaran tunggal yang bercabang (*fan out*) ke sejumlah koneksi kolateral yang diinginkan (setiap koneksi membawa sinyal yang sama dari keluaran elemen pemroses tersebut). Keluaran dari elemen pemroses tersebut dapat merupakan sembarang jenis persamaan matematis yang diinginkan. Seluruh proses yang berlangsung pada setiap elemen harus benar-benar dilakukan secara lokal, yaitu keluaran hanya bergantung pada nilai masukan pada saat itu yang diperole
tersimpan dalam memori lokal".

### 2.5.2.

Ada beberapa tipe jaringan syaraf tiruan, namun demikian hampir semua memiliki komponen dasar yang sama. Seperti halnya otak manusia,

jaringan syaraf juga terd
*neuron-neuron* tersebut.
informasi yang diterima melalui sambungan keluarnya menuju ke
yang lain. Pada jaringan syaraf tiruan, hubungan ini dikena
Informasi yang diterima
Gambar 2.1 menunjukkan struktur
Gambar 2.1 Struktur Jaringan Syaraf
Melihat pada
sel *neruon* biologis. *Neuron*
sama pula dengan *neuron*
*neuron* dengan bobot masukan tertentu. Input ini akan diproses oleh sua
perambatan yang akan menjumlahkan semua nil
penjumlahan ini kemudian akan dibandingkan dengan suatu nilai ambang
(*threshold*) tertentu melalui fungsi aktivasi setiap
melewati suatu nilai ambang tertentu, maka
sebaliknya maka *neuron*
diaktifkan, maka *neuron*
outputnya ke semua *neuron*
Pada jaringan syaraf tiruan,
lapisan-lapisan yang disebut dengan lapisan
pada satu lapisan akan dihubungkan dengan lapisan
sesudahnya. Informasi yang diberikan pada jaringan syaraf tiruan akan
diperoleh melalui koneksi dan nilai yang

**Komponen Jaringan Syaraf Tiruan**

terdiri dari beberapa *neuron*, dan ada hubungan antara
*Neuron-neuron* tersebut akan mentransformasikan
*neuron*
dikenal dengan nama bobot.
disimpan pada suatu nilai tertentu pada bobot tersebut.
*neuron* pada jaringan syaraf tiruan.
Tiruan
Gambar 2.1, *neuron* buatan ini sebenarnya mirip dengan
*Neuron-neuron* buatan tersebut bekerja dengan cara yang
*neuron-neuron* biologis. Informasi (input) akan dikirim ke
nilai-nilai bobot yang masuk. Hasil
) *neuron*. Apabila suatu input
*neuron* akan diaktifkan, tapi jika
tersebut tidak akan diaktifkan. Apabila *neuron*
tersebut akan mengirimkan output melalui bobot
yang berhubungan dengannya. Demikian seterusnya.
*neuron-neuron* akan dikumpulkan dalam
*neuron*. Biasanya *neuron*
lapisan-lapisan sebelum dan



h , *neuron-neuron*
l suatu fungsi
. tersebut
bobot-bobot
engannya. *neuron-neuron*

dirambatkan dari lapisan ke lapisan, mulai dari lapisan input sampai ke lapisan output melalui lapisan yang lainnya
tersembunyi. Lapisan tersembunyi
lapisan) tapi untuk lapisan tunggal hanya terdapat satu lapisan yaitu satu buah output.

**2.5.3.**

Faktor terpenting untuk menentukan kelakuan suatu
fungsi aktivasi dan pola bobotnya. Ada beberapa ar
yaitu dijelaskan sebagai berikut.

**a. Jaringan dengan Lapisan Tunggal (**

Jaringan dengan lapisan tung
dengan bobot-bobot terhubung. Jaringan ini hanya menerima input kemudian secara langsung akan mengolahnya menjadi output tanpa melalui lapisan tersembunyi.

Gambar 2.2 Jaringan dengan

Pada G
yaitu $X_1$, $X_2$, dan X
dan $Y_2$. *Neuron-neuron*
besar hubungan antara dua buah
bersesuaian.

**b. Jaringan dengan Lapisan Jamak**

Jaringan dengan banyak lapisan memiliki satu atau lebih lapisan
yang terletak diantara lapisan input dan output (memiliki satu atau lebih lainnya, yang sering dikenal dengan nama lapisan
hanya ada pada jaringan *multi layer*

**Arsitektur Jaringan Syaraf Tiruan**

*neuron*
arsitektur jaringan syaraf tiruan,

***Single Layer*)**

tunggal hanya memiliki satu lapisan
*Single Layer*
Gambar 2.2 tersebut lapisan input memiliki tiga
, $X_3$. Pada lapisan output memiliki dua buah *neuron*
pada kedua lapisan saling berhubungan. Seberapa
*neuron* ditentukan oleh besarnya bobot yang

***(Multi Layer)***



, (banyak
adalah
itektur gal ambar *neuron*,
yaitu $Y_1$
esarnya
lapisan tersembunyi).
Umumnya ada lapisan bobot
saling bersebelahan. Jaringan dengan
permasalahan yang lebih sulit dari pada jaringan dengan
saja dengan pembelajaran yang lebih rumit. Namun demikian, pada banyak kasus, pembelajaran dengan jaringan
menyelesaikan masalah.

Gambar 2.3 Jaringan dengan
Dalam JST sering ditambahkan sebuah unit dengan masukan yang
nilainya selalu=1. Unit yang sedemikian itu disebut
sebagai sebuah masukan yang nilainya=1 (Siang, J.J. : 1005)
Pada setiap
aktivasi. Fungsi ini adalah fungsi umum yang akan digunakan untuk membawa
input menuju output yang diinginkan. Fungsi aktivasi inilah yang akan
menentukan besarnya bobot. Contoh dari fungsi aktivasi ini antara l

**1. *Linier / Pureline***

Fungsi linier akan membawa input ke output yang sebanding.
digambarkan sebagai berikut
Untuk lebih jelasnya bisa dilihat pada Gambar 2.3
bobot-bobot yang terletak diantara dua lapisan yang
*multi layer* ini dapat menyelesaikan
*single layer*
*multi layer* ini lebih sukses dalam
*Multi Layer*

**2.5.4. Bias**

bias. Bias dapat dipandang
1005).

**2.5.5. Fungsi Aktivasi**

lapisan pada jaringan syaraf tiruan terdapat fungsi
ungsi lain
15
2.3.
*layer*, tentu
ias. Fungsi ini

**2. *Tansig***

Tansig adalah fungsi
sebagai fungsi aktivasi
Gambar 2.5 Fungsi Aktivasi Sigmoid bipolar
Fungsi ini akan membawa nilai input pada output dengan menggunakan
rumus *hyperbolic*
adalah 1 dan minimal
Algoritma dari fungsi ini adalah

**3. *Logsig***

*Logsig* atau *log - s*
output dengan penghitungan
Gambar 2.4 Fungsi Aktivasi Linear
*sigmoid tangen* (*sigmoid bipolar*) yang
aktivasi.
*tangen sigmoid*. Nilai maksimal output dari fungsi ini
-1.
lgoritma *sigmoid* adalah fungsi aktivasi yang membawa input ke
*log-sigmoid*. Nilai outputnya antara 0 hingga 1.
Gambar 2.6 Fungsi Aktivasi *Sigmoid Biner*
16
digunakan
.

17
Algoritma dari fungsi ini adalah
, _ €_ ¡7 _t_ __

_ __e_ j_
_
Masih ada banyak fungsi transfer yang biasa digunakan di jaringan syaraf tiruan antara lain *hardlim, hardlims, comset, netinv, poslin, radbas, satlin, satlins, softmax, tribas,* dan lain-lain.

**2.7. Metode *Backpropagation***

Secara garis besar, mengapa algoritma ini disebut sebagai *backpropagation*, adalah karena: ketika JST diberikan pola masukan sebagai pola pelatihan maka pola tersebut diteruskan ke unit-unit pada lapisan tersembunyi kemudian diteruskan ke unit-unit pada lapisan keluaran. Unit-unit pada lapisan keluaran memberikan tanggapan yang disebut keluaran JST. Saat keluaran JST tidak sama dengan target yang diinginkan maka keluaran akan disebarkan mundur (backward) ke unit – unit pada lapisan tersembunyi dan diteruskan ke unit – unit pada lapisan masukan (Martiana, 2007).

**2.6.1. Metode Pembelajaran *Backpropagation***

Pembelajaran *backpropagation* menggunakan metode pencarian titik minimum untuk mencari bobot dengan eror minimum. Dalam proses pencarian ini terdapat dua macam mode yaitu mode *incremental* dan mode kelompok. Pada mode *incremental*, bobot diubah setiap kali pola masukan diberikan ke jaringan. Sebaliknya, pada mode kelompok, bobot diubah setelah semua pola masukan diberikan ke jaringan. Eror (dan suku perubahan bobot) yang terjadi dalam setiap pola masukan dijumlahkan untuk menghasilkan bobot baru. *Matlab* menggunakan mode pembelajaran kelompok dalam iterasinya. Perubahan bobot dilakukan per*epoch,*
(Haris, 2005).

Algoritma *backpropagation* standar/*gradient descent* (GD) memakai algoritma penurunan gradien standar *(gradient descent algorithm)*. Bobot dan bias diubah pada arah dimana untuk kerja fungsi menurun paling cepat, yaitu dalam arah negatif gradiennya. Dalam *Matlab*, algoritma *backpropagation* standar
18
dilakukan dengan fungsi *traingd*. Parameter metode ini yang ada dalam *Matlab*, dan nilai defaultnya adalah

1. net.trainParam.epochs.100; jumlah *epoch* maksimum
2. net.trainParam.goal 0; *goal* yang diharapkan
3. net.trainParam.Ir 0.01; *learning rate*
4. net.trainParam.max_fail 5; jumlah maksimum kegagalan validasi
5. net.trainParam.mc 0.9; konstanta momentum
6. net.trainParam.min_grad 1e-10; *gradient* kinerja minimum
7. net.trainParam.show 25; jumlah *epoch* yang akan ditunjukkan kemajuannya
8. net.trainParam.time inf; jumlah waktu pembelajaran maksimum (dalam detik).

Metode *backpropagation* standar sering kali terlalu lambat. Beberapa modifikasi dilakukan dengan cara mengganti fungsi pembelajarannya. Secara umum, modifikasi dapat dikelompokkan dalam dua kategori. Kategori pertama menggunakan teknik *heuristik*, yang dikembangkan dari metode penurunan

tercepat yang dipakai dalam *backpropagation* standar. Kategori kedua menggunakan metode penurunan tercepat dan juga optimasi numerik, (Haris, 2006).
Metode *backpropagation* memiliki banyak variasi metode pembelajaran. Adapun metode yang dipakai dalam bahasan ini adalah
1. algoritma *gradient descent* dengan momentum (GDM)
2. algoritma *gradient descent* dengan *Adaptive Learning Rate* dan momentum (GDX)
3. algoritma *Levenberg-Marquadt* (LM).

**2.6.1.1. Algoritma Gradien Descent dengan Momentum (GDM)**

Algoritma ini memakai teknik *heuristik*, yaitu memakai analisis kinerja pada algoritma *steepest* (*gradien*t) *descent standard* dengan menambahkan momentum. Dengan momentum, perubahan bobot tidak hanya didasarkan pada error *epoch* saat itu. Perubahan bobot saat ini dilakukan dengan memperhitungkan juga perubahan bobot pada *epoch* sebelumnya. Dengan demikian kemungkinan

terperangkap ke titik minimum lokal dapat dihindari. Momentum bernilai dari 0 sampai 1. Momentum bernilai 0 berarti perubahan bobot hanya berdasarkan *epoch* saat ini.
Dalam *Matlab*, algoritma GDM dilakukan dengan fungsi *traingdm.*
Parameter khusus metode ini yang ada dalam Matlab, dan nilai *default*nya adalah net.trainParam.mc 0.9 (untuk konstanta momentum). Parameter lainnya seperti fungsi *traingd*.

**2.6.1.2. Algoritma Gradient Descent dengan Adaptive Learning Rate dan Momentum (GDX)**

Algoritma ini memakai teknik *heuristik*, yaitu memakai analisis kinerja pada algoritma *steepest (gradient) descent standard* dengan memakai learning rate dan momentum. Fungsi ini akan memperbaiki bobot-bobot berdasarkan *gradient descent* dan *learning rate* yang bersifat *adaptive* seperti *traingda*, dan juga menggunakan momentum seperti *traingdm*. Learning Rate dapat diubah sesuai eror proses pembelajaran. Jika eror sekarang lebih besar dibandingkan eror sebelumnya, maka *learning rate* diturunkan. Jika sebaliknya, maka *learning rate* dinaikkan.
Penggunaaan *adaptive learning rate* dikombinasikan dengan momentum. Fungsi pelatihan ini memiliki kecepatan pelatihan yang tinggi sehingga dipakai sebagai *default* dalam pembelajaran *backpropagation* dalam Matlab. Algoritma ini juga disebut *SuperSAB* (*Super Self-Adapting Backpropagation*), (Hristev, 1998).
Dalam Matlab, algoritma GDX dilakukan dengan fungsi *traingdx*.
Parameter khusus metode ini yang ada dalam Matlab, dan nilai *default*nya
1. net.trainParam.lr_inc 1.05; rasio untuk menaikkan *learning rate*
2. net.trainParam.lr_dec 0.7; rasio untuk menurunkan *learning rate*
3. net.trainParam.max_perf_inc 1.04; kinerja kenaikkan maksimum
4. net.trainParam.mc 0.9; konstanta momentum.

Parameter lainnya seperti fungsi *traingd.*


**2.6.1.3. Algoritma *Levenberg-Marquadt* (LM)**

Seperti metode quasi-Newton, algoritma *levenberg marquadt* didesain dengan menggunakan pendekatan turunan kedua tanpa harus menghitung matriks

*Hessian*. Apabila jaringan syaraf tiruan *feedforward* menggunakan fungsi kinerja *sum of square*, maka matriks *Hessian* H dapat didekati sebagai
K __ ¢/ ` ¢.
Dan gradien dapat dihitung menggunakan persamaan di bawah ini.
¢£ __ ¢/ ` e
dengan J adalah matriks *Jacobian* yang berisi turunan pertama dari eror jaringan terhadap bobot, dan *e* adalah suatu vektor yang berisi eror jaringan. Matriks *Jacobian* dapat dihitung dengan teknik *backpropagation* standar yang lebih sederhana dibanding menghitung matriks *Hessian*.
Algoritma *levenberg marquadt* menggunakan pendekatan matriks *Hessian* berikut ini,
d)* __ d) __ _¢/_¢ _ __¤_ _¢/e_
Ketika konstanta _ adalah 0 (nol), algoritma ini akan seperti metode Newton, menggunakan pendekatan matriks *Hessian*. Untuk _ yang besar, algoritma ini menjadi *gradient descent* dengan jumlah iterasi yang lebih kecil. Metode Newton lebih cepat dan lebih akurat dalam mendekati nilai eror minimum, jadi tujuan algoritma ini adalah merubah metode Newton menjadi secepat mungkin. Oleh karena itu, _ berkurang setelah setiap *step* berhasil (mengurang kinerja fungsi) dan akan bertambah hanya ketika untuk sesaat iterasinya meningkatkan kinerja fungsi. Di sini, kinerja fungsi akan selalu dikurangi pada setiap iterasi dari algoritma, (*Matlab Help*).

### 2.6.2. Notasi dalam *Backpropagation*

Dimisalkan K _ 3_dv
k_ _v
k__&_ _dv
;_ _v
;_ adalah himpunan pasangan pola
masukan dan target dengan banyak pola = q. Menurut Kusumadewi (2002), untuk satu pasangan pola masukan dan target didefinisikan
z¥ ¦ §¨©ª«¨¬_«y _ -4_____z¥_ _z_ z_&_ _ _ z®_
¯° ¦ ¯¨±_y ¯_«y _ «4_____ ° _ _ _ _&_ _ ¯Y_
21
²¥ ¦ ª¬-¯ _§¨©ª«¨¬_«y _ -_{¨³¨ __¨{-©¨¬_-¬{ª¯
´µ ¦ ª¬-¯ _¯ y±©y§¶ª¬·-_«y _ ,_{¨³¨ __¨{-©¨¬_¯ y±©y§¶ª¬·-
¹º ¦ ª¬-¯ _«y_ª¨±¨¬_«y _ «_{¨³¨ __¨{-©¨¬_ª¯{ª¯
º»- ¦ ¶¶¯ _³¨±-_¶-¨©_³-__¨{-©¨¬_¯ y±©y§¶ª¬·-_«y__¨{-©¨¬_«y_ª¨±¨¬
»µ- ¦ ¶¶¯ _¶-¨©_³-__¨{-©¨¬_§¨©ª«¨¬_«y__¨{-©¨¬_«y__¨{-©¨¬_¯ y±©y§¶ª¬·-
¼º ¦ ½¨«¯ ±_{y±ª¶¨¾¨¬_ª¬¯ª«_§y±ª¶¨³¼_¶¶¯ _º»µ
¼µ ¦ ½¨«¯ ±_{y±ª¶¨¾¨¬_ª¬¯ª«_§y±ª¶¨³¼_¶¶¯ _»µ¥
_»µ¥ ¦ {y±ª¶¨³¼¨¬_¶¶¯ _»µ¥
_º»µ ¦ {y±ª¶¨³¼¨¬_¶¶¯ _º»µ
__º»- ¦ {y±ª¶¨³¼¨¬_¶-¨©_º»-
_»µ- ¦ {y±ª¶¨³¼¨¬_¶-¨©_»µ-
¿ ¦ _¨,ª_{y§¶y_¨,¨±¨¬.

### 2.6.3. Fase Pelatihan Backpropagation

Fase pelatihan *backpropagation* dengan *n* unit masukan (ditambah sebuah bias), sebuah lapisan tersembunyi yang terdiri dari *p* unit tersembunyi (ditambah sebuah bias), serta *m* unit keluaran (Fausset, L. : 1994).
*a. Fase I : Propagasi maju dari pola masukan*

Selama propagasi maju, setiap unit masukan *Xi* dengan *i* = 1, . . ., n menerima masukan *xi* dan meneruskan ke setiap unit tersembunyi *Zj* dengan *j =1,…., p*. Setiap unit tersembunyi menghitung nilai aktivasi $z_j$ dengan fungsi aktivasi yang ditentukan dan meneruskan $z_j$ ke setiap unit keluaran $Y_k$ dengan *k = 1,…, m.* setiap unit keluaran menghitung aktivasinya dengan fungsi aktivasi yang ditentukan untuk menghasilkan keluaran jaringan $y_k$. Selanjutnya setiap unit keluaran membandingkan $y_k$ dengan nilai target yang harus dicapai $t_k$. Selisih $t_k – y_k$ adalah kesalahan yang terjadi. Jika kesalahan ini lebih kecil dari batas toleransi yang ditentukan, maka epoch dihentikan. Sebaliknya ketika kesalahan masih lebih besar dari batas toleransinya, maka bobot dalam jaringan akan diperbaiki untuk mengurangi kesalahan yang terjadi.

22

*b. Fase II : Propagasi mundur error*

Berdasarkan kesalahan lalu _◦ dihitung. _◦ digunakan untuk mendistribusikan kesalahan (di unit keluaran À)) kembali ke semua unit tersembunyi yang terhubung langsung dengan c). _◦ juga digunakan untuk memperbaiki bobot-bobot diantara lapisan keluaran dan lapisan tersembunyi. Dengan cara yang sama, _µ dihitung untuk setiap unit tersembunyi *Zj*. _µ digunakan untuk memperbaiki bobot-bobot diantara lapisan tersembunyi dan lapisan masukan.

*c. Fase III perubahan bobot*

Setelah semua faktor m dihitung, bobot semua lapisan diperbaiki bersamaan. Perubahan bobot $w_{kj}$ (dari unit tersembunyi $Z_j$ ke unit keluaran $Y_k$) didasarkan atas m). Perubahan bobot $v_{ij}$ (dari unit masukan $X_i$ ke unit tersembunyi $Z_j$) didasarkan atas mw.

Pada fase pengujian hanya digunakan salah satu fase dari pelatihan yaitu alur maju (propagasi maju). Berikut contoh langkah-langkah dalam pelatihan dalam *backpropagation* dengan satu lapisan tersembunyi dengan fungsi aktivasi *sigmoid biner* dan *pureline*, (Kusumadewi, 2002).

Langkah 1 : Inisialisasi bobot (dipilih nilai yang terkecil).

Langkah 2 : Jika kondisi tidak tercapai lakukan langkah 3-10.

Langkah 3 : Untuk setiap pasangan pelatihan lakukan langkah 4-9.

**Propagasi maju :**

Langkah 4 : Setiap unit masukan ($X_i$, *i=1, …, n*) menerima sinyal $x_i$ dan menghantarkan sinyal ini ke semua unit di lapisan di atasnya (lapisan tersembunyi).

Langkah 5 : Setiap unit tersembunyi ($Z_j$, *j=1, …., p*) jumlahkan bobot masukannya, ÁÂte_w __ _w- __# dv_wv

jV

0- .

Menghitung $z_j$ dengan fungsi aktivasi yang sudah ditentukan.

a. Dengan fungsi aktivasi pureline

Áw _ wlÁÂte_wp __ _w- __# dv_wv

jV

0 .

b. Dengan fungsi aktivasi sigmoid biner

23

Áw _ wlÁÂte_wp _

*ÃÄÅÂÆÇ.È.
Untuk setiap $z_j$, selanjutnya diteruskan ke semua unit keluaran.
Langkah 6 : Setiap unit keluaran ($Y_k$, $k = 1, 2, \ldots, m$) jumlahkan bobot sinyal masukannya
cÂte_) __É)- __:Áw_É)w_
U
W0
Menghitung $Y_k$ dengan fungsi aktivasi.
a. Fungsi aktivasi pureline
c) _ w_cÂte_)_ _ É)- __:Áw_É)w
U
W0

__
b. Fungsi aktivasi sigmoid biner
c) _ w_cÂte_)_ __

_ e ÊÂjÃ_Ë

_
**Propagasi mundur :**
Langkah 7 : Setiap unit keluaran ($Y_k$, $k = 1, 2, \ldots, m$) menerima pola target yang saling berhubungan pada masukan pola pelatihan, hitung kesalahan (eror).
_ —

%
_:__) __c)_
i
)0
Menghitung gradient (turunan pertama) negative $E$

É)w
—

É)w
q

%
:__) __c)_
i
)0
r
__ ì
ÎÍÉÈ
_u
_
# l_) _ _w_cÂte_)_p_ i
)0 v
____:__) _ _w_cÂte_)_
i
)0
__wk_cÂte_)__

É)w
__cÂte_)__
a. Dengan fungsi aktivasi pureline diperoleh

É)w
___:__) __c)_Áw
i
)0
atau dapat ditulis menjadi
24
m) __:__) __c)_
i
)0

_
Hitung _É)w dengan laju _ (untuk memperbaiki bobot nantinya)
_É)w ___Q_

É)w
____Q q_:__) __c)_Áw
i
)0
r _ _Q_m)_Áw
set ,t__ _ _&_Î_s,t_H _ _&_ X_
b. Dengan fungsi aktivasi sigmoid biner

É)w
___:__) __c)__lc)_ _ c)_pÁw
i
)0
atau dapat ditulis menjadi
m) __:__) __c)__lc)_ _ c)_p_
i
)0
Hitung _É)w dengan laju _
_É)w ___Q_

É)w
____Q q_:__) __c)__lc)_ _ c)_pÁw
i
)0
r
_ _Q_m)_Áw
set ,t__ _ _&_Î_s,t_H _ _&_ X_
Menghitung _É)- yang nanti akan digunakan untuk merubah É)-
_É)- _ _Q_m)_
Langkah 8 : Setiap unit lapisan tersembunyi *($Z_j$, j=1, …, p)* jumlahkan hasil perubahan masukan dari unit-unit lapisan di atasnya.
mÂte_w _:m)_É)w
i
)0
Menghitung gradien (turunan pertama) negative *E*

_
_wv

__

_wv
q

%
:__) __c)_
i
)0
r
___ ì
ÌÏËÐ
_u
_
# l_) _ _w_cÂte_)_p_ i
) 0 v
25

_
_wv
____:__) _ _w_cÂte_)_
i
)0
__wk_cÂte_)__

_wv
__cÂte_)__
a. Digunakan fungsi aktivasi *pureline*

_
_wv
____:__) _ c)_

_wv
i
)0
__cÂte_)_
____:m)

_wv
i
)0
__cÂte_)_
____:m)É)w

_wv
i
)0
_Áw_
____:m)É)wwk_ÁÂte_)_

_wv
i
)0
_ÁÂte_w_

_ _q:m)É)w
i
)0
r dv_
atau dapat ditulis menjadi
mw __ q:m)_É)w
i
)0
r_
Menghitung __wv dengan laju _
__wv ___Q_

_
_wv

_
____Q_ qÑ Ò:m)_É)w
i
)0
Ó dvr
_ Q_mw_dv
set ,t_H _ _&_ X_s,t_7 _ _&_ t_
b. Dengan fungsi aktivasi sigmoid biner

_
_wv
____:__) _ c)__c)_ _ c)__
i
)0
_

_wv
__cÂte_)_
___# m) _ ì
ÏÏÈÐ
_cÂte_)_ ¡
) 0
26

_
_wv
____:m)_É)w _

_wv
_Áw
i
)0
___:m)É)wwk_ÁÂte_)_

_wv
i
)0
_ÁÂte_w
____:m)_É)w__c)_ _ c)___dv_
i
)0
____q:Ôm)_É)w_c)__ _ c)_Õ
i

)0
r_dv
atau dapat ditulis menjadi
mw __ q:m)_É)w_c)__ _ c)
i
)0
_r_
Menghitung __wv dengan laju _
__wv ___Q_
_
_wv
____Q_ ÖÑ_.:m)_É)w_c)
i
)0
__ _ c)_._dv×
_ _Q_mw_dv_
Terakhir, menghitung __w- yang digunakan untuk merubah _w- :
__wv _ _Q_mw_

**Perubahan bobot:**

Langkah 9 : Setelah semua target dan input selesai dilatih dan dihitung masingmasing
besar perubahan bobotnya pada setiap unit. Langkah selanjutnya adalah merubah nilai bobot.

Langkah 10 : Proses terhenti jika maksimum *epoch* tercapai.

**2.6.4. Fungsi Aktivasi dalam Backpropagation**

Dalam *backpropagation*, fungsi aktivasi yang dipakai harus memenuhi beberapa syarat sebagai berikut

1. kontinu



2. memiliki turunan pertama
3. merupakan fungsi yang tidak turun.

Salah satu fungsi yang memenuhi ketiga syarat tersebut sehingga sering dipakai adalah fungsi *sigmoid biner* yang memiliki *range* (0,1). Fungsi lain yang sering dipakai adalah fungsi *sigmoid bipolar* dengan *range* (-1,1). Fungsi sigmoid memiliki nilai maksimum 1. Untuk pola yang targetnya lebih dari 1, pola masukan dan keluaran harus terlebih dahulu ditransformasi sehingga semua polanya memiliki *range* yang sama seperti fungsi *sigmoid* yang dipakai, (Anugerah, 2007).

Sesuai dengan batasan masalah, pada jaringan syaraf tiruan *backpropagation* akan digunakan fungsi aktivasi *sigmoid biner* dan *pureline*. Dimana data harus ditransformasi terlebih dahulu dalam range [0.1:0.9] untuk mencegah nilai berada di asimtot *biner*.

**2.6.5. *Backpropagation* dalam Peramalan**

Ada beberapa langkah yang harus ditempuh dalam meramalkan indeks keuangan dengan metode JST *backpropagation*.

**1. Transformasi data**

Sebelum melakukan pelatihan pada jaringan yang akan digunakan untuk peramalan terlebih dahulu dilakukan transformasi data. Transformasi data diperlukan agar kestabilan taburan data dapat dicapai sekaligus untuk menyesuaikan nilai data dengan *range* fungsi aktivasi yang digunakan dalam

jaringan (Siang, 2005:121).
Pada skripsi ini digunakan fungsi aktivasi *sigmoid biner* yang memiliki range (0:1) sehingga data harus ditransformasikan terlebih dahulu ke dalam range ini. Untuk mencegah nilai berada di asimtot *biner* maka data ditransformasi ke dalam range [0.1:0.9]. Melihat pada Gambar 2.6 untuk fungsi aktivasi *sigmoid biner (log-sigmoid)*, nilai tidak akan pernah menyentuh 0 dan 1.
Berikut adalah rumus transformasi datanya
28
Ø_7_ _ Ù
dv _ ~
Ú _ ~
_Û _ Q_Ü _ Q
__ Ù
dv _ ~
Ú _ ~
_$_Ý _ $_ _Ü _ $_
__ Ù
dv _ ~
Ú _ ~
_$_Þ_Ü _ $_
dengan l : nilai minimum dari seluruh data
h : nilai maksimum dari seluruh data
ω : nilai tertinggi interval
_ : nilai terendah interval.

**2. Pembagian data.**

Aspek pembagian data harus ditekankan agar jaringan mendapat data pelatihan yang secukupnya dan data pengujian dapat menguji prestasi pelatihan yang dilakukan. Jumlah data yang kurang untuk proses pelatihan akan menyebabkan jaringan tidak dapat mempelajari taburan data dengan baik. Sebaliknya, data yang terlalu banyak untuk proses pelatihan akan melambatkan poses pemusatan (konvergensi). Masalah *overtraining* (data pelatihan yang berlebihan) akan menyebabkan jaringan cenderung untuk menghafal data yang dimasukan daripada mengeneralisasi, (Anugerah, 2007).
Seluruh data yang ada merupakan data pemodelan yang nantinya akan dipakai untuk meramalkan. Data pemodelan ini akan dibagi menjadi dua yaitu data pelatihan dan data pengujian. Komposisi data pelatihan dan pengujian yang akan digunakan adalah : 60% untuk data pelatihan dan 40% untuk data pengujian. Dengan jumlah keseluruhan data IHSG yang digunakan adalah 1208. Hal ini berarti diperoleh sebanyak 725 data pelatihan dan 483 data pengujian.

**3. Perancangan jaringan yang optimum.**

Langkah selanjutnya setelah pembagian data adalah penentuan jumlah unit masukan, jumlah lapisan tersembunyi, jumlah unit lapisan tersembunyi dan jumlah unit lapisan keluaran yang akan digunakan dalam jaringan. Belum ada aturan yang pasti untuk menentukan jumlah lapisan tersembunyi dan jumlah unit dalam setiap lapisan. Yang biasa dilakukan
29
dalam beberapa penelitian, untuk menentukan jumlah unit tersebut digunakan

langkah *trial* (eksperimen) dan *error.*
Penggunaan jaringan dengan dua atau lebih lapisan tersembunyi dalam masalah peramalan kebanyakan tidak memberikan pengaruh yang besar terhadap prestasi jaringan untuk melakukan peramalan. Selain dapat melambatkan proses pelatihan yang disebabkan bertambahnya unit, terlalu banyaknya lapisan tersembunyi akan mengakibatkan masalah *overtrained* yaitu jaringan tidak dapat men-*generalize* dengan baik. Sebuah lapisan tersembunyi sudah cukup bagi *backpropagation* untuk mengenali pola masukan dan target dengan tingkat ketelitian yang ditentukan. (Fausset, L:1994).
Penentuan bilangan unit tersembunyi yang optimum diperoleh secara pengembangan melalui *trial and error* dari t
ß% unit sampai n (n
adalah jumlah unit masukan). Dimana untuk variasi unit input yang akan digunakan pada penelitian ini adalah 1, 2, 3, 4, 5, 6, dan 7 unit input. Untuk menghindari kurangnya pembelajaran pola data, khusus unit input yang kurang dari sama dengan 3 _t S C_ digunakan aturan unit *hidden* n hingga 2n. Sehingga variasi unit lapisan *hidden* yang digunakan adalah 3, 4, 5, 6, dan 7.

**4. Memilih dan Menggunakan Struktur Jaringan yang Optimum**

Jaringan yang dibangun akan dinilai keakuratan ramalannya.
Aturan penilaian yang digunakan adalah berdasar nilai *Percentage Error* (PE), *Mean Absolute Percentage Error* (*MAPE)* dan *Mean Square Error* (MSE). Pendekatan MSE digunakan untuk menilai prestasi jaringan yang dilatih, semakin kecil nilai MSE yang dihasilkan semakin baik prestasi jaringan dalam mempelajari pola data. Ketepatan model diukur secara relatif menggunakan MAPE, PE dan dilihat juga dari nilai MSE-nya. Nilai PE, MAPE dan MSE diperoleh dari persamaan berikut

__ _
_c_ _ cl_
c_
_d_ $$_à
30
áâ__ _:
N__N
t
j

_
áa_ __

t
:ev
_
j
V0

—

t
_:_c_ _ cl_

j
V0
dengan c_ __nilai aktual pada waktu t
cI_ __nilai ramalan pada waktu t
t __ jumlah data.
Berdasarkan nilai MSE terendah dari proses pelatihan diperoleh jaringan yang optimum. Keakuratan ramalan jaringan dilihat dari nilai PE, MAPE, dan MSE dari proses pengujian.

**5. Pemilihan Jaringan yang Optimum dan Penggunaannya dalam Peramalan**

Langkah-langkah pemilihan jaringan optimum sebagai berikut
a. proses pelatihan dilakukan dengan struktur jaringan yang memiliki variasi bilangan lapisan tersembunyi dan unit dalam setiap lapisan. Kemudian berdasarkan kaedah nilai MSE diperoleh jaringan yang optimum
b. selanjutnya proses pengujian. Di sini keakuratan ramalan jaringan akan ditentukan berdasarkan nilai PE, MAPE, dan MSE
c. peramalan menggunakan struktur jaringan optimum yang telah dipilih.



## BAB III
## METODE PENELITIAN

Penelitian ini dilaksanakan dengan metode studi kasus yaitu dengan menganalisa data yang ada dan memodelkannya dengan menggunakan dua buah metode yaitu metode GARCH dan jaringan syaraf tiruan *backpropagation*. Hasil peramalan dari kedua metode dibandingkan berdasarkan nilai MAPE dan MSE. Data yang digunakan adalah data sekunder yang diambil dari data IHSG harian *(*secara *trading days)* di BEI.

### A. Analisis Data

**a) Analisis model GARCH**

1. Memplot data untuk melihat pola data dan stasioneritasnya.
2. Data yang belum stasioner diubah ke dalam bentuk log *return* untuk menstasionerkan data terhadap rata-rata.
3. Dengan bantuan *software Matlab* dibuat plot ACF untuk mengidentifikasi korelasi pada log *return* (*return series*).
4. Dilakukan uji *Ljung-Box-Pierce* untuk membuktikan adanya autokorelasi. Jika terdapat autokorelasi maka pemodelan runtun waktu dapat dilakukan.
5. Identifikasi efek heteroskedastisitas dari residu model rata-rata bersyarat dengan bantuan *software Matlab 7.0.1.* Pada penelitian sebelumnya yang dilakukan oleh Widyanti, untuk membuktikan asumsi heteroskedastisitas perlu dibuat terlebih dahulu model mean bersyarat dengan bantuan *software Eviews 4.1*. Dalam hal ini *software Matlab 7.0.1* telah menyediakan *sintax* untuk uji asumsi tersebut, yaitu menggunakan fungsi *archtest* (berdasarkan uji *Lagrange Multiplier*).
6. *Overfitting* (pemilihan) model, orde *p* dan *q* yang digunakan dalam model GARCH bervariasi antara 1 sampai dengan 3, dan diperoleh model GARCH terbaik dengan orde tertentu. Tahap pemilihan model dilakukan menurut kaedah nilai AIC.
7. Analisis model.

Dengan bantuan *software Matlab* dalam analisis model GARCH dilakukan dalam dua tahap, yaitu

a. tahap Estimasi Parameter (menggunakan *maximum likelihood estimation* dan direpresentasikan dengan fungsi *garchfit* dan *garchdisp* yang ada di *Matlab*) dari model yang terpilih melalui *overfitting*

b. tahap *postestimation* (estimasi akhir). Di sini terdapat 2 buah uji, yaitu

- uji autokorelasi pada *residual distandardisasi* untuk pemeriksaaan diagnostik model. Uji yang digunakan adalah *Ljung-Box-Pierce Q-Test*
- uji efek ARCH pada *residual distandardisasi* untuk mengetahui apakah model GARCH yang digunakan sudah cukup baik.

8. Menghitung hasil ramalan dari pendekatan *return* yang diperoleh.

**b) Simulasi dan rancangan struktur jaringan syaraf tiruan *backpropagation*.**

1. Transformasi data

Berdasarkan fungsi aktivasi yang digunakan yaitu *sigmoid biner.* Berikut adalah rumus transformasi datanya

[illegible]

dengan l : nilai minimum dari seluruh data

h : nilai maksimum dari seluruh data

ω : nilai tertinggi interval

_ : nilai terendah interval pembagian data.

33

2. Pembagian data

Seluruh data yang ada merupakan data pemodelan yang nantinya akan dipakai untuk meramalkan. Data pemodelan ini akan dibagi menjadi dua yaitu data pelatihan dan data pengujian. Komposisi data pelatihan dan pengujian yang akan digunakan adalah : 60% untuk data pelatihan dan 40% untuk data pengujian. Dari data IHSG periode 2 Januari 2004 sampai 31 Desember 2008, diperoleh jumlah keseluruhan data adalah 1208. Hal ini berarti terdapat sejumlah 725 data pelatihan dan 483 data pengujian.

3. Perancangan jaringan yang optimum

Langkah selanjutnya setelah pembagian data adalah penentuan jumlah unit masukan, jumlah lapisan tersembunyi, jumlah unit lapisan tersembunyi dan jumlah unit lapisan keluaran yang akan digunakan dalam jaringan.

4. Memilih dan Menggunakan Struktur Jaringan yang Optimum

Jaringan yang dibangun akan dinilai keakuratan ramalannya. Aturan penilaian yang digunakan adalah berdasar nilai PE, MAPE dan MSE. Pendekatan MSE digunakan untuk menilai prestasi jaringan yang

dilatih semakin kecil nilai MSE yang dihasilkan semakin baik prestasi jaringan dalam mempelajari pola data. Ketepatan model diukur secara relatif menggunakan PE dan dilihat juga dari nilai MSE-nya.

5. Pemilihan Jaringan yang Optimum dan Penggunaannya dalam Peramalan

Langkah-langkah pemilihan jaringan optimum sebagai berikut

a. proses pelatihan dilakukan dengan struktur jaringan yang memiliki variasi bilangan lapisan tersembunyi dan unit dalam setiap lapisan. Berdasarkan kaedah nilai MSE diperoleh jaringan yang optimum

b. selanjutnya dilakukan proses pengujian. Di sini keakuratan ramalan jaringan akan ditentukan berdasarkan nilai PE, MAPE, dan MSE

c. peramalan menggunakan struktur jaringan optimum yang telah dipilih

34

**B. Penarikan Simpulan**

Setelah hasil peramalan dengan masing-masing metode diperoleh, ditentukan nilai MAPE dan MSE-nya. Nilai MAPE dan MSE yang lebih kecil mengidentifikasikan keakuratan hasil ramalan data IHSG yang lebih baik.

35

**BAB IV**

**HASIL DAN PEMBAHASAN**

**4.1. Deskripsi Data**

Data di sini adalah data yang diambil dari data IHSG saat *trading days* (hari Senin – Jum'at) di BEJ sejak periode 2 Januari 2004 sampai dengan akhir Desember 2008. Jumlah keseluruhan data adalah 1208 dengan tabel data terlampir. Sifat data dan stasioneritasnya dapat dilihat melalui plot data *financial series* (menggunakan perangkat lunak *Matlab 7.0.1*). Hasil plot data *financial time series* menggunakan *Matlab* dapat dilihat pada Gambar 4.1.

Gambar 4.1 Grafik Data IHSG Periode 2 Januari 2004 Sampai 31 Desember 2008

Dari grafik di atas terlihat bahwa IHSG mengalami fluktuasi dari waktu ke waktu, dan dapat dilihat juga bahwa grafik tersebut menunjukkan adanya *trend* naik dan *trend* turun. Hal ini berarti data IHSG belum stasioner.

**4.2. Hasil**

Pada sub-bab ini dijelaskan hasil analisis data, baik dengan menggunakan model GARCH maupun jaringan syaraf tiruan *backpropagation.*

Jan 2004 Jan 2005 Jan 2006 Jan2007 Jan 2008
500
1000
1500
2000
2500
3000
Exchange Rate
Indeks Harga Saham Gabungan

36

**4.2.1. Model GARCH**

Di dalam *Matlab* terdapat tiga tahapan untuk mensimulasikan model GARCH, yaitu *Pre estimation,* Estimasi parameter dan *Postestimation.* Selanjutnya akan dibahas masing-masing tahapannya.

**4.2.2.1. *Pre Estimation Analysis***

Data runtun waktu *financial* terlebih dahulu diubah kedalam bentuk *return series* atau lebih dikenal dengan log *return*. Fungsi *return series* ini dirumuskan sebagai berikut :

_– _ ___ _– _ ___ _–
Langkah selanjutnya, dilihat apakah *return series* memiliki korelasi atau tidak melalui plot ACF-nya, kemudian asumsi diperkuat dengan melakukan uji menggunakan *Ljung Box Pierce*, dan terakhir dilakukan uji untuk menentukan ada tidaknya efek ARCH pada *return series*.

#### 4.2.2.1.1. *Return Series*

Berdasarkan plot data pada gambar 4.1 terlihat bahwa grafik datanya mengalami trend naik dan trend turun, sehingga dapat disimpulkan bahwa data belum stasioner. Agar data menjadi stasioner, dalam analisis *financial time series* kita bisa menggunakan *log-return* untuk mempermudah analisis data sekaligus menjadikan data stasioner terhadap rata-rata.

Alasan kenapa log *return* menjadi pilihan utama dalam analisis ini adalah:

1. *Log-return* bersifat bebas skala sehingga lebih obyektif sebagai bahan perbandingan.
2. *Log-return* bersifat stasioner.

Perintah yang digunakan dalam *Matlab* untuk mengubah data kedalam *log-terurn* (*return series*) adalah:

```
>> ihsg = price2ret(ihsg);
```

Untuk memperjelas sifat stasioner dari *return series*, dapat dilihat plot *return series*-nya. Dengan menggunakan *software Matlab* diperoleh *output* seperti terlihat pada Gambar 4.2. Dari garfik pada Gambar 4.2 terlihat bahwa plot *return*



*series* menunjukkan rata-rata *log- return* yang konsisten (stasioner) dengan nilai yang mendekati nol.

Gambar 4.2 Grafik *Return Series* IHSG Periode 2 Januari 2004 Sampai 31 Desember 2008

#### 4.2.2.1.2. Plot ACF *Return Series*

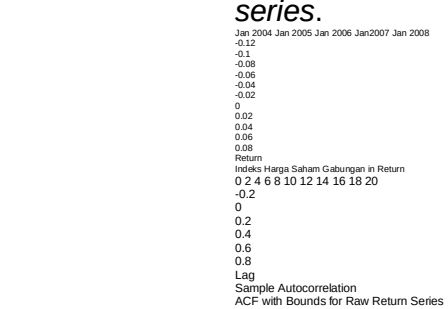
Di *Matlab*, perintah untuk membuat plot ACF adalah sebagai berikut

```
>> figure,autocorr(ihsg)
title('ACF with Bounds for Raw Return Series')
```

hasil yang diperoleh dapat dilihat pada Gambar 4.3 di bawah.

Gambar 4.3 Plot ACF *Return Series*

Dapat dilihat pada Gambar 4.3 di atas, *lag* pertama pada plot ACF *return series* signifikan tidak sama dengan 0, hal ini berarti autokorelasi pada return series dipenuhi. Untuk membuktikannya dilakukan uji *Ljung Box Pierce* pada *return series*.






Digunakan _ _ ___ dan

$H_0$ = tidak terdapat korelasi pada *return series*

$H_1$ = terdapat autokorelasi pada *return series.*

Dengan fungsi *lbqtest* di *Matlab* didapatkan hasil seperti terlihat pada Tabel 4.1.

Tabel 4.1 Hasil Uji Ljung Box Pierce

| ***Lag*** | **pValue** | **Stat** | **Critic Value** |
|---|---|---|---|
| 6 | 0.0001 | 27.4935 | 12.5916 |
| 12 | 0.0001 | 38.3106 | 21.0261 |
| 18 | 0.0003 | 45.6239 | 28.8693 |
| 24 | 0.0005 | 53.6259 | 36.4150 |

Dari output diperoleh nilai *pValue* yaitu 0.0001, 0.0003, dan 0.0005 dimana nilai ini lebih kecil dari _ _ ___, maka dalam uji ini Ho ditolak atau dengan kata lain terdapat autokorelasi pada *return series.* Berarti model runtun waktu dapat dibuat.

**4.2.2.1.3. Efek Heteroskedastisitas *Return Series***

Uji efek ARCH dilakukan dengan menggunakan fungsi *archtest* (dimana fungsi ini didasarkan pada *Engle's ARCH test* 1982), digunakan _ _ ___ dan

Ho = tidak terdapat efek ARCH pada data return

H1 = terdapat efek ARCH pada data *return*.

Output uji efek heteroskedastik (*archtest*) ini dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil *archtest* Pada *Return Series*

| ***Lag*** | **pValue** | **Stat** | **Critic Value** |
|---|---|---|---|
| 6 | 0 | 215.3317 | 12.5916 |
| 12 | 0 | 238.7144 | 21.0261 |
| 18 | 0 | 246.7501 | 28.8693 |
| 24 | 0 | 252.9406 | 36.4150 |

Diperoleh nilai *pValue* adalah 0 dan jauh lebih kecil dari _ _ ___ sehingga Ho ditolak atau dengan kata lain efek ARCH pad *return series* dipenuhi. Hal ini berarti model runtun waktu heteroskedastisitas dapat dibuat.



**4.2.2.2. *Overfitting Model***

Sebelum melakukan estimasi parameter dan *postestimation* dilakukan *overfitting* model GARCH dengan variasi orde p dan q untuk mendapat model GARCH terbaik dengan orde tertentu. Variasi model yang dicobakan adalah: GARCH(1,2), GARCH(2,1), GARCH(2,2), GARCH(3,1), GARCH(3,2), GARCH(1,3), GARCH(2,3), dan GARCH(3,3). Pemilihan model dilakukan memalui perhitungan nilai AIC. Hasil output analisis untuk *overfitting* dapat dilihat di Lampiran 5. Kesimpulan yang dapat diambil dari *overfitting* model GARCH dengan parameter paling berpengaruh adalah GARCH(1,1).

**4.2.2.3. Estimasi Parameter**

Model GARCH yang akan diestimasi di sini adalah model terbaik yang diambil dari hasil *overfitting* menggunakan bantuan *software Matlab* yaitu GARCH(1,1). Dalam mengestimasi parameter model GARCH digunakan fungsi *garchfit* (kaedah yang digunakan dalam fungsi ini adalah *Maximum Likelihood Estimation*). Untuk melihat hasil estimasi digunakan fungsi garchdisp, dengan output diperlihatkan pada Tabel 4.3.

Tabel 4.3 Hasil Estimasi Parameter GARCH(1,1)

| **Parameter** | ***Value*** | ***Standard Error*** | **Statistik T** |
|---|---|---|---|
| K | 1.1267e-005 | 1.793e-006 | 6.2840 |
| GARCH(1) | 0.79419 | 0.015441 | 51.4353 |
| ARCH(1) | 0.16974 | 0.018529 | 9.1611 |

Dalam *Matlab output* statistik-*T* digunakan sebagai ukuran nilai

standar deviasi estimasi parameter tidak sama dengan nol. Menggunakan aturan distribusi normal, dapat dilihat bahwa seluruh parameter memiliki nilai statistik *T* yang lebih besar dari 2 (digunakan Z_ = $Z_{0.05}$ = 1.96 _ 2), berarti seluruh parameter signifikan sehingga diperoleh model GARCH(1,1) sebagai berikut:

_–
– – ——— – ——— – ———————–
– – ———————–
_
dimana _– _ – ——— – ——— – – ———————— – – ———
40

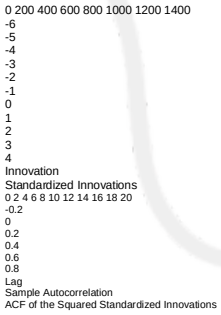
#### 4.2.2.4. *Postestimation Analysis*

Pada *postestimation* dilakukan pengujian terhadap *residual (Innovations)* distandardisasi apakah terdapat efek ARCH ataukah tidak. Sedangkan uji korelasi serial diberlakukan pada kuadrat *residu* distandardisasinya. Hal ini bertujuan untuk pemeriksaan diagnostik model. Uji yang digunakan adalah uji *Ljung Box Pierce* untuk pengujian korelasi serial dan *archtest* untuk uji efek ARCH. Sebeum dilakukan uji dapat dilihat plot dari *residual* distandardisasi dan ACF *residual* kuadratnya (lihat Gambar 4.4 dan 4.5). Gambar 4.4 menunjukkan bahwa grafik *residual* distandardisasinya stasioner terhadap *mean*.

Gambar 4.4 Plot *Standardized Innovations (Residual* Distandardisasi*)*

Pada Gambar 4.5 yang merupakan plot ACF dari kuadrat *residual* distandardisasi, dapat dilihat bahwa nilai ACF dari *lag-lag* yang ditampilkan pada plot ACF tidak ada yang signifikan tidak sama dengan nol (tidak memotong garis) sehingga autokorelasi tidak dipenuhi pada kuadrat *residual* distandardisasi. Agar lebih yakin pada hasil ini, dilakukan uji *Ljung Box Pierce*.

Gambar 4.5 Plot ACF Kuadrat *Residual* Distandardisasi




41

Hasil uji *Ljung Box Pierce* manggunakan *Matlab* untuk pemeriksaan autokorelasi (korelasi serial) pada kuadrat *residual* distandardisasi dapat dilihat pada Tabel 4.4.

Table 4.4 Hasil Uji *Ljung Box Pierce* Pada Kuadrat *Residu* Distandardisasi

| ***Lag*** | **H** | **pValue** | **Stat** | **Critic Value** |
|---|---|---|---|---|
| 6 | 0 | 0.7741 | 3.2711 | 12.5916 |
| 12 | 0 | 0.8365 | 7.3096 | 21.0261 |
| 18 | 0 | 0.9638 | 8.8185 | 28.8693 |
| 24 | 0 | 0.9342 | 14.5133 | 36.4150 |

Ditentukan

Ho = tidak terdapat korelasi serial pada *residual* distandardisasi

$H_1$ = terdapat korelasi serial pada *residual* distandardisasi.

Dapat dilihat dari Tabel 4.4 diperoleh nilai H adalah 0 artinya Ho

diterima atau dengan kata lain tidak terdapat korelasi serial pada *residual* distandardisasi. Hal ini berarti model cukup baik untuk digunakan dalam peramalan nilai IHSG.

Untuk melihat ada tidaknya efek ARCH dalam *residual* distandardisasi, digunakan fungsi *archtest* dan diperoleh output seperti yang ditunjukkan oleh Tabel 4.5.

Tabel 4.5 Hasil Uji Efek ARCH pada *Residual* Distandardisasi

| ***Lag*** | **H** | **pValue** | **Stat** | **Critic Value** |
|---|---|---|---|---|
| 6 | 0 | 0.7705 | 3.2991 | 12.5916 |
| 12 | 0 | 0.8597 | 6.9684 | 21.0261 |
| 18 | 0 | 0.9789 | 7.9854 | 28.8693 |
| 24 | 0 | 0.9622 | 13.2298 | 36.4150 |

Ditentukan

Ho = tidak terdapat efek ARCH pada residual distandardisasi

H1 = terdapat efek ARCH pada *residual* distandardisasi.

Dari output diperoleh nilai H adalah 0 sehingga Ho diterima atau dengan kata lain tidak terdapat efek ARCH pada *residual* distandardisasi. Berarti model sudah cukup baik.

42

**4.2.2.5. Peramalan**

Dalam peramalan digunakan model GARCH(1,1) untuk meramalkan data IHSG 19 langkah ke depan atau satu bulan berikutnya yaitu Januari 2009. Menggunakan *Matlab* diperoleh output pendekatan nilai *return* untuk 19 langkah ke depan sebesar 0. Tabel output dari hasil peramalan *return series* dapat dilihat di Lampiran 1. Hasil pendekatan nilai *return* digunakan untuk menghitung nilai peramalan IHSG. Nilai peramalan tersebut diperoleh melalui rumus sebagai berikut

—– – ——
_
_ !"
–
– — ——
_
_ !"
–
– —
_
_ !"
#_ __#_

Hasil peramalan menggunakan model GARCH (1,1) diperlihatkan oleh Gambar 4.6.

Gambar 4.6 Hasil Ramalan IHSG Menggunakan GARCH(1,1) dan JST*backpropagation*

Tabel hasil perhitungan eror dan PE berdasarkan nilai peramalan yang diperoleh dapat dilihat di Lampiran 2. Berdasarkan nilai eror dan PE diperoleh nilai MSE dan MAPE sebagai berikut

$%& __
'
_(')#_ _ #*_+
,- _ _-———

$/_& __(
0120
'
'
_3____4 _ ____4
1200
1250
1300
1350
1400
1450
1500
1 2 3 4 5 6 7 8 9 10111213141516171819
Hasil Ramalan dg
GARCH(1,1)
Hasil Ramalan
dengan JSTbackpropagation
Data Aktual
43

**4.2.2. Metode Jaringan Syaraf Tiruan *Backpropagation***

Setelah melakukan transformasi data asli pada range [0.1:0.9], arsitektur jaringan syaraf tiruan *backpropagation* dibangun menggunakan *Matlab* dengan fungsi berikut

*BPNN = newff(minmax(Input), [X1 X2 X3 ...], {'TF1' 'TF2' 'TF3' ...}, 'train');*

dimana

BPNN : nama jaringan,

Input : nama file masukan (input),

X : menyatakan banyaknya neuron (unit) tiap lapisan, dan jumlah “X” menyatakan banyaknya lapisan,

TF : menyatakan fungsi aktivasi yang dipakai (di sini kita memakai *logsig* dan *pureline,*

*train* : metode pembelajaran yang dipakai (*traingdm, traingdx, dan trainlm*).

Hasil yang diperoleh melalui beberapa pelatihan dengan penggunaan nilai ambang batas untuk eror minimum sebesar 0.0001 dan variasi unit tiap lapisan serta tiga jenis metode pembelajaran berbeda, didapatkan 5 besar rancangan jaringan optimum. Rancangan terbaik diantara kelima-nya adalah jaringan dengan unit input sejumlah 5, dan jumlah unit tersembunyi adalah 3 unit. Hal ini dikarenakan nilai MSE pada saat pelatihan cukup kecil yaitu 0.000182859, meski bukan yang terkecil akan tetapi nilai MSE pada saat pengujian untuk struktur jaringan ini (5 – 3 – 1) adalah yang terkecil yaitu 0.000141621. Dari tiga metode pembelajaran yang diterapkan, berdasarkan konvergensinya diperoleh yang terbaik adalah metode *levenberg marquadt* karena metode ini memiliki kemampuan yang baik untuk meredam eror saat pelatihan. Untuk lebih jelasnya dapat dilihat Tabel 4.6.

44

Tabel 4.6 Perbandingan Nilai MSE Saat Pelatihan dan Pengujian dari 5 Terbaik Struktur Jaringan (Masih Data Transformasi)

**Struktur Jaringan**

***Train* Fungsi Aktivasi**

**Nilai MSE**
Pelatihan Pengujian
2 - 3 - 1 LM pureline - logsig 0.000185063 0.060886486
4 - 3 - 1 LM logsig - logsig 2.71E-05 0.004124855
5 - 3 - 1 LM pureline - logsig 0.000182859 **0.000141621**
7 - 5 - 1 LM logsig - logsig 2.65E-05 0.059298874
7 - 3 - 1 LM logsig - logsig 2.56E-05 0.009387664
7 - 7 - 1 LM logsig - logsig 2.47E-05 0.007155415
Untuk perbandingan nilai PE, MAPE, dan MSE setelah pengujian dari 5 besar struktur jaringan dapat dilihat di lampiran 4. Hasil ramalan setelah transformasi balik dengan struktur jaringan terpilih diperlihatkan oleh Gambar 4.6. Tabel hasil peramalan beserta nilai eror dan PE untuk metode jaringan syaraf tiruan *backpropagation* dalam meramalkan data IHSG dapat dilihat di Lampiran 3. Berdasarkan nilai eror dan PE diperoleh nilai MSE dan MAPE sebagai berikut
$%& __
'
_(')#_ _ #*_+
,- _ __55__
$/_& __(
0120
'
'
_3____4 _ ____4.

**4.3. Pembahasan**

Dari rancangan jaringan syaraf tiruan optimum diperoleh nilai MSE saat pelatihan sebesar 0.00018. Ini menggambarkan bahwa jaringan mampu mempelajari nilai taburan data dengan baik berdasarkan 725 pola data yang diberikan. Nilai MSE setelah pengujian yang diperoleh sebesar 0.00014. Nilai ini sedikit lebih kecil dibandingkan dengan MSE saat pelatihan. Hal ini bisa jadi disebabkan oleh keberhasilan pembagian komposisi data untuk pelatihan dan pengujian berhasil dengan baik. Dan transformasi yang dilakukan pada *range* [0.1:0.9] berdasarkan fungsi aktivasi *logsig* yang digunakan juga sudah tepat. Berdasarkan hasil ramalan data IHSG menggunakan model GARCH(1,1) menghasilkan nilai MAPE yang relatif kecil yaitu 0.0269, nilai ini tidak jauh beda dengan nilai MAPE untuk hasil peramalan dengan jaringan syaraf 45
tiruan *backpropagation* yang besarnya 0.0216. Hal ini berarti keakuratan ramalan model GARCH(1.1) untuk memprediksi IHSG hampir sama dengan jaringan syaraf tiruan *backpropagation*. Akan tetapi jika dilihat dari nilai MSE-nya, model GARCH(1,1) menghasilkan nilai MSE sebesar 1890.26 dimana nilai ini jauh lebih besar dibandingkan dengan nilai MSE yang dihasilkan oleh JST *backpropagation* yang hanya sebesar 1033.10. Hal ini berarti jaringan syaraf tiruan memiliki kemampuan meredam eror yang lebih baik. Untuk lebih jelasnya diperlihatkan oleh Gambar 4.6. Pada Gambar 4.6 terlihat bahwa grafik hasil peramalan menggunakan JST-*backpropagation* lebih mendekati grafik nilai aktualnya dibandingkan dengan grafik peramalan GARCH.