PENERAPAN JARINGAN SARAF TIRUAN UNTUK MEMPREDIKSI JUMLAH PENUMPANG BUS DENGAN METODE BACKPROPAGATION

(Studi Kasus Pada Terminal Bus Kabupaten Kebumen)

TUGAS AKHIR

Fana Pralita

06 611 004

JURUSAN STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2010

PENERAPAN JARINGAN SARAF TIRUAN UNTUK MEMPREDIKSI JUMLAH PENUMPANG BUS DENGAN METODE BACKPROPAGATION

(Studi Kasus PadaTerminal Bus Kabupaten Kebumen)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat

Untuk Memperoleh Gelar Sarjana Sains Bidang Statistika

Fana Pralita

06 611 004

JURUSAN STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2010

HALAMAN PENGESAHAN PEMBIMBING

TUGAS AKHIR

Judul : Penerapan Jaringan Saraf Tiruan untuk Memprediksi Jumlah Penumpang Bus dengan Metode Backpropagation (Studi Kasus Pada Terminal Bus Kabupaten Kebumen).

Nama Mahasiswa : Fana Pralita

Nomor Mahasiswa : 06 611 004

TUGAS AKHIR INI TELAH DIPERIKSA DAN

DISETUJUI UNTUK DIUJIKAN

Yogyakarta, 1 September 2010

Pembimbing

Prof. Akhmad Fauzy, S.Si., M.Si., Ph.D.

HALAMAN PENGESAHAN

TUGAS AKHIR

PENERAPAN JARINGAN SARAF TIRUAN UNTUK MEMPREDIKSI JUMLAH PENUMPANG BUS DENGAN METODE BACKPROPAGATION

(Studi Kasus Pada Terminal Bus Kabupaten Kebumen)

Nama : Fana Pralita

Nomor Mahasiswa : 06 611 004

TUGAS AKHIR INI TELAH DIUJIKAN

PADA TANGGAL 5 JANUARI 2011

Nama Penguji	Tanda Tangan
1. Prof. Akhmad Fauzy, S.Si., M.Si., Ph.D	
2. Drs. Supriyono, M.Sc	
3. RB.Fajriya Hakim, S.Si, M.Si	

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Islam Indonesia

Yandi Syukri, S.Si., M.Si., Apt.

HALAMAN PERSEMBAHAN

Karya sederhana ini aku persembahkan untuk :

Kedua orang tuaku, yang selalu berdo'a, mennyayangi ku, membimbing, memotivasi, membiayai dan berkorban untukku setiap saat.

Spesial Terima kasih yang teramat banyak untuk :

Sahabat terbaikku

MOTTO

"Jangan berharap orang lain mau menghargai Anda kalau Anda sendiri sungkan untuk menghargai diri Anda sendiri"

(Aris Ahmad Jaya)

"Kesuksesan dimulai ketika kita mulai menciptakan impian jauh kedepan, dan saat kita berkomitmen untuk mencapai impian itu, maka selanjutnya impian itu yang akan menjadi magnet dan menarik kita kesana"

(Aris Ahmad Jaya)

"Allah mengangkat orang-orang yang beriman dari golonganmu dan juga orang-orang yang dikaruniai ilmu pengetahuan hingga beberapa derajat"

(Q.S. Al-Mujaadilah; 58:11)

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Assalamu'alaikum Wr. Wb.

Alhamdulillahirabil'alamin puji syukur penulis panjatkan kehadirat Allah SWT atas segala limpahan hidayah, pertolongan dan karunia-Nya sehingga penulisan Tugas Akhir dengan judul **"PENERAPAN JARINGAN SARAF TIRUAN UNTUK MEMPREDIKSI JUMLAH PENUMPANG BUS DENGAN METODE BACKPROPAGATION (Studi Kasus Pada Terminal Bus Kabupaten Kebumen)"** dapat diselesaikan dengan baik. Sholawat beriring salam semoga selalu terlimpahkan pada Nabi Muhammad SAW, keluarga, sahabat hingga para pengikutnya semoga selalu istiqomah berpegang teguh pada ajarannya hingga akhir zaman.

Penulisan Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar sarjana (S1) Jurusan Statistika di Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Islam Indonesia, Yogyakarta.

Penghargaan yang tiada terkira kepada semua pihak yang telah memberikan andilnya dalam penyelesaian laporan Tugas Akhir ini. Melalui kesempatan ini penulis mengucapkan terima kasih kepada :

1. Allah SWT, yang telah melimpahkan rahmat, keberkahan, serta petunjuk kepada hamba-Nya.
2. Bapak Yandi Syukri, S.Si,. M.Si., Apt., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.

3. Ibu Kariyam, M.Si, selaku Ketua Jurusan Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.
4. Bapak Prof. Akhmad Fauzy, S.Si., M.Si., Ph.D., selaku dosen pembimbing yang telah meluangkan waktu untuk memberikan bimbingan, konsultasi, dukungan serta motivasi dalam menyelesaikan Tugas Akhir ini.
5. Kedua orangtuaku tercinta atas segala perhatian, doa, kesabaran dan kasih sayangnya yang tiada pernah terhenti kepada penulis.
6. Seluruh dosen statistika dan karyawan F-MIPA UII.
7. Semua teman-teman Statistika 2006 yang sudah membantu selama ini sampai selesai.
8. Semua pihak yang telah membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis menyadari sepenuhnya bahwa Tugas Akhir ini masih banyak memiliki kekurangan, meski segenap pengetahuan dan kemampuan telah penulis curahkan untuk itu. Oleh karenanya, kritik dan saran yang bersifat membangun dari berbagai pihak akan penulis terima dengan senang dan berbangga hati. Semoga Tugas Akhir ini dapat berguna bagi kita semua, Amin.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, September 2010

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang sebelumnya pernah diajukan untuk memperoleh gelar kesarjanaan disuatu perguruan tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 1 September 2010

Fana Pralita

PENERAPAN JARINGAN SARAF TIRUAN UNTUK MEMPREDIKSI JUMLAH PENUMPANG BUS DENGAN METODE BACKPROPAGATION

(Studi Kasus PadaTerminal Bus Kabupaten Kebumen)

ABSTRAK

Jaringan Saraf Tiruan (JST) adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan saraf biologi. Prediksi dengan JST dapat dilakukan, karena JST memiliki kemampuan untuk mengingat dan membuat generalisasi dari apa yang sudah dilakukan sebelumnya. Penerapan JST untuk prediksi jumlah penumpang bus di Kabupaten Kebumen dengan menggunakan metode Backpropagation berdasarkan pada data bulan Januari 2004 sampai Juli 2009. Dari hasil pelatihan dan pengujian data yang telah dilakukan, dihasilkan nilai prediksi untuk Agustus 2009 adalah sebesar 21857 orang.

Key word: Jaringan Saraf Tiruan (JST), Prediksi Jumlah Penumpang, Algoritma Backpropagation.

APPLICATION FOR PREDICTING NEURAL NETWORK TOTAL PASSENGER BUS WITH BACKPROPAGATION METHOD

Abstract

Artificial Neural Network (ANN) are information processing systems which have characteristics similar to biological neural networks. Prediction by ANN can be done, because ANN has the ability to remember and make generalizations from what has been done previously. Application of neural networks to predict the number of bus passengers in Kebumen using Backpropagation method based on data from January 2004 until July 2009. From the results of training and testing data that has been done, the resulting predictive value for August 2009 amounted to 21,857 persons.

Key word: Artificial Neural Network (ANN), Prediction Number of Passengers, Backpropagation Algorithm.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kabupaten Kebumen dengan jumlah penduduknya sekitar 1212809 jiwa (wikipedia, 2005) yang letak geografisnya sebelah utara berbatasan dengan wilayah Kabupaten Banjarnegara, bagian selatan berbatasan dengan Samudra Hindia, bagian barat berbatasan dengan Kabupaten Banyumas dan Kabupaten Cilacap, sedangkan bagian timur berbatasan dengan Kabupaten Wonosobo dan Kabupaten Purworejo.

Dalam pembahasan kali ini, akan lebih menjelaskan jumlah pengguna alat transportasi tersebut. Pada waktu-waktu tertentu jumlah penumpang pada suatu terminal bus akan mengalami lonjakan penumpang yang cukup banyak, akan tetapi di waktu lain pula dapat terjadi penurunan jumlah penumpang.

Hal inilah yang mendorong para statistisi untuk dapat membantu pihak pengelola terminal dalam mengeluarkan ataupun menetapkan kebijakan. Dalam hal ini kebijakan yang diambil adalah untuk dapat mengatasi lonjakan jumlah penumpang pada periode waktu tertentu. Kita tidak akan tahu dengan pasti berapa jumlah penumpang pada bulan ke depan, tetapi dengan alat bantu statistik ini, kita dapat mengetahui perkiraan jumlah penumpang di periode waktu ke depan.

Seiring dengan berkembangnya teknologi dan globalisasi, alat-alat statistik yang digunakan untuk peramalan juga sudah mulai berkembang. Seperti yang diungkapkan oleh Andrijasa dan Mistianingsih (2010) bahwa peramalan dapat

dilakukan dengan berbagai cara, salah satunya adalah dengan mengembangkan teknik kecerdasan buatan, yang dalam hal ini yang paling banyak digunakan untuk maksud di atas adalah menggunakan *Artificial Neural Network (ANN)*, atau yang lebih dikenal dengan istilah Jaringan Saraf Tiruan (JST).

Konsep dari JST untuk memprediksi jumlah penumpang dengan cara pola data jumlah penumpang periode masa lalu yang dimasukkan kedalam sistem dilakukan proses pelatihan menggunakan Jaringan Saraf Tiruan (JST) dan algoritma pembelajaran *Backropagation*. Setelah dilakukan proses pelatihan, sistem akan menghasilkan bobot yang akan digunakan untuk memprediksi jumlah penumpang pada periode selanjutnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan di atas, maka rumusan masalah dalam penulisan ini adalah penerapan Jaringan Saraf Tiruan untuk memprediksi berapa jumlah penumpang pada periode berikutnya.

1.3 Batasan Masalah

Berdasarkan rumusan masalah diatas, yang menjadi batasan masalah adalah:

1. Data yang digunakan adalah data yang berasal dari terminal bus Kabupaten Kebumen.
2. Data tersebut hanya berkisar antara bulan Januari 2004 sampai Juli 2009.

3. Metode yang digunakan dalam pembahasan ini adalah dengan menggunakan Jaringan Saraf Tiruan (JST).
4. Software yang digunakan untuk melatih dan menguji JST ini adalah dengan menggunakan software R 2.5 dan SPSS 16.

1.4 Tujuan

Berdasarkan latar belakang dan rumusan masalah di atas, maka penelitian ini dilaksanakan dengan tujuan untuk menerapkan Jaringan Saraf Tiruan dalam memprediksi jumlah penumpang pada periode selanjutnya.

1.5 Manfaat

Manfaat dari penelitian tugas akhir ini adalah:

1. Memberikan informasi yang dapat digunakan oleh pengelola terminal bus Kabupaten Kebumen untuk dapat menjadi suatu bahan pertimbangan bagi pengelola terminal uuntuk dapat memaksimalkan terminal tersebut.
2. memberikan pengetahuan kepada pihak lain tentang penerapan Jaringan Saraf Tiruan yang dapat digunakan untuk memprediksi jumlah penumpang.

BAB II

LANDASAN TEORI

2.1 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan saraf biologi.

JST dibentuk sebagai generalisasi model matematika dari jaringan saraf biologi bahwa (Siang, 2005):

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

2.2 Model Neuron

Neuron adalah unit pemroses informasi yang menjadi dasar dalam pengoperasian jaringan sararf tiruan.

2.2.1 Fungsi Aktivasi

Dalam jaringan saraf tiruan, fungsi aktivasi dipakai untuk menentukan keluaran suatu neuron. Fungsi yang digunakan dalam jaringan saraf tiruan dengan menggunakan metode *backpropagation* adalah menggunakan fungsi aktivasi sigmoid (Kusumadewi, 2003).

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

Fungsi sigmoid sering dipakai karena nilai fungsinya yang terletak antara 0 dan 1 dan dapat diturunkan dengan mudah.

$$f'(x) = f(x)\,(1-f(x))$$

2.2.2 Bias dan Threshold

Kadang-kadang dalam jaringan ditambahkan sebuah unit masukan yang nilainya selalu =1. Unit yang demikian itu disebut bias. Bias dapat dipandang sebagai sebuah input yang nilainya =1. Bias berfungsi untuk mengubah nilai *threshold* menjadi =0 (bukan =a). Jika melibatkan bias, maka keluaran unit penjumlahan adalah:

$$net = b + \sum_i x_i w_i$$

Fungsi aktivasi *threshold* menjadi:

$$f(\text{net}) = \begin{cases} 1 & \text{jika} \quad \text{net} \geq 0 \\ -1 & \text{jika} \quad \text{net} < 0 \end{cases}$$

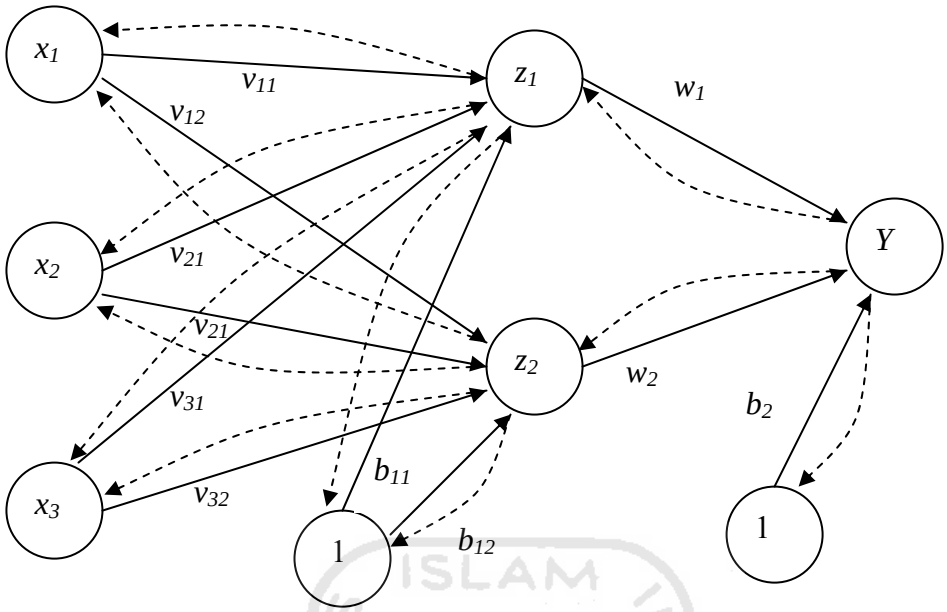
2.3 Back Propagation

Jaringan Saraf Tiruan (JST) dengan layar tunggal memiliki keterbatasan dalam pengenalan pola. Kelemahan ini bisa ditanggulangi dengan menambahkan satu atau beberapa layar tersembunyi diantara layar masukan dan keluaran. Meskipun penggunaan lebih dari satu layar tersembunyi memiliki kelebihan manfaat untuk beberapa kasus, tapi pelatihannya memerlukan waktu yang lama. Maka umumnya orang mulai mencoba dengan sebuah layar tersembunyi terlebih dahulu.

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

2.4 Arsitektur *Backpropagation*

Backpropagation memiliki beberapa unit yang ada dalam satu atau lebih layar tersembunyi. Gambar di bawah ini merupakan arsitektur backpropagation dengan n buah masukan (ditambah sebuah bias), sebuah layar tersembunyi yang terdiri dari p unit (ditambah sebuah bias), serta m buah unit keluaran. v_{ji} merupakan bobot garis dari unit masukan x_i ke unit layar tersembunyi z_j (v_{j0} merupakan bobot garis yang menghubungkan bias di unit masukan ke unit layar tersembunyi z_j). w_{kj} merupakan bobot dari unit layar tersembunyi z_j ke unit keluaran y_k (w_{k0} merupakan bobot dari bias di layar tersembunyi ke unit keluaran z_k).



Gambar 2.3 Arsitektur jaringan *backpropagation*

2.5 Pelatihan Standar *Backpropagation*

Pelatihan *Backpropagation* meliputi 3 fase. Fase pertama adalah fase maju. Pola masukan dihitung maju mulai dari layar masukan hingga layar keluaran menggunakan fungsi aktivasi yang ditentukan. Fase kedua adalah fase mundur. Selisih antara keluaran jaringan dengan target yang diinginkan merupakan kesalahan yang terjadi. Kesalahan tersebut dipropagasikan mundur, dimulai dari garis yang berhubungan langsung dengan unit-unit di layar keluaran. Fase ketiga adalah modifikasi bobot untuk menurunkan kesalahan yang terjadi.

Fase I: Propagasi maju

a. Sinyal masukan (x_i) dipropagasikan ke layar tersembunyi menggunakan fungsi aktivasi yang ditentukan.
b. Hasil dari setiap unit layar tersembunyi (z_j) tersebut selanjutnya dipropagasikan maju lagi ke layar tersembunyi berikutnya (jika ada) menggunakan fungsi aktivasi yang ditentukan hingga menghasilkan keluaran jaringan (y_k).
c. Berikutnya, keluaran jaringan (y_k) tersebut dibandingkan dengan target yang harus dicapai (t_k). Selisih t_k-y_k adalah error. Jika error melebihi batas toleransinya maka akan dilakukan propagasi mundur untuk memodifikasi bobot garis dalam jaringan. Jika error lebih kecil dari batas toleransi yang ditentukan maka iterasi dihentikan.

Fase II: Propagasi mundur

a. Faktor δ_k (k=1,2,...,m) yang dihitung sebagai error dari $t_k - y_k$ dipakai untuk mendistribusikan kesalahan di unit y_k ke semua unit tersembunyi dan mengubah bobot garis yang terhubung langsung dengan unit keluaran y_k.
b. Dengan cara yang sama dengan faktor δ_k, dihitung faktor δ_j di setiap unit di layar tersembunyi yang akan digunakan sebagai dasar perubahan bobot semua garis yang berasal dari unit yang tersembunyi di layar bawahnya. Demikian seterusnya sehingga semua faktor δ di unit tersembunyi yang berhubungan langsung dengan unit input dihitung.

Fase III: Perubahan bobot

a. Bobot akan dimodifikasi secara bersamaan setelah semua faktor δ dihitung. Modifikasi bobot menggunakan faktor δ dari neuron di layar atasnya. Sebagai contoh faktor δ_k yang ada di unit output digunakan sebagai dasar modifikasi bobot garis yang menuju ke layar keluaran.

Algoritma pelatihan untuk jaringan dengan sattu layar tersembunyi (dengan fungsi aktivasi sigmoid biner) adalah sebagai berikut:

Langkah 0 : Inisialisasi semua bobot dengan bilangan random kecil

Langkah 1 : Jika kondisi penghentian belum terpenuhi, lakukan langkah 2 -9

Langkah 2 : Untuk setiap pasang data pelatihan, lakukan langkah 3-9

Fase I : Propagasi maju

Langkah 3 : Tiap unit input menerima sinyal dan kemudian meneruskan-nya ke unit tersembunyi diatasnya

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

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

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

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

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

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

Fase II: Propagasi mundur

Langkah 6 : Hitung faktor δ unit keluaran berdasarkan error di setiap unit keluaran y_k (k = 1,2,...,m)

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

δ_k merupakan error yang akan dipakai dalam modifikasi bobot layar di bawahnya (langkah 7)

Hitung suku perubahan bobot w_{kj} (yang akan dipakai nanti untuk merubah bobot w_{kj}) dengan laju percepatan α.

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

Langkah 7 : Hitung faktor δ unit tesembunyi berdasarkan error di setiap unit tersembunyi z_j (j = 1,2,...,p)

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

Faktor δ unit tersembunyi:

$$\delta\_net_j f'(z\_net_j) = \delta\_net_j z_j (1 - z_j)$$

Hitung suku perubahan bobot v_{ji} (yang akan dipakai nanti memodifikasi bobot v_{ji})

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

Fase III : Modifikasi Bobot

Langkah 8 : Hitung semua modifikasi bobot

Perubahan bobot garis yang menuju ke unit keluaran:

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

Perubahan bobot garis yang menuju ke unit tersembunyi :

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

Setelah pelatihan selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola. Dalam hal ini, hanya propagasi maju (langkah 4 dan 5) saja yang dipakai untuk menentukan keluaran jaringan (Siang, 2005).

2.6 Tentang R

R adalah bahasa untuk komputasi statistik dan grafik yang mirip dengan bahasa S yang dikembangkan di Bell Laboratories (dahulu AT & T, sekarang Lucent Technologies) oleh John Chambers. Salah satu kelebihan dari R adalah dapat membuat grafik dengan baik, termasuk simbol dan rumus matematika yang diperlukan (r-project, 2010).

R merupakan integrasi dari fasilitas perangkat lunak untuk manipulasi data, perhitungan, dan tampilan grafis. Hal ini meliputi:

- Menangani data secara efektif dan menyimpannya,
- Sebuah suite dari operator untuk perhitungan d array dalam matriks khusus,
- Terintegrasi untuk analisis data,
- Fasilitas grafis untuk analisis data dan tampilan layar,

- Sebuah bahasa pemrograman yang berkembang dengan baik, sederhana dan efektif yangg mencakup loop, conditional, fungsi rekursif yang ditentukan pengguna dan input serta outputnya.

2.7 Package ”AMORE”

Paket AMORE digunakan untuk jaringan saraf yang berbeda dengan yang lain yang telah ditulis oleh Manuel Castejón Limas, Joaquín B. Ordieres Meré, Ana González Marcos, Francisco Javier Martínez de Pisón Ascacibar, Alpha V. Pernía Espinoza, Fernando Alba Elías yang dapat melakukan pelatihan data berupa jaringan feedforward multilayer.

Dalam R dengan menggunakan package AMORE, standar jaringan saraf tiruan yang dituliskan adalah:

```
> net <- newff(n.neurons=c(1,2,1),learning.rate.global=0.1,
      momentum.global=0.5, error.criterium="LMS", Stao=NA,
      hidden.layer="sigmoid", output.layer="sigmoid",
      method="ADAPTgdwm")
```

Maksud dari fungsi diatas adalah:

newff	: fungsi untuk membuat jaringan saraf umpan maju.
n.neurons=c(1,2,1)	: argumen untuk menyatakan jumlah neuron di layar input (1 buah), di layar tersembunyi (2 buah), di layar output (1 buah).
learning.rate=0.1	: tingkat pembelajaran yang digunakan pada setiap neuron.

momentum.global=0.5 : momentum yang digunakan pada setiap neuron.

error.criterium : argumen yang digunakan untuk menghitung nilai kesalahan antara hasil keluaran pada layar output yang dibandingkan dengan targetnya.

Stao=NA : parameter Stao untuk kriteria error TAO, jika argumen error.criterium tidak memiliki TAO error, maka untuk argumen Stao kita letakkan pilihan NA (Not Available).

hidden layer dan output layer : argument yang diberikan pada jaringan untuk fungsi aktivasi yang akan digunakan pada masing-masing layar baik pada layar tersembunyi maupun layar output.

Metode yang digunakan untuk pelatihan ada beberapa macam, diantaranya: ADAPTgd, ADAPTgdwm, BACTHgd, BACTHgdwm. Dalam pembahasan kali ini menggunakan metode ADAPTgdwm dengan beberapa istilah variabelnya:

- delta : merupakan elemen untuk koreksi dari derivatif nilai deltaE pada bobot dan bias.
- learning.rate : tingkat pembelajaran yang digunakan pada setiap neuron.
- momentum :mirip dengan variabel learning.rate dan momentum.global, tapi sekali lagi, metode pelatihan yang baru dapat menggunakan variabel ini untuk menetapkan nilai momentum tingkat yang berbeda untuk setiap neuron.
- Former.weight.change : berisi nilai perubahan berat pada setiap iterasi.

- Former.bias.change : berisi nilai perubahan bias pada setiap iterasi.

2.8 Least Mean Square

Metode least mean square ini memiliki prosedur hitung mirip dengan sum square error dengan rumus (Minsky dan Papert, 2010):

$E = \sum_p Ep = \sum_p \sum_i (t_{ip} - o_{ip})^2$, dimana, $o_p = \sum_j w_{ij} i_j + bias_i$

Yakni t_{ip} merupakan nilai target yang diinginkan,

o_{ip} merupakan output keluaran,

w_{ij} merupakan nilai bobot yang digunakan,

i merupakan nilai input

2.9 Analisis Runtun Waktu

Data time series (runtun waktu) adalah jenis data yang dikumpulkan menurut urutan waktu dalam suatu urutan waktu tertentu. Dalam kasus diskrit, frekuensi dapat berupa misalnya detik, menit, jam, hari, minggu, bulan atau tahun (Rosadi, 2006). Sehingga analisis runtun waktu adalah analisis data yang berupa urutan waktu dalam suatu urutan waktu tertentu.

Dengan analisis runtun waktu, kita bisa meramalkan kejadian di masa datang atas dasar serangkaian data masa lalu, yang merupakan hasil observasi berbagai variabel menurut waktu dan digambarkan dalam bentuk grafik.

2.10 ARIMA (*Autoregressive Integrated Moving Average*)

Model – model *Autoregressive Integrated Moving Average* (ARIMA) telah dipelajari secara mendalam oleh George Box dan Gwilym Jenkins (1976), dan nama mereka sering disinonimkan dengan proses ARIMA yang diterapkan

untuk analisis deret berkala, peramalan dan pengendalian. Model Autoregressive (AR) pertama kali diperkenalkan oleh Yule (1926) dan kemudian dikembangkan oleh Walker (1931), sedangkan model Moving Average (MA) pertama kali digunakan oleh Slutzky (1937). Akan tetapi Wold-lah (1938) yang menghasilkan dasar – dasar teoritis dari proses kombinasi ARMA. Wold membentuk model ARMA yang dikembangkan pada tiga arah identifikasi efisiensi dan prosedur penaksiran (untuk proses AR, MA, dan ARMA campuran), perluasan dari hasil tersebut untuk mencakup deret berkala musiman (*seasonal time series*) dan pengembangan sederhana yang mencakup proses – proses non stasioner ARIMA.

2.11 Langkah – langkah Analisis Runtun Waktu

2.11.1 Plot Data

Langkah awal sebelum kita melakukan analisis runtun waktu adalah melakukan plot dari data yang telah kita miliki. Hasil plot dari data inilah yang akan membantu kita untuk menentukan proses analisis selanjutnya.

2.11.2 Stasioneritas

Stasioneritas berarti bahwa tidak terdapat pertumbuhan atau penurunan pada data. Data secara kasarnya harus horizontal sepanjang sumbu waktu. Dengan kata lain, fluktuasi data berada di sekitar suatu nilai rata – rata yang konstan, tidak tergantung pada waktu dan ragam dari fluktuasi tersebut pada pokoknya tetap konstan setiap waktu. (Makridakis dkk, 1999)

Proses runtun waktu yang stasioner, hasil plot datanya menunjukkan bahwa semua data berkisar di titik yang sama (mean) untuk setiap t dan fluktuasinya konstan. Bentuk visual dari suatu plot deret berkala seringkali cukup

untuk meyakinkan para peramal bahwa data tersebut adalah stasioner atau tidak stasioner.

Stasioneritas dapat dibagi menjadi dua, yaitu :

a. Stasioneritas dalam hal varians

Suatu data runtun waktu dikatakan stasioner dalam hal varians, jika struktur data dari waktu ke waktu mempunyai fluktuasi yang tetap / konstan, tidak berubah – ubah atau tidak ada perubahan variasi besarnya fluktuasi. Secara visual untuk melihat suatu data sudah stasioner dalam hal varians dapat dibantu dengan menggunakan plot dari data. Apabila hasil dari plot tersebut menunjukkan bahwa data tidak stasioner dalam varians, maka dapat dilakukan dengan cara mentransformasi data tersebut.

b. Stasioneritas dalam hal mean

Suatu data dapat dikatakan stasioner dalam hal mean, jika tidak ada unsur trend di dalam data. Untuk melihat suatu data sudah stasioner dalam mean dapat dilakukan dengan melihat plot dari data tersebut. Apabila hasil dari plot tersebut menunjukkan belum stasioner, maka dapat dilakukan *differencing* (pembedaan). Differencing dilakukan dengan mengambil selisih antar data time seriesnya yaitu dengan rumus $d_t = y_{t+1} - y_t$.

2.11.3 Pemodelan Awal

Setelah dilakukan plot data dan stasioneritas data, maka langkah selanjutnya adalah membuat model awal dari ARIMA. Model awal inilah yang

nantinya akan menentukan model mana yang akan kita gunakan. ARIMA (*Autoregressive Integrated Moving Average)*, dari nama ARIMA sendiri, artinya dalam analisis ini mencakup pemodelan AR (*Autoregressive*), *Integrated* dengan malakukan proses *differencing,* dan MA (*Moving Average*). Model awal ARIMA ini dapat di cari dengan cara membuat correlogram atau plot dari data yang sudah stasioner dalam varians maupun dalam mean.

2.11.4 Overfitting Model ARIMA

Salah satu prosedur pemeriksaan diagnostic (diagnostic checking) yanng dikemukakan oleh Box dan Jenkins adalah *overfitting*. Overfitting yaitu menggunakan beberapa parameter lebih banyak dari pada yang diperlukan atau memilih AR orde kedua bilamana AR orde pertama telah ditetapkan. Alternatif model lainnya yang akan dicoba haruslah model yang parameternya telah memenuhi syarat dalam uji overall maupun uji parsial. Dalam overfitting terdapat suatu prinsip *parsimony*, yaitu memilih model yang jumlah parameternya lebih sedikit (Makridakis dkk, 1999).

2.11.5 *Diagnostic Checking*

Setelah diperoleh estimasi terbaik untuk parameter-parameter dalam model, langkah selanjutnya adalah melakukan pengujian asumsi-asumsi atau *diagnostic checking* untuk menguji kelayakan model. Jika model tidak layak, disarankan untuk modifikasi model. Pendekatan yang digunakan dalam *diagnostic checking* ini adalah analisis residual dan *overfitting*.

Langkah-langkah analisis residual adalah sebagai berikut :

a. Memeriksa "plot of residual overtime". Model dikatakan layak jika residualnya berkisar dititik nol dan tidak membentuk tren juga variansi dari plot homoskedastik.
b. Memeriksa kenormalan dari residual dengan membuat histogram dari residual yang telah distandarisasi. Model dikatakan layak jika histogram residual mempunyai kecendrungan membentuk distribusi normal simetris di titik nol.
c. Memeriksa independensi dari a_t dalam model dengan membuat plot fungsi autokorelasi. Model dikatakan layak jika fungsi autokorelasi untuk lag 1,2,… dst tidak secara signifikan berbeda dengan nol. Jika ada 1 lag yang secara signifikan berbeda dengan nol, bukan berarti e_t tidak independen. Ini dikarenakan nilai korelasi tersebut merupakan estimasi, jadi akan selalu mengandung resiko kesalahan.

2.11.6 Pemilihan Model Terbaik

Kriteria yang digunakan untuk pemilihan model ARIMA yang terbaik setelah dilakukan identifikasi model dan *diagnostic checking* adalah:

Sum Square of Residual **(SSR)** adalah nilai jumlahan dari kuadrat residual/error

$$SSR = \sum_{i=1}^{n} e^2{}_i \text{ , e = residual/error}$$

2.11.7 Peramalan/*Forecasting*

Langkah terakhir dalam proses runtun waktu adalah inferensi/peramalan runtun waktu di masa mendatang berdasarkan tingkah geraknya di masa lalu (data sebelumnya).

BAB III

METODOLOGI PENELITIAN

3.1 Populasi Penelitian

Data yang digunakan merupakan data sekunder yang diambil dari Dinas Perhubungan Kabupaten Kebumen yang lebih dispesifikkan pada terminal bus Kabupaten Kebumen. Data tersebut adalah tentang jumlah penumpang di terminal bus Kabupaten Kebumen dari bulan Januari 2004 sampai Juli 2009.

3.2 Tempat dan Waktu Penelitian

Pengambilan data yang dilakukan di Dinas Perhubungan Kabupaten Kebumen pada tanggal 1-30 Juli 2009 pada saat melakukan Kerja Praktek di dinas tersebut yang di tempatkan di Terminal Bus Kabupaten Kebumen.

3.3 Variabel Penelitian

Dalam hal ini, variabel yang digunakan adalah jumlah penumpang. Untuk menghitung jumlah penumpang ini, di terminal bus tersebut sudah ada petugas yang khusus menangani tentang jumlah penumpang.

3.4 Alat dan Cara Organisir Data

3.4.1 Alat Organisir Data

Dalam analisis menggunakan Jaringan Saraf Tiruan (JST) ini ada beberapa alat yang digunakan untuk mengerjakan data atau organisir data, diantaranya adalah:

a. SPSS 16

   Software ini digunakan untuk melakukan proses transformasi data agar data jumlah penumpang tersebut berkisar antara 0 dan 1.

b. R 2.5

   Software R 2.5 ini digunakan untuk melakukan proses pelatihan, pengujian, serta prediksi data, sehingga dapat dihasilkan nilai prediksi datanya.

c. E-Views

   Sedangkan untuk software E-Views ini digunnakan untuk menghitung nilai prediksi data dengan analisis runtun waktu (time series). Kemudian hasil prediksi dengan E-Views ini akan dibandingkan dengan nilai prediksi data menggunakan Jaringan Saraf Tiruan yang telah dihitung menggunakan software R 2.5.

3.4.2 Cara Organisir Data

Dalam analisis menggunakan Jaringan Saraf Tiruan (JST) ini ada beberapa langkah yang harus dilakukan terlebih dahulu sebelum masuk ke tahap pelatihan dan pengujian data. Fungsi aktivasi yang digunakan adalah fungsi sigmoid, oleh

karena itu, data jumlah penumpang ini harus dilakukan transformasi terlebih dahulu agar nilainya berkisar antara 0 dan 1.

a. Transformasi data

Transformasi data ini berfungsi untuk mengubah data jumlah penumpang yang bernilai ribuan menjadi nilai yang berkisar antara 0 dan 1. Hal ini dilakukan mengingat fungsi aktivasi yang digunakan adalah fungsi sigmoid. Rumus yang digunakan untuk melakukan transformasi adalah $x' = \frac{0.8(x-a)}{b-a} + 0.1$, dimana:

x' merupakan nilai hasil transformasi,

x merupakan data jumlah penumpang,

a merupakan nilai terkecil dari data jumlah penumpang tersebut,

b merupakan nilai terbesar dari data jumlah penumpang.

Untuk melakukan proses transformasi ini digunakan software SPSS 16.

b. Proses Pelatihan Data

Pada proses pelatihan ini fungsinya adalah untuk melatih jaringan agar memiliki pola dan bobot yang diinginkan. Pada setiap kali pelatihan, suatu input diberikan ke jaringan. Jaringan akan memproses dan mengeluarkan keluaran. Selisish antara keluaran jaringan dan target merupakan kesalahan yang terjadi. Jaringan akan memodifikasi bobot sesuai dengan kesalahan tersebut. (Siang, 2005). Proses pelatihan ini akan berhenti jika nilai pelatihannya sudah mendekati nilai target yang diinginkan.

c. Proses Pengujian Data

Pada proses pengujian dilakukan setelah proses pelatihan selesai yang fungsinya adalah untuk menguji apakah jaringan menghasilkan output sesuai dengan yang diinginkan pada waktu input yang belum pernah dipelajari oleh jaringan dimasukkan. (Puspitaningrum, 2006)

d. Prediksi Data

Tahap terakhir dari proses ini adalah melakukan prediksi pada periode berikutnya berdasarkan periode yang sebelumnya. Hasil prediksi data ini akan dibandingkan pula dengan nilai prediksi menggunakan analisis runtun waktu menggunakan software E-Views.

3.4.3 Pengujian Dibandingkan dengan Time Series

Langkah-langkah dalam peramalan dengan menggunakan analisis runtun waktu adalah:

3.4.3.1 Plot Data

Langkah awal sebelum kita melakukan analisis runtun waktu adalah melakukan plot dari data yang telah kita miliki. Hasil plot dari data inilah yang akan membantu kita untuk menentukan proses analisis selanjutnya.

3.4.3.2 Stasioneritas

Stasioneritas berarti bahwa tidak terdapat pertumbuhan atau penurunan pada data. Data secara kasarnya harus horizontal sepanjang sumbu waktu. Dengan kata lain, fluktuasi data berada di sekitar suatu nilai rata – rata yang konstan, tidak

tergantung pada waktu dan ragam dari fluktuasi tersebut pada pokoknya tetap konstan setiap waktu. (Makridakis, 1999)

Stasioneritas dapat dibagi menjadi dua, yaitu:

a. Stasioneritas dalam hal varians

Suatu data runtun waktu dikatakan stasioner dalam hal varians, jika struktur data dari waktu ke waktu mempunyai fluktuasi yang tetap / konstan, tidak berubah – ubah atau tidak ada perubahan variasi besarnya fluktuasi. Secara visual untuk melihat suatu data sudah stasioner dalam hal varians dapat dibantu dengan menggunakan plot dari data. Apabila hasil dari plot tersebut menunjukkan bahwa data tidak stasioner dalam varians, maka dapat dilakukan dengan cara mentransformasi data tersebut.

b. Stasioneritas dalam hal mean

Suatu data dapat dikatakan stasioner dalam hal mean, jika tidak ada unsur trend di dalam data. Untuk melihat suatu data sudah stasioner dalam mean dapat dilakukan dengan melihat plot dari data tersebut. Apabila hasil dari plot tersebut menunjukkan belum stasioner, maka dapat dilakukan *differencing* (pembedaan). Differencing dilakukan dengan mengambil selisih antar data time seriesnya yaitu dengan rumus $d_t = y_{t+1} - y_t$.

3.4.3.3 Pemodelan Awal

Setelah dilakukan plot data dan stasioneritas data, maka langkah selanjutnya adalah membuat model awal dari ARIMA.

3.4.3.4 Overfitting Model ARIMA

Overfitting yaitu menggunakan beberapa parameter lebih banyak dari pada yang diperlukan atau memilih AR orde kedua bilamana AR orde pertama telah ditetapkan. Alternatif model lainnya yang akan dicoba haruslah model yang parameternya telah memenuhi syarat dalam uji overall maupun uji parsial. Dalam overfitting terdapat suatu prinsip *parsimony*, yaitu memilih model yang jumlah parameternya lebih sedikit (Makridakis, 1999).

3.4.3.5 *Diagnostic Checking*

Setelah diperoleh estimasi terbaik untuk parameter-parameter dalam model, langkah selanjutnya adalah melakukan pengujian asumsi-asumsi atau *diagnostic checking* untuk menguji kelayakan model. Jika model tidak layak, disarankan untuk modifikasi model. Pendekatan yang digunakan dalam *diagnostic checking* ini adalah analisis residual dan *overfitting*.

Langkah-langkah analisis residual adalah sebagai berikut:

d. Memeriksa "plot of residual overtime". Model dikatakan layak jika residualnya berkisar dititik nol dan tidak membentuk tren juga variansi dari plot homoskedastik.
e. Memeriksa kenormalan dari residual dengan membuat histogram dari residual yang telah distandarisasi. Model dikatakan layak jika histogram residual mempunyai kecendrungan membentuk distribusi normal simetris di titik nol.

f. Memeriksa independensi dari a_t dalam model dengan membuat plot fungsi autokorelasi. Model dikatakan layak jika fungsi autokorelasi untuk lag 1,2,… dst tidak secara signifikan berbeda dengan nol. Jika ada 1 lag yang secara signifikan berbeda dengan nol, bukan berarti e_t tidak independen. Ini dikarenakan nilai korelasi tersebut merupakan estimasi, jadi akan selalu mengandung resiko kesalahan.

3.3.6 Pemilihan Model Terbaik

Setelah semua model dilakukan **diagnostic checking,** ***maka langkah selanjutnya adalah memilih model yang terbaik yang akan digunakan untuk proses peramalan atau*** **forecasting.**

3.3.7 Peramalan/*Forecasting*

Langkah terakhir dalam proses runtun waktu adalah inferensi/peramalan runtun waktu di masa mendatang berdasarkan tingkah geraknya di masa lalu (data sebelumnya).

BAB IV
PEMBAHASAN

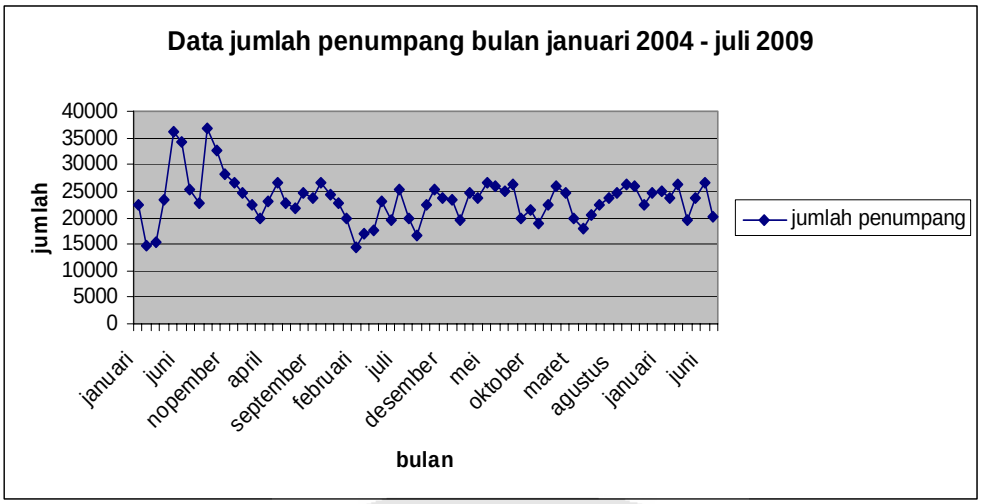
Berikut merupakan data jumlah penumpang Terminal Bus Kota Kebumen bulan Januari 2004 sampai Juli 2009 :

Tabel 4.1 Data jumlah penumpang terminal bus Kebumen periode Januuari 2004 sampai Juli 2009.

Tahun	Bulan	Jumlah Penumpang	Tahun	Bulan	Jumlah Penumpang
2004	Januari	22425		Oktober	22530
	Februari	14723		November	25432
	Maret	15235		Desember	23726
	April	23476	2007	Januari	23364
	Mei	36255		Februari	19476
	Juni	34213		Maret	24752
	Juli	25242		April	23524
	Agustus	22634		Mei	26476
	September	36723		Juni	25833
	Oktober	32524		Juli	24926
	November	28233		Agustus	26153
	Desember	26720		September	19834
2005	Januari	24523		Oktober	21522
	Februari	22425		November	18953
	Maret	19763		Desember	22425
	April	23152	2008	Januari	25763
	Mei	26476		Februari	24625
	Juni	22825		Maret	19753
	Juli	21676		April	17826
	Agustus	24582		Mei	20423
	September	23696		Juni	22526
	Oktober	26523		Juli	23763
	November	24426		Agustus	24515
	Desember	22825		September	26233
2006	Januari	19758		Oktober	25867
	Februari	14255		November	22455
	Maret	16924		Desember	24504
	April	17525	2009	Januari	24932
	Mei	23159		Februari	23526
	Juni	19674		Mareet	26115
	Juli	25422		April	19423
	Agustus	19755		Mei	23526
	September	16546		Juni	26424
				Juli	20272

Sumber : Terminal Bus Kabupaten Kebumen bulan Januari 2004 sampai Juli 2009.

Dari data di atas dapat dibuat grafik untuk dapat mempermudah membaca, yaitu:



Gambar 4.1 Grafik jumlah penumpang terminal bus Kebumen

4.1 Transformasi Data

Karena fungsi aktivasi yang digunakan adalah fungsi sigmoid dengan range [0,1], maka data diatas harus dilakukan transformasi terlebih dahulu supaya data tersebut berada pada interval [0,1].

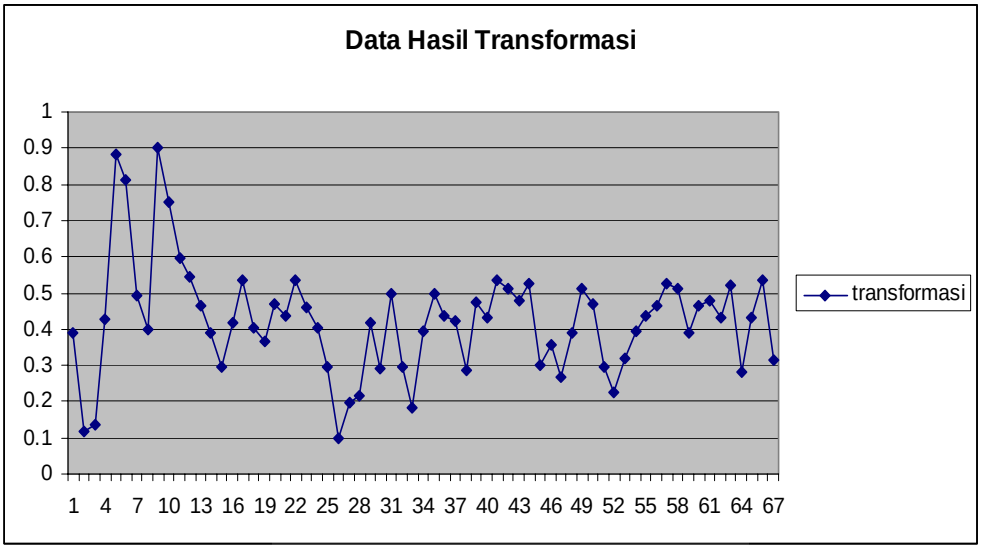
Jika a adalah data minimum dan b adalah data maksimum, maka transformasi linear yang digunakan untuk mentransformasi data di atas agar berada pada range [0,1] adalah:

$$x' = \frac{0.8(x-a)}{b-a} + 0.1$$

Dalam tabel 4.1 nilai minimum a = 14255 dan nilai maksimum b = 36723. Dengan transformasi ini maka data terkecil menjadi 0.1 dan data terbesar menjadi 0.9 yang selanjutnya dapat digunakan sebagai data pelatihan *Backpropagation*. Berikut merupakan hasil dari transformasi data jumlah penumpang:

Tabel 4.2 Hasil transformasi data jumlah penumpang

No	Data	Transformasi		No	Data	Transformasi
1	22425	0.390902617		35	25432	0.497970447
2	14723	0.116663699		36	23726	0.437226277
3	15235	0.134894072		37	23364	0.424336835
4	23476	0.428324729		38	19476	0.285899947
5	36255	0.883336301		39	24752	0.473758234
6	34213	0.810628449		40	23524	0.430033826
7	25242	0.49120527		41	26476	0.535143315
8	22634	0.398344312		42	25833	0.512248531
9	36723	0.9		43	24926	0.479953712
10	32524	0.750489585		44	26153	0.523642514
11	28233	0.5977034		45	19834	0.298646965
12	26720	0.543831227		46	21522	0.358750223
13	24523	0.465604415		47	18953	0.267277906
14	22425	0.390902617		48	22425	0.390902617
15	19763	0.296118925		49	25763	0.509756098
16	23152	0.416788321		50	24625	0.469236247
17	26476	0.535143315		51	19753	0.295762863
18	22825	0.405145095		52	17826	0.227149724
19	21676	0.364233577		53	20423	0.319619014
20	24582	0.467705181		54	22526	0.394498843
21	23696	0.436158092		55	23763	0.438543707
22	26523	0.536816806		56	24515	0.465319566
23	24426	0.462150614		57	26233	0.526491009
24	22825	0.405145095		58	25867	0.513459142
25	19758	0.295940894		59	22455	0.391970803
26	14255	0.1		60	24504	0.464927897
27	16924	0.195032936		61	24932	0.480167349
28	17525	0.216432259		62	23526	0.430105038
29	23159	0.417037565		63	26115	0.522289478
30	19674	0.292949973		64	19423	0.284012818
31	25422	0.497614385		65	23526	0.430105038
32	19755	0.295834075		66	26424	0.533291793
33	16546	0.181573794		67	20272	0.314242478
34	22530	0.394641268				



Gambar 4.2 Transformasi data jumlah penumpang terminal bus Kebumen

4.2 Pelatihan Data

Pada proses pelatihan ini fungsinya adalah untuk melatih jaringan agar memiliki pola dan bobot yang diinginkan. Pada setiap kali pelatihan, suatu input diberikan ke jaringan. Jaringan akan memproses dan mengeluarkan keluaran. Selisih antara keluaran jaringan dan target merupakan kesalahan yang terjadi. Jaringan akan memodifikasi bobot sesuai dengan kesalahan tersebut. (Siang,2005) Dengan menggunakan software R, maka setelah dilakukan pelatihan pertama terhadap 48 data hasil transformasi adalah:

Yang menjadi input dalam proses pelatihan ini adalah data bulan Januari 2004 sampai Desember 2007 dengan nama variabelnya adalah input.

```
>input[,1]<-c(0.39090, 0.11666, 0.13489, 0.42832, 0.88333, 0.81062, 0.49120,
0.39834, 0.9, 0.75049, 0.59770, 0.54383, 0.46560, 0.39090, 0.29611, 0.41678,
0.53514, 0.40514, 0.36423, 0.46770, 0.43615, 0.53681, 0.46215, 0.40514,
0.29594, 0.1, 0.19503, 0.21643, 0.41703, 0.29294, 0.49761, 0.29583, 0.18157,
0.39464, 0.49797, 0.43722, 0.42433, 0.28589, 0.47375, 0.43003, 0.53514,
0.51224, 0.47995, 0.52364, 0.29864, 0.35875, 0.26727, 0.39090)
```

Dengan targetnya adalah data dari bulan Januari 2005 sampai Desember 2008, dengan nama variabelnya adalah target:

```
> target[,1]<-c(0.46560, 0.39090, 0.29611, 0.41678, 0.53514, 0.40514, 0.36423,
0.46770, 0.43615, 0.53681, 0.46215, 0.40514, 0.29594, 0.1, 0.19503, 0.21643,
0.41703, 0.29294, 0.49761, 0.29583, 0.18157, 0.39464, 0.49797, 0.43722,
0.42433, 0.28589, 0.47375, 0.43003, 0.53514, 0.51224, 0.47995, 0.52364,
0.29864, 0.35875, 0.26727, 0.39090, 0.50975, 0.46923, 0.29576, 0.22714,
0.31961, 0.39449, 0.43854, 0.46531, 0.52649, 0.51345, 0.39197, 0.46492)
```

Pelatihan ini dilakukan dengan fungsi *newff*, yang merupakan perintah untuk melakukan proses jaringan saraf tiruan. Dalam perintah ini berisi jumlah neuron yang digunakan, learning rate, fungsi aktivasi, dan error seperti dibawah ini:

```
> net1 <- newff(n.neurons=c(1,2,1), learning.rate.global=0.2,
+ momentum.global=0.5, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid",method="ADAPTgdwm")
```

Setelah perintah ini dikerjakan, proses selanjutnya adalah melakukan pelatihan untuk data tersebut dengan perintah *train* yang berisi input, target, error dan hasil dari perintah *newff* sebelumnya.

```
> result<-train(net,input,target,error.criterium="LMS",
+ report=TRUE,show.step=300,n.shows=10)
```

Hasil dari proses pelatihan diatas dengan nama variable y adalah:

net1	**transformasi net1**
0.3981261	22627.87152
0.3728592	21918.25063
0.3745041	21964.44765
0.401621	22726.02579
0.4429995	23888.14096
0.436656	23709.98376
0.4074912	22890.89035
0.3988208	22647.38217
0.4444325	23928.38676
0.4313061	23559.73182

0.4173812	23168.651
0.4123908	23028.49562
0.4051026	22823.80652
0.3981261	22627.87152
0.3892974	22379.91748
0.4005431	22695.75296
0.4115832	23005.81417
0.3994559	22665.21895
0.3956372	22557.97076
0.4052986	22829.31118
0.4023524	22746.56715
0.4117384	23010.17296
0.4047805	22814.76034
0.3994559	22665.21895
0.3892816	22379.47374
0.3713621	21876.20458
0.379976	22118.12596
0.381938	22173.22873
0.4005665	22696.41015
0.3890032	22371.65487
0.4080889	22907.67676
0.3892714	22379.18727
0.3787457	22083.57298
0.3984753	22637.6788
0.4081225	22908.62041
0.4024524	22749.37565
0.4012483	22715.55851
0.3883494	22353.2929
0.4058633	22845.17078
0.4017808	22730.51377
0.4115832	23005.81417
0.4094521	22945.96223
0.4064419	22861.42076
0.4105135	22975.77165
0.3895322	22386.51184
0.3951261	22543.61652
0.386625	22304.86313
0.3981261	22627.87152

Karena hasil pelatihan tersebut masih belum mendekati nilai target yang diinginkan, maka dilakukan pelatihan lagi sampai mendekati nilai target yang diinginkan dengan memperkecil nilai *learning rate*nya.

```
> net2 <- newff(n.neurons=c(1,2,1), learning.rate.global=0.1,
+  momentum.global=0.5, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid", method="ADAPTgdwm")
```

Net2 merupakan nama variable untuk proses pelatihan yang kedua. Hasil pelatihan selanjutnya dengan mengurangi atau memperkecil nilai *learning rate* seperti pada perintah diatas adalah:

net2	**transformasi net2**
0.3952231	22546.34076
0.3678114	21776.48317
0.3696266	21827.46306
0.398956	22651.17926
0.4430349	23889.13517
0.4362256	23697.89598
0.405208	22826.76668
0.3959659	22567.2023
0.4445796	23932.51807
0.4305124	23537.44075
0.415712	23121.77152
0.4104144	22972.98842
0.4026663	22755.38304
0.3952231	22546.34076
0.3857428	22280.08654
0.3978056	22618.87028
0.4095566	22948.89711
0.3966445	22586.26078
0.3925584	22471.50266
0.402875	22761.24438
0.3997361	22673.08837
0.4097215	22953.52833
0.4023234	22745.75269
0.3966445	22586.26078
0.3857258	22279.60909
0.3661546	21729.95194
0.3756286	21996.02923
0.3777682	22056.1199
0.3978306	22619.5724
0.3854255	22271.17517
0.4058435	22844.6147
0.3857148	22279.30016

0.3742838	21958.26052
0.3955965	22556.8277
0.4058792	22845.61733
0.3998427	22676.08223
0.3985584	22640.01266
0.3847197	22251.35277
0.4034761	22778.12627
0.3991264	22655.96494
0.4095566	22948.89711
0.4072926	22885.31267
0.4040917	22795.41539
0.4084204	22916.98693
0.3859961	22287.20047
0.3920105	22456.11489
0.3828557	22199.00233
0.3952231	22546.34076

Karena hasil pelatihan kedua juga masih belum mendekati nilai target yang diinginkan, maka dilakukan pelatihan ketiga dengan memperkecil nilai *learning rate*nya lagi, hasil dari pelatihan tersebut adalah:

```
> net3 <- newff(n.neurons=c(1,2,1), learning.rate.global=0.01,
+ momentum.global=0.5, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid", method="ADAPTgdwm")
```

net3	**transformasi net3**
0.3945726	22528.07147
0.3818631	22171.12516
0.3827092	22194.88788
0.3962949	22576.44227
0.4165899	23146.42734
0.4134535	23058.34155
0.3991768	22657.38043
0.3949154	22537.69901
0.4173019	23166.42386
0.4108236	22984.48081
0.4040131	22793.20791
0.4015746	22724.72264
0.3980056	22624.48728
0.3945726	22528.07147
0.390191	22405.01424

0.3957643	22561.54037
0.4011797	22713.63187
0.3952286	22546.49523
0.3933422	22493.51569
0.3981017	22627.18624
0.3966547	22586.54725
0.4012556	22715.76353
0.3978475	22620.04704
0.3952286	22546.49523
0.3901832	22404.79517
0.3810902	22149.41827
0.3855014	22273.30682
0.386495	22301.21208
0.3957758	22561.86334
0.3900442	22400.89136
0.3994696	22665.60372
0.3901781	22404.65194
0.3848764	22255.75369
0.3947449	22532.91052
0.399486	22666.06431
0.3967038	22587.92622
0.3961115	22571.29148
0.3897175	22391.71599
0.3983788	22634.9686
0.3963735	22578.64975
0.4011797	22713.63187
0.4001371	22684.35045
0.3986625	22642.93631
0.4006564	22698.93499
0.3903083	22408.30861
0.3930891	22486.40737
0.3888542	22367.47021
0.3945726	22528.07147

Karena nilai output yang dihasilkan semakin jauh dari target, maka dilakukan pelatihan lagi dengan menambahkan satu lapisan hidden layer lagi pada jumlah neuron hidden serta memperkecil nilai momentumnya. Hasil dari pelatihan tersebut adalah:

```
> net4 <- newff(n.neurons=c(1,3,1), learning.rate.global=0.1,
+ momentum.global=0.4, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid", method="ADAPTgdwm")
```

net4	transformasi net4
0.3949129	22537.6288
0.3674491	21766.30797
0.3692639	21817.27663
0.3986576	22642.7987
0.4426781	23879.11444
0.4359229	23689.39465
0.4049279	22818.90007
0.395658	22558.55493
0.4442069	23922.05079
0.4302375	23529.72019
0.4154526	23114.48627
0.4101467	22965.47007
0.4023791	22747.31702
0.3949129	22537.6288
0.3854045	22270.58538
0.3975036	22610.38861
0.4092872	22941.33101
0.3963388	22577.6752
0.3922399	22462.55759
0.4025884	22753.19521
0.3994401	22664.77521
0.4094524	22945.97065
0.4020351	22737.65578
0.3963388	22577.6752
0.3853874	22270.10513
0.3657933	21719.80483
0.3752698	21985.95233
0.3774125	22046.13006
0.3975286	22611.09073
0.3850863	22261.64874
0.4055651	22836.79583
0.3853764	22269.79619
0.3739235	21948.1415
0.3952875	22548.14944
0.4056009	22837.80128
0.3995469	22667.77469
0.3982587	22631.59559
0.3843787	22241.77579
0.4031911	22770.12204
0.3988285	22647.59842
0.4092872	22941.33101
0.4070179	22877.59772
0.4038085	22787.46172

0.4081484	22909.34781
0.3856584	22277.71616
0.3916903	22447.12208
0.3825101	22189.29616
0.3949129	22537.6288

Pelatihan keempat ini masih belum mendekati nilai targetnya, maka dilakukan pelatihan lagi dengan memperbesar nilai momentumnya menjadi 0.5. hasil dari pelatihan kelima ini adalah:

```
> net5 <- newff(n.neurons=c(1,3,1), learning.rate.global=0.1,
+ momentum.global=0.5, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid", method="ADAPTgdwm")
```

net5	**transformasi net5**
0.3954235	22551.969
0.3687336	21802.38316
0.3704734	21851.24544
0.3991196	22655.77397
0.4434445	23900.63878
0.436548	23706.95058
0.4053371	22830.39245
0.3961579	22572.59462
0.4450095	23944.59181
0.430768	23544.61928
0.4158497	23125.63882
0.4105382	22976.46535
0.4028055	22759.29247
0.3954235	22551.969
0.3860977	22290.0539
0.3979792	22623.74583
0.40968	22952.3628
0.3968294	22591.4537
0.3927931	22478.09421
0.4030132	22765.12572
0.3998936	22677.51176
0.409845	22956.99683
0.4024643	22749.70987
0.3968294	22591.4537
0.3860811	22289.58769
0.3671494	21757.8909

0.3762559	22013.64695
0.3783279	22071.83907
0.3980039	22624.43953
0.3857872	22281.33351
0.4059709	22848.19273
0.3860703	22289.28438
0.3749563	21977.14769
0.3957927	22562.33798
0.4060065	22849.19255
0.3999994	22680.48315
0.3987253	22644.70005
0.3850969	22261.94644
0.4036115	22781.92898
0.3992886	22660.52033
0.40968	22952.3628
0.4074172	22888.81206
0.4042246	22799.14789
0.4085439	22920.45543
0.3863457	22297.01898
0.3922532	22462.93112
0.3832764	22210.81769
0.3954235	22551.969

Karena untuk pelatihan kelima ini hasilnya mulai naik atau mendekati nilai target, maka proses pelatihan dihentikan dan diteruskan dengan proses pengujian. Dalam proses pengujian ini yang digunakan untuk pengujian adalah menggunakan hasil dari proses pelatihan yang mendekati nilai target yaitu dengan menggunakan pelatihan yang kelima atau *net*5. Proses pengujian ini dilakukan terhadap data ke-55 yang merupakan data bulan Juli 2008.

4.3 Pengujian Data

Pada proses pengujian dilakukan setelah proses pelatihan selesai yang fungsinya adalah untuk menguji apakah jaringan menghasilkan output sesuai

dengan yang diinginkan pada waktu input yang belum pernah dipelajari oleh jaringan dimasukkan. (Puspitaningrum, 2006)

Pada proses pengujian ini dilakukan misalnya pada data ke-55 yaitu data bulan Juli 2008 dengan target bulan Juli 2009 yaitu sebesar 20272.

```
> input[,1]<-c(0.438544)
```

Dalam proses pengujian ini, langsung kita lakukan proses pengujian tanpa dilakukan proses *train* lagi, karena pada proses pengujian ini sudah menggunakan fungsi perintah untuk jaringan saraf tiruan pada proses pelatihan yang dianggap sudah mendekati target.

Hasil pengujian terhadap data input tersebut adalah sebesar 0.3712812, dan apabila ditransformasikan kembali ke data yang sebenarnya adalah sebesar 21883.64. Hasil pengujian ini mendekati nilai target yang dikehendaki yaitu sebesar 20272, sehingga dapat dilakukan ke proses selanjutnya yaitu prediksi untuk bulan selanjutnya.

Proses pengujian ini hanya dilakukan satu kali, tidak seperti pada proses pelatihan yang dilakukan berulang kali. Hal ini karena pada proses pengujian tujuannya adalah hanya untuk menguji data yang sebelumnya sudah dilakukan pelatihan. Proses selanjutnya setelah proses pelatihan dan pengujian selesai adalah memprediksi data untuk priode selanjutnya.

4.4 Prediksi Data

Data yang digunakan diatas adalah data yang berdasarkan pada bulan, oleh karena itu untuk memprediksi digunakan data bulan berikutya satu tahun

sebelumnya. Hasil dari prediksi jumlah penumpang untuk bulan Agustus 2009 adalah:

```
> input<-matrix(0,1,1)
> input[,1]<-c(0.51346)
> y<-sim(net5,input)
> y
        [,1]
[1,] 0.3706909
```

Data tersebut merupakan data transformasi dari data jumlah penumpang, untuk hasil yang sebenarnya dikembalikan ke data aslinya, yaitu:

$$x' = \frac{0.8(x-a)}{b-a} + 0.1$$

$$0.3706909 = \frac{0.8(x-14255)}{36723-14255} + 0.1$$

$$0.3706909 = \frac{0.8(x-14255)}{22468} + 0.1$$

$$x = 21857$$

Jadi prediksi untuk jumlah penumpang bus di terminal bus Kabupaten Kebumen adalah sebesar 21857.

Hasil diatas merupakan hasil prediksi menggunakan Jaringan Saraf Tiruan dengan menggunakan bantuan software R. Hasil tersebut akan dibandingkan dengan analisis runtun waktu dengan bantuan software E-Views untuk melihat bahwa prediksi menggunakan Jaringan Saraf Tiruan tersebut dapat digunakan untuk peramalan.

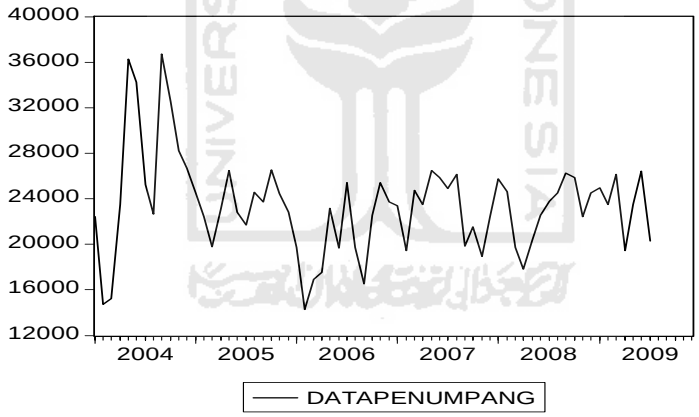
4.5 Analisis Runtun Waktu

Langkah-langkah pada analisis runtun waktu dengan model ARIMA adalah :

1. Plotting data
2. Stasioneritas
3. Estimasi model
4. Uji asumsi residual/ *diagnostic checking*
5. Pemilihan model terbaik
6. Peramalan

4.6 Plotting Data

Dari data diatas, kita olah dengan menggunakan dua software statistika yaitu : Eviews. Plot dari data di atas adalah sebagai berikut :

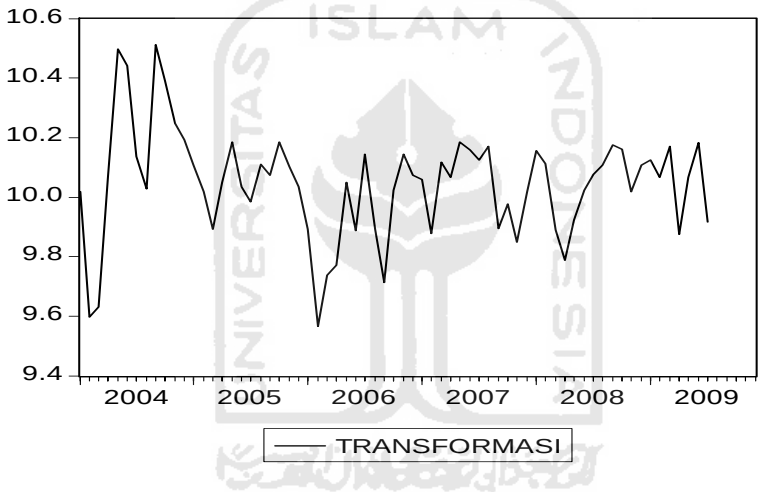


Gambar 4.3 Plot data jumlah penumpang Kota Kebumen

Plot data diatas merupakan plot data jumlah penumpang dengan sumbu x adalah menunjukkan bulan yang dimulai dari bulan Januari 2004 sampai Juli 2009 dan sumbu y menunjukkan jumlah penumpang. Plot diatas tidak stasioner dalam mean dan variansi, sehingga dalam analisis runtun waktu, kasus seperti ini dapat diatasi dengan melakukan *differencing* untuk menstasionerkan mean dan transformasi untuk menstasionerkan variansi. Dalam prakteknya, transformasi akan dilakukan terlebih dahulu, baru kemudian dilakukan *differencing*.

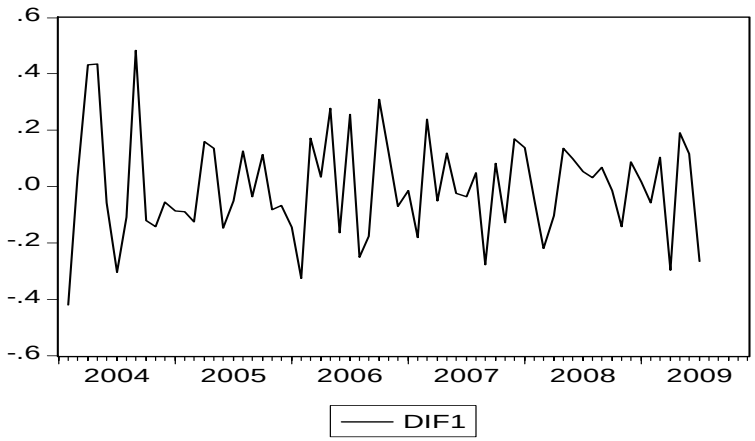
4.7 Transformasi dan Differencing

Dari plot data pada gambar 4.1 terlihat bahwa data tidak stasioner karena memiliki bentuk yang tidak teratur, karena terdapat nilai yang lebih tinggi ataupun yang lebih rendah. Hal ini dapat ditunjukkan pada bulan Februari sampai Juli 2004 yang nilainya sangat jauh dari jumlah penumpang bulan – bulan lainnya. Karena data belum stasioner, maka data perlu *differencing*. Sebelumnya akan dilakukan transformasi untuk menstasionerkan variansi. Transformasi yang digunakan adalah dengan menggunakan transformasi logaritmik.



Gambar 4.4 Hasil plot transformasi data jumlah penumpang

Plot diatas menunjukkan hasil transformasi dari data, dengan sumbu x menunjukkan bulan yaitu dari bulan Januari 2004 sampai Juli 2009, sedangkan sumbu y menunjukkan nilai variansi dari hasil transformasi. Setelah dilakukan transformasi dan data sudah stasioner dalam variansi, langkah selanjutnya adalah menstasionerkan data dalam mean, sehingga perlu dilakukan *differencing* orde satu. Hasil plot data setelah di *differencing* adalah :



Gambar 4.5 Plot *differencing* orde satu dari data jumlah penumpang

Plot diatas menunjukkan hasil transformasi dari data, dengan sumbu x menunjukkan bulan yaitu dari bulan Januari 2004 sampai Juli 2009, sedangkan sumbu y menunjukkan nilai mean dari hasil *differencing*. Dari grafik di atas, sudah dapat menunjukkan bahwa data sudah stasioner dalam hal mean.

4.8 Identifikasi Model ARIMA

Dari hasil di atas yang sudah stasioener dalam varians dan mean, kemudian kita buat correlogramnya untuk mengetahui model awal dari ARIMA yang akan kita gunakan untuk forecasting. Hasil dari correlogramnya adalah :

Tabel 4.4 Plot *ACF* dan *PACF* dari data jumlah penumpang Kota Kebumen

Correlogram of D1

Date: 01/13/10 Time: 02:14
Sample: 2004:01 2009:12
Included observations: 66

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	-0.081	-0.081	0.4482	0.503
		2	-0.273	-0.282	5.6886	0.058
		3	-0.232	-0.310	9.5288	0.023
		4	0.006	-0.180	9.5311	0.049
		5	0.200	0.008	12.465	0.029
		6	-0.067	-0.179	12.802	0.046
		7	0.019	0.005	12.828	0.076
		8	0.045	0.062	12.983	0.112
		9	-0.141	-0.168	14.556	0.104
		10	-0.099	-0.174	15.349	0.120
		11	-0.001	-0.123	15.349	0.167
		12	0.188	-0.018	18.280	0.107
		13	0.047	-0.068	18.470	0.140
		14	-0.044	0.026	18.636	0.179
		15	0.017	0.111	18.662	0.229

Dari hasil diatas, maka dapat kita tulis model awal untuk ARIMA adalah ARIMA (2,1,1). Hasil dari model awal tersebut kemudian kita lakukan overfitting

untuk mengetahui model mana yang cocok digunakan untuk melakukan forecasting.

Tabel 4.5 Tabel *Overfitting* model ARIMA

Model	Overfitting
1	ARIMA (2,1,1)
2	ARIMA (2,1,0)
3	ARIMA (1,1,1)
4	ARIMA (0,1,1)
5	ARIMA (1,1,0)

4.9 Estimasi Model

Estimasi parameter setiap model over fitting di atas untuk mencari model yang layak untuk digunakan.

Asumsi-asumsi yang diperlukan dalam analisis runtun waktu yaitu :

1. Residual berdistribusi normal
2. Tidak ada autokorelasi dalam residual
3. Model bersifat homoskedastik (variansi residual konstan)

Untuk asumsi kedua dan ketiga diharuskan untuk dipenuhi. Sedangkan untuk asumsi yang pertama bisa ditoleransi jika memang tidak dipenuhi.

4.9.1 ARIMA (2,1,1) d(log(data)) C AR(1) AR(2) MA(1)

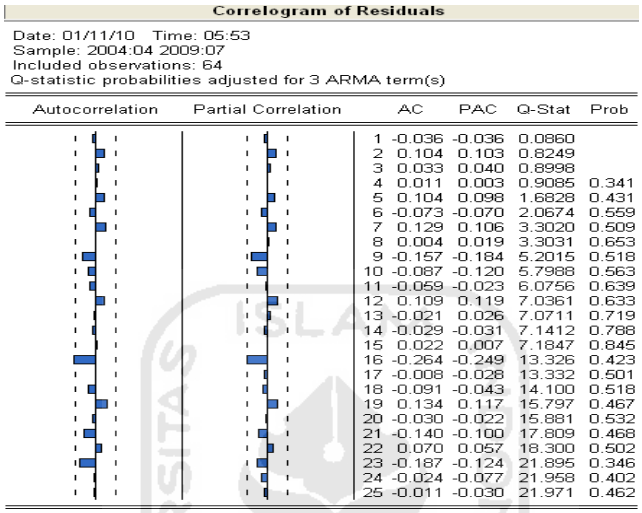
Tabel 4.6 Estimasi model ARIMA (2,1,1)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	-0.003036	0.003346	-0.907140	0.3680
AR(1)	0.488951	0.132779	3.682432	0.0005
AR(2)	-0.229535	0.119834	-1.915441	0.0602
MA(1)	-0.886408	0.075861	-11.68461	0.0000

Dari hasil output diatas, dapat diambil kesimpulan bahwa dengan menggunakan nilai $\alpha = 5\%$, maka untuk nilai constan dengan probabilitas sebesar 0.3680 tidak signifikan sehingga harus dilakukan pengujian ulang tanpa memasukkan nilai constan. Kemudian untuk parameter AR(1) dan MA(1) sudah signifikan, terlihat dengan nilai probabilitasnya sebesar 0.0005 dan 0.0000.

Sedangkan untuk parameter AR(2) sebesar 0.0602 tidak signifikan, karena nilai probabilitasnya lebih dari nilai α = 5%.

Tabel 4.7 Plot *ACF* dan *PACF* dari model ARIMA (2,1,1) untuk estimasi *autokorelasi*



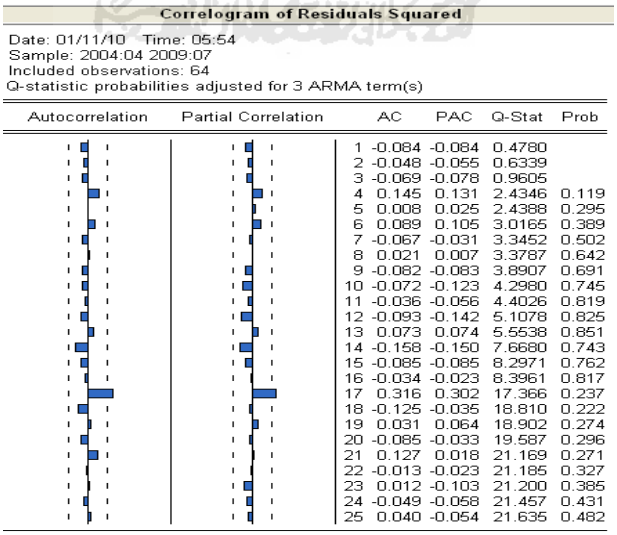
Correlogram of Residuals

Date: 01/11/10 Time: 05:53
Sample: 2004:04 2009:07
Included observations: 64
Q-statistic probabilities adjusted for 3 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	-0.036	-0.036	0.0860	
		2	0.104	0.103	0.8249	
		3	0.033	0.040	0.8998	
		4	0.011	0.003	0.9085	0.341
		5	0.104	0.098	1.6828	0.431
		6	-0.073	-0.070	2.0674	0.559
		7	0.129	0.106	3.3020	0.509
		8	0.004	0.019	3.3031	0.653
		9	-0.157	-0.184	5.2015	0.518
		10	-0.087	-0.120	5.7988	0.563
		11	-0.059	-0.023	6.0756	0.639
		12	0.109	0.119	7.0361	0.633
		13	-0.021	0.026	7.0711	0.719
		14	-0.029	-0.031	7.1412	0.788
		15	0.022	0.007	7.1847	0.845
		16	-0.264	-0.249	13.326	0.423
		17	-0.008	-0.028	13.332	0.501
		18	-0.091	-0.043	14.100	0.518
		19	0.134	0.117	15.797	0.467
		20	-0.030	-0.022	15.881	0.532
		21	-0.140	-0.100	17.809	0.468
		22	0.070	0.057	18.300	0.502
		23	-0.187	-0.124	21.895	0.346
		24	-0.024	-0.077	21.958	0.402
		25	-0.011	-0.030	21.971	0.462

Dari correlogram diatas terlihat bahwa ada indikasi autokorelasi, karena terdapat lag yang keluar dari batas kendali yang ada.

Tabel 4.8 Plot *ACF* dan *PACF* dari model ARIMA (2,1,1) untuk estimasi *heteroskedastisitas*

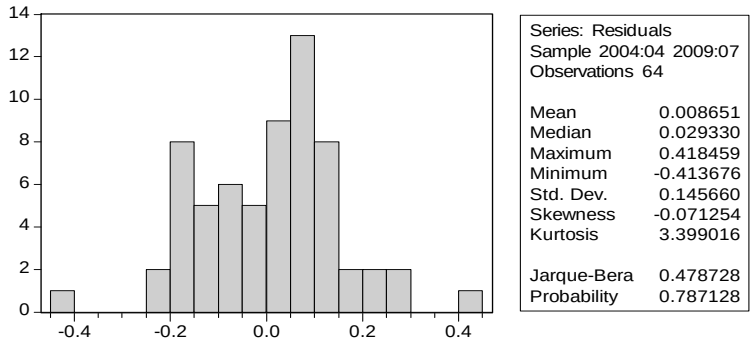


Correlogram of Residuals Squared

Date: 01/11/10 Time: 05:54
Sample: 2004:04 2009:07
Included observations: 64
Q-statistic probabilities adjusted for 3 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	-0.084	-0.084	0.4780	
		2	-0.048	-0.055	0.6339	
		3	-0.069	-0.078	0.9605	
		4	0.145	0.131	2.4346	0.119
		5	0.008	0.025	2.4388	0.295
		6	0.089	0.105	3.0165	0.389
		7	-0.067	-0.031	3.3452	0.502
		8	0.021	0.007	3.3787	0.642
		9	-0.082	-0.083	3.8907	0.691
		10	-0.072	-0.123	4.2980	0.745
		11	-0.036	-0.056	4.4026	0.819
		12	-0.093	-0.142	5.1078	0.825
		13	0.073	0.074	5.5538	0.851
		14	-0.158	-0.150	7.6680	0.743
		15	-0.085	-0.085	8.2971	0.762
		16	-0.034	-0.023	8.3961	0.817
		17	0.316	0.302	17.366	0.237
		18	-0.125	-0.035	18.810	0.222
		19	0.031	0.064	18.902	0.274
		20	-0.085	-0.033	19.587	0.296
		21	0.127	0.018	21.169	0.271
		22	-0.013	-0.023	21.185	0.327
		23	0.012	-0.103	21.200	0.385
		24	-0.049	-0.058	21.457	0.431
		25	0.040	-0.054	21.635	0.482

Dari correlogram diatas dapat kita simpulkan bahwa ada indikasi heteroskedastisitas, ini dapat ditunjukkan dengan adanya lag yang keluar.



Gambar 4.6 Histogram residual model ARIMA (4,2,1)

Hipotesis

H_0 : residual normal

H_1 : residual tidak normal

Tingkat signifikansi $\alpha = 0.05$

Daerah kritis, tolak H_0 jika p-value < α

Keputusan

Karena nilai p-value = 0,7871 > $\alpha = 0.05$, terima H_0

Kesimpulan

Karena kita menerima H_0, artinya residual berdistribusi normal.

4.9.2 ARIMA (2,1,1) tanpa c d(log(data)) AR(1) AR(2) MA(1)

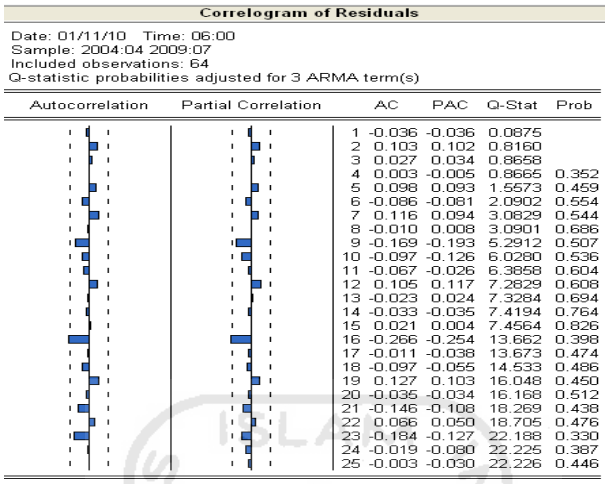
Tabel 4.9 Estimasi model ARIMA (2,1,1)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
AR(1)	0.481768	0.133757	3.601814	0.0006
AR(2)	-0.235529	0.119828	-1.965554	0.0539
MA(1)	-0.866747	0.082879	-10.45801	0.0000

Dari hasil output diatas, setelah nilai constan tidak dimasukkan dalam analisis diperoleh hasil bahwa dengan $\alpha = 5\%$, parameter AR(1) dan MA(1) sudah signifikan, terlihat dengan nilai probabilitasnya sebesar 0.0006 dan 0.0000. Sedangkan untuk parameter AR(2) sebesar 0.0539 tidak signifikan.

Uji residual :

Tabel 4.10 Plot *ACF* dan *PACF* dari model ARIMA (2,1,1) untuk estimasi *autokorelasi*



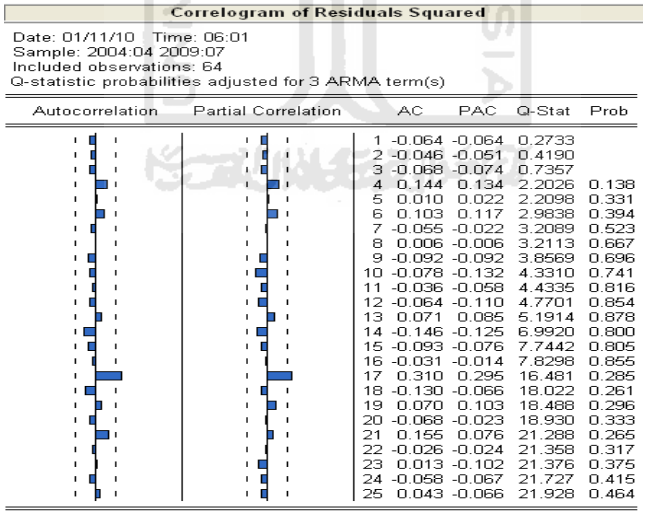
Correlogram of Residuals

Date: 01/11/10 Time: 06:00
Sample: 2004:04 2009:07
Included observations: 64
Q-statistic probabilities adjusted for 3 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	-0.036	-0.036	0.0875	
		2	0.103	0.102	0.8160	
		3	0.027	0.034	0.8658	
		4	0.003	-0.005	0.8665	0.352
		5	0.098	0.093	1.5573	0.459
		6	-0.086	-0.081	2.0902	0.554
		7	0.116	0.094	3.0829	0.544
		8	-0.010	0.008	3.0901	0.686
		9	-0.169	-0.193	5.2912	0.507
		10	-0.097	-0.126	6.0280	0.536
		11	-0.067	-0.026	6.3858	0.604
		12	0.105	0.117	7.2829	0.608
		13	-0.023	0.024	7.3284	0.694
		14	-0.033	-0.035	7.4194	0.764
		15	0.021	0.004	7.4564	0.826
		16	-0.266	-0.254	13.662	0.398
		17	-0.011	-0.038	13.673	0.474
		18	-0.097	-0.055	14.533	0.486
		19	0.127	0.103	16.048	0.450
		20	-0.035	-0.034	16.168	0.512
		21	-0.146	-0.108	18.269	0.438
		22	0.066	0.050	18.705	0.476
		23	-0.184	-0.127	22.188	0.330
		24	-0.019	-0.080	22.225	0.387
		25	-0.003	-0.030	22.226	0.446

Dari correlogram diatas terlihat bahwa ada indikasi autokorelasi, karena terdapat lag yang keluar dari batas kendali yang ada.

Tabel 4.11 Plot *ACF* dan *PACF* dari model ARIMA (2,1,1) untuk estimasi *heteroskedastisitas*

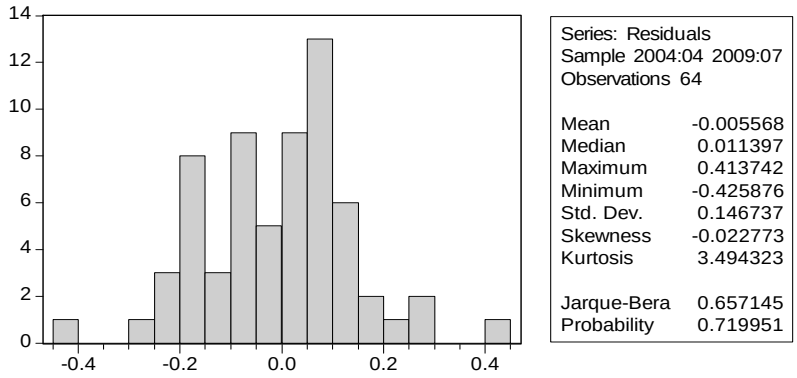


Correlogram of Residuals Squared

Date: 01/11/10 Time: 06:01
Sample: 2004:04 2009:07
Included observations: 64
Q-statistic probabilities adjusted for 3 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	-0.064	-0.064	0.2733	
		2	-0.046	-0.051	0.4190	
		3	-0.068	-0.074	0.7357	
		4	0.144	0.134	2.2026	0.138
		5	0.010	0.022	2.2098	0.331
		6	0.103	0.117	2.9838	0.394
		7	-0.055	-0.022	3.2089	0.523
		8	0.006	-0.006	3.2113	0.667
		9	-0.092	-0.092	3.8569	0.696
		10	-0.078	-0.132	4.3310	0.741
		11	-0.036	-0.058	4.4335	0.816
		12	-0.064	-0.110	4.7701	0.854
		13	0.071	0.085	5.1914	0.878
		14	-0.146	-0.125	6.9920	0.800
		15	-0.093	-0.076	7.7442	0.805
		16	-0.031	-0.014	7.8298	0.855
		17	0.310	0.295	16.481	0.285
		18	-0.130	-0.066	18.022	0.261
		19	0.070	0.103	18.488	0.296
		20	-0.068	-0.023	18.930	0.333
		21	0.155	0.076	21.288	0.265
		22	-0.026	-0.024	21.358	0.317
		23	0.013	-0.102	21.376	0.375
		24	-0.058	-0.067	21.727	0.415
		25	0.043	-0.066	21.928	0.464

Dari correlogram diatas terlihat bahwa ada indikasi heteroskedastisitas, karena terdapat lag yang keluar dari batas kendali yang ada.



Gambar 4.7 Histogram residual model ARIMA (3,2,1)

Hipotesis

H_0 : residual normal

H_1 : residual tidak normal

Tingkat signifikansi $\alpha = 0.05$

Daerah kritis, tolak H_0 jika p-value < α

Keputusan

Karena nilai p-value = 0,7199 > $\alpha = 0.05$, terima H_0

Kesimpulan

Karena kita menerima H_0, artinya residual berdistribusi normal.

4.9.3 ARIMA (2,1,0) d(log(data)) C AR(1) AR(2)

Tabel 4.12 Estimasi model ARIMA (2,1,0)

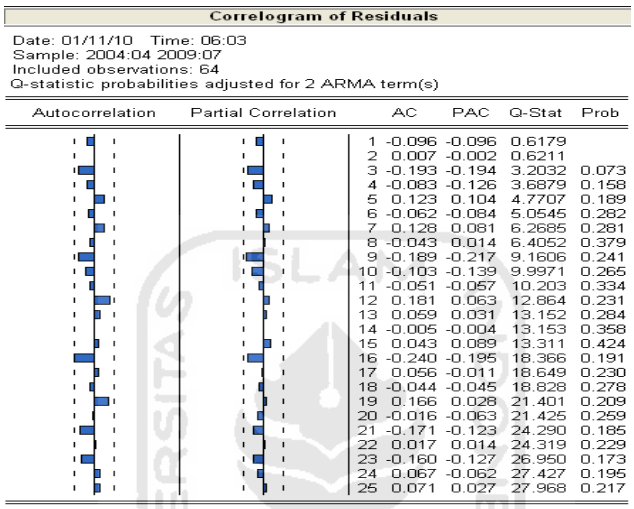
Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.004048	0.015536	0.260524	0.7953
AR(1)	-0.107557	0.124323	-0.865148	0.3903
AR(2)	-0.292004	0.119305	-2.447551	0.0173

Dari hasil output diatas, dapat diambil kesimpulan bahwa dengan menggunakan nilai $\alpha = 5\%$, maka untuk nilai constan dengan probabilitas sebesar 0.7953 tidak signifikan sehingga harus dilakukan pengujian ulang tanpa memasukkan nilai constan. Kemudian untuk parameter AR(2) dengan nilai probabilitas sebesar 0.0173 sudah signifikan sedangkan untuk parameter AR(1)

belum signifikan, terlihat dengan nilai probabilitasnya sebesar 0.3903 yang lebih dari nilai α = 5%.

Uji residual :

Tabel 4.13 Plot *ACF* dan *PACF* dari model ARIMA (2,1,0) untuk estimasi *autokorelasi*



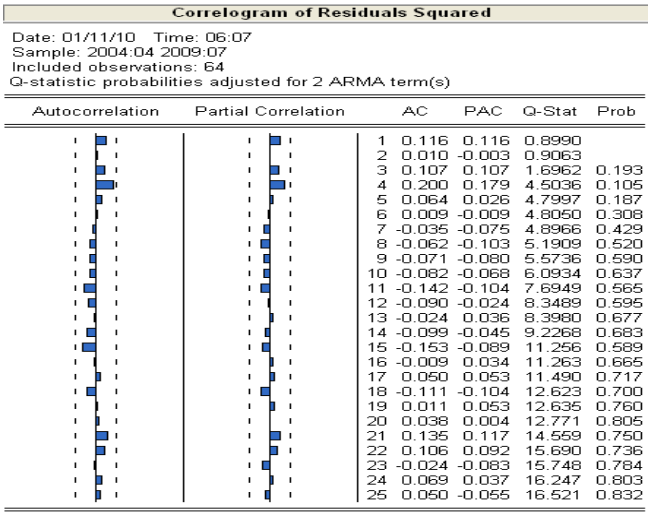
Correlogram of Residuals

Date: 01/11/10 Time: 06:03
Sample: 2004:04 2009:07
Included observations: 64
Q-statistic probabilities adjusted for 2 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	-0.096	-0.096	0.6179	
		2	0.007	-0.002	0.6211	
		3	-0.193	-0.194	3.2032	0.073
		4	-0.083	-0.126	3.6879	0.158
		5	0.123	0.104	4.7707	0.189
		6	-0.062	-0.084	5.0545	0.282
		7	0.128	0.081	6.2685	0.281
		8	-0.043	0.014	6.4052	0.379
		9	-0.189	-0.217	9.1606	0.241
		10	-0.103	-0.139	9.9971	0.265
		11	-0.051	-0.057	10.203	0.334
		12	0.181	0.063	12.864	0.231
		13	0.059	0.031	13.152	0.284
		14	-0.005	-0.004	13.153	0.358
		15	0.043	0.089	13.311	0.424
		16	-0.240	-0.195	18.366	0.191
		17	0.056	-0.011	18.649	0.230
		18	-0.044	-0.045	18.828	0.278
		19	0.166	0.028	21.401	0.209
		20	-0.016	-0.063	21.425	0.259
		21	-0.171	-0.123	24.290	0.185
		22	0.017	0.014	24.319	0.229
		23	-0.160	-0.127	26.950	0.173
		24	0.067	-0.062	27.427	0.195
		25	0.071	0.027	27.968	0.217

Dari correlogram diatas terlihat bahwa ada indikasi autokorelasi, karena terdapat lag yang keluar dari batas kendali yang ada.

Tabel 4.14 Plot *ACF* dan *PACF* dari model ARIMA (2,1,0) untuk estimasi *heteroskedastisitas*

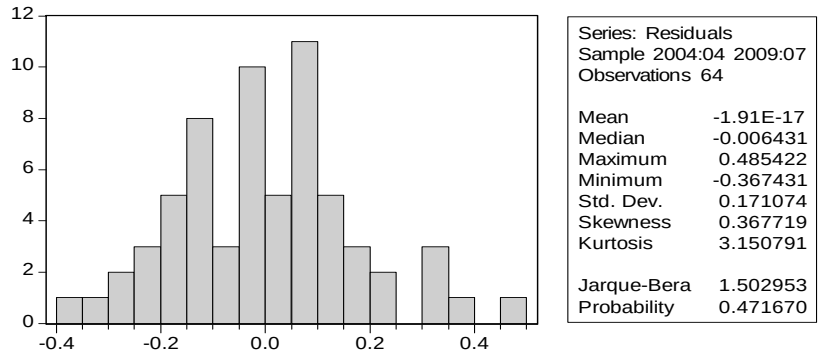


Correlogram of Residuals Squared

Date: 01/11/10 Time: 06:07
Sample: 2004:04 2009:07
Included observations: 64
Q-statistic probabilities adjusted for 2 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.116	0.116	0.8990	
		2	0.010	-0.003	0.9063	
		3	0.107	0.107	1.6962	0.193
		4	0.200	0.179	4.5036	0.105
		5	0.064	0.026	4.7997	0.187
		6	0.009	-0.009	4.8050	0.308
		7	-0.035	-0.075	4.8966	0.429
		8	-0.062	-0.103	5.1909	0.520
		9	-0.071	-0.080	5.5736	0.590
		10	-0.082	-0.068	6.0934	0.637
		11	-0.142	-0.104	7.6949	0.565
		12	-0.090	-0.024	8.3489	0.595
		13	-0.024	0.036	8.3980	0.677
		14	-0.099	-0.045	9.2268	0.683
		15	-0.153	-0.089	11.256	0.589
		16	-0.009	0.034	11.263	0.665
		17	0.050	0.053	11.490	0.717
		18	-0.111	-0.104	12.623	0.700
		19	0.011	0.053	12.635	0.760
		20	0.038	0.004	12.771	0.805
		21	0.135	0.117	14.559	0.750
		22	0.106	0.092	15.690	0.736
		23	-0.024	-0.083	15.748	0.784
		24	0.069	0.037	16.247	0.803
		25	0.050	-0.055	16.521	0.832

Dari correlogram diatas terlihat bahwa ada indikasi homoskedastisitas, karena tidak terdapat lag yang keluar dari batas kendali yang ada.



Gambar 4.8 Histogram residual model ARIMA (2,2,1)

Hipotesis

H_0 : residual normal

H_1 : residual tidak normal

Tingkat signifikansi $\alpha = 0.05$

Daerah kritis, tolak H_0 jika p-value < α

Keputusan

Karena nilai p-value = 0,4716 > $\alpha = 0.05$, terima H_0

Kesimpulan

Karena kita menerima H_0, artinya residual berdistribusi normal.

4.9.4 ARIMA (2,1,0) tanpa c d(log(data)) AR(1) MA(1)

Tabel 4.15 Estimasi model ARIMA (2,1,0)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
AR(1)	-0.105858	0.123214	-0.859140	0.3936
AR(2)	-0.291759	0.118401	-2.464164	0.0165

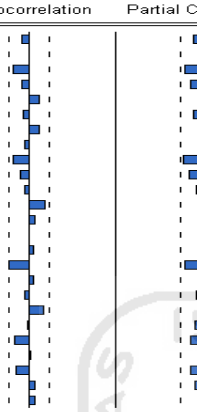
Dari hasil output diatas, setelah nilai constan tidak dimasukkan dalam analisis diperoleh hasil bahwa dengan $\alpha = 5\%$, parameter AR(1) juga masih tidak signifikan yaitu sebesar 0.3936 yang lebih dari nilai $\alpha = 5\%$ dan AR(2) sudah signifikan, terlihat dengan nilai probabilitasnya sebesar 0.0165.

Uji residual :

Tabel 4.16 Plot *ACF* dan *PACF* dari model ARIMA (2,1,0) untuk estimasi *autokorelasi*

Correlogram of Residuals

Date: 01/11/10 Time: 06:11
Sample: 2004:04 2009:07
Included observations: 64
Q-statistic probabilities adjusted for 2 ARMA term(s)



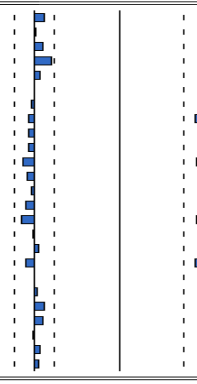
Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	-0.098	-0.098	0.6398	
		2	0.007	-0.002	0.6435	
		3	-0.193	-0.194	3.2112	0.073
		4	-0.083	-0.126	3.6943	0.158
		5	0.123	0.104	4.7796	0.189
		6	-0.063	-0.084	5.0687	0.280
		7	0.128	0.080	6.2850	0.279
		8	-0.042	0.015	6.4208	0.378
		9	-0.189	-0.216	9.1668	0.241
		10	-0.103	-0.139	9.9954	0.265
		11	-0.051	-0.057	10.203	0.334
		12	0.181	0.063	12.860	0.232
		13	0.058	0.032	13.143	0.284
		14	-0.005	-0.004	13.145	0.359
		15	0.043	0.090	13.307	0.424
		16	-0.240	-0.194	18.361	0.191
		17	0.057	-0.011	18.649	0.230
		18	-0.045	-0.045	18.831	0.278
		19	0.166	0.029	21.403	0.209
		20	-0.016	-0.063	21.427	0.258
		21	-0.171	-0.123	24.289	0.185
		22	0.018	0.014	24.321	0.229
		23	-0.160	-0.127	26.951	0.172
		24	0.067	-0.062	27.428	0.195
		25	0.070	0.027	27.962	0.217

Dari correlogram diatas terlihat bahwa ada indikasi no autokorelasi, karena tidak terdapat lag yang keluar dari batas kendali yang ada.

Tabel 4.17 Plot *ACF* dan *PACF* dari model ARIMA (2,1,0) untuk estimasi *heteroskedastisitas*

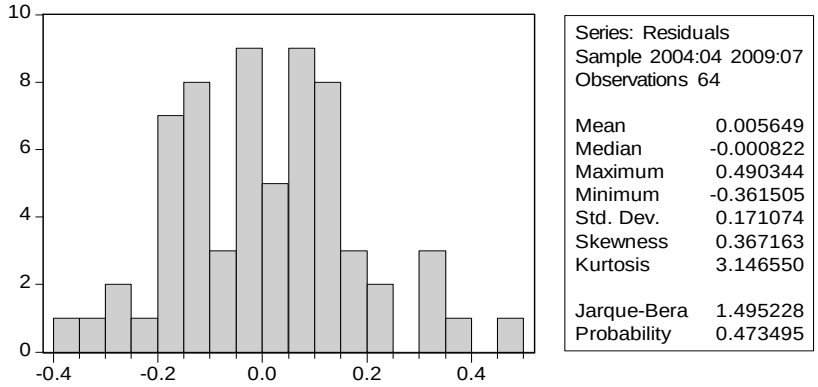
Correlogram of Residuals Squared

Date: 01/11/10 Time: 06:13
Sample: 2004:04 2009:07
Included observations: 64
Q-statistic probabilities adjusted for 2 ARMA term(s)



Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.119	0.119	0.9497	
		2	0.012	-0.003	0.9589	
		3	0.102	0.102	1.6763	0.195
		4	0.208	0.188	4.7170	0.095
		5	0.070	0.028	5.0654	0.167
		6	-0.003	-0.022	5.0661	0.281
		7	-0.039	-0.076	5.1801	0.394
		8	-0.062	-0.105	5.4659	0.486
		9	-0.068	-0.077	5.8201	0.561
		10	-0.074	-0.055	6.2535	0.619
		11	-0.138	-0.098	7.7781	0.557
		12	-0.088	-0.020	8.4035	0.589
		13	-0.031	0.027	8.4829	0.669
		14	-0.102	-0.054	9.3543	0.672
		15	-0.149	-0.088	11.275	0.588
		16	-0.013	0.029	11.290	0.663
		17	0.050	0.055	11.515	0.715
		18	-0.111	-0.102	12.645	0.699
		19	-0.007	0.039	12.649	0.759
		20	0.038	0.011	12.791	0.804
		21	0.121	0.096	14.218	0.771
		22	0.110	0.103	15.441	0.751
		23	-0.024	-0.078	15.502	0.797
		24	0.075	0.045	16.101	0.811
		25	0.054	-0.045	16.412	0.837

Dari correlogram diatas terlihat bahwa ada indikasi homoskedastisitas, karena tidak terdapat lag yang keluar dari batas kendali yang ada



Gambar 4.9 Histogram residual model ARIMA (1,2,1)

Hipotesis

H_0 : residual normal

H_1 : residual tidak normal

Tingkat signifikansi $\alpha = 0.05$

Daerah kritis, tolak H_0 jika p-value < α

Keputusan

Karena nilai p-value = 0,4734 > $\alpha = 0.05$, terima H_0

Kesimpulan

Karena kita menerima H_0, artinya residual berdistribusi normal.

4.9.5 ARIMA (1,1,1) d(log(data)) C AR(1) MA(1)

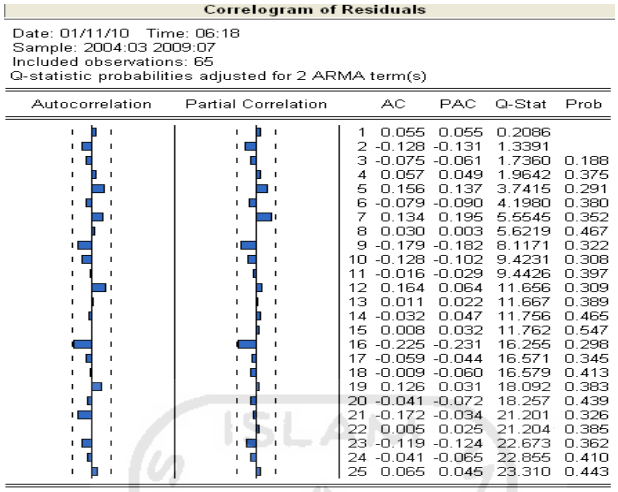
Tabel 4.18 Estimasi model ARIMA (1,1,1)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	-0.002348	0.002521	-0.931306	0.3553
AR(1)	0.479074	0.111079	4.312914	0.0001
MA(1)	-0.963005	0.040002	-24.07372	0.0000

Dari hasil output diatas, dapat diambil kesimpulan bahwa dengan menggunakan nilai $\alpha = 5\%$, maka untuk nilai constan dengan probabilitas sebesar 0.3553 tidak signifikan sehingga harus dilakukan pengujian ulang tanpa memasukkan nilai constan. Kemudian untuk parameter AR(1) dan MA(1) sudah signifikan, terlihat dengan nilai probabilitasnya sebesar 0.0001 dan 0.0000.

Uji residual :

Tabel 4.19 Plot *ACF* dan *PACF* dari model ARIMA (1,1,1) untuk estimasi *autokorelasi*



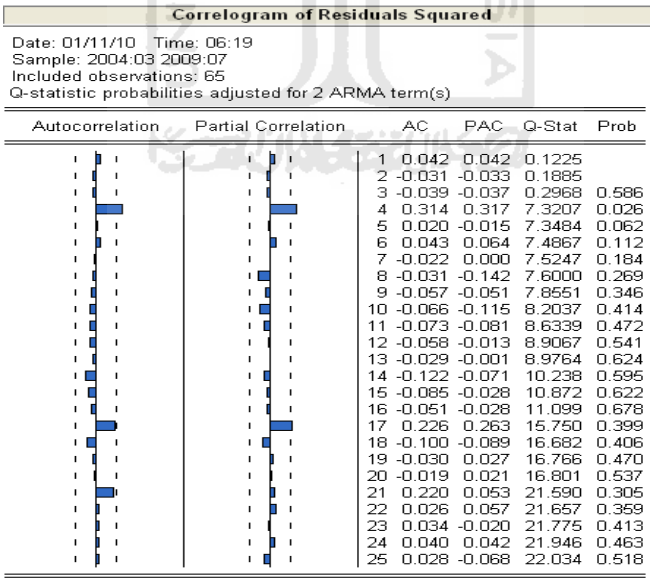
Correlogram of Residuals

Date: 01/11/10 Time: 06:18
Sample: 2004:03 2009:07
Included observations: 65
Q-statistic probabilities adjusted for 2 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.055	0.055	0.2086	
		2	-0.128	-0.131	1.3391	
		3	-0.075	-0.061	1.7360	0.188
		4	0.057	0.049	1.9642	0.375
		5	0.156	0.137	3.7415	0.291
		6	-0.079	-0.090	4.1980	0.380
		7	0.134	0.195	5.5545	0.352
		8	0.030	0.003	5.6219	0.467
		9	-0.179	-0.182	8.1171	0.322
		10	-0.128	-0.102	9.4231	0.308
		11	-0.016	-0.029	9.4426	0.397
		12	0.164	0.064	11.656	0.309
		13	0.011	0.022	11.667	0.389
		14	-0.032	0.047	11.756	0.465
		15	0.008	0.032	11.762	0.547
		16	-0.225	-0.231	16.255	0.298
		17	-0.059	-0.044	16.571	0.345
		18	-0.009	-0.060	16.579	0.413
		19	0.126	0.031	18.092	0.383
		20	-0.041	-0.072	18.257	0.439
		21	-0.172	-0.034	21.201	0.326
		22	0.005	0.025	21.204	0.386
		23	-0.119	-0.124	22.673	0.362
		24	-0.041	-0.065	22.855	0.410
		25	0.065	0.045	23.310	0.443

Dari correlogram diatas terlihat bahwa ada indikasi no autokorelasi, karena tidak terdapat lag yang keluar dari batas kendali yang ada.

Tabel 4.20 Plot *ACF* dan *PACF* dari model ARIMA (1,1,1) untuk estimasi *heteroskedastisitas*

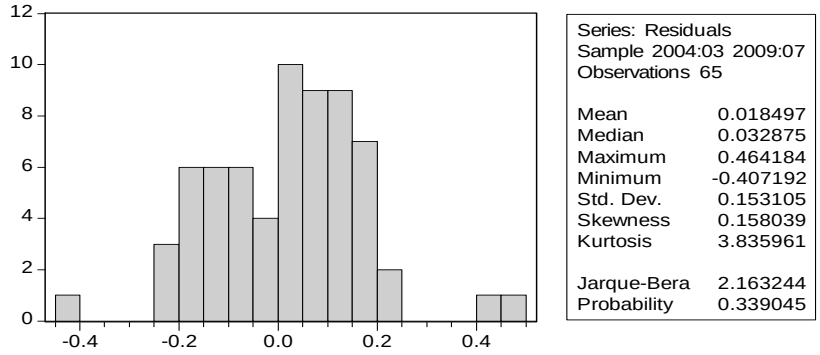


Correlogram of Residuals Squared

Date: 01/11/10 Time: 06:19
Sample: 2004:03 2009:07
Included observations: 65
Q-statistic probabilities adjusted for 2 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.042	0.042	0.1225	
		2	-0.031	-0.033	0.1885	
		3	-0.039	-0.037	0.2968	0.586
		4	0.314	0.317	7.3207	0.026
		5	0.020	-0.015	7.3484	0.062
		6	0.043	0.064	7.4867	0.112
		7	-0.022	0.000	7.5247	0.184
		8	-0.031	-0.142	7.6000	0.269
		9	-0.057	-0.051	7.8551	0.346
		10	-0.066	-0.115	8.2037	0.414
		11	-0.073	-0.081	8.6339	0.472
		12	-0.058	-0.013	8.9067	0.541
		13	-0.029	-0.001	8.9764	0.624
		14	-0.122	-0.071	10.238	0.595
		15	-0.085	-0.028	10.872	0.622
		16	-0.051	-0.028	11.099	0.678
		17	0.226	0.263	15.750	0.399
		18	-0.100	-0.089	16.682	0.406
		19	-0.030	0.027	16.766	0.470
		20	-0.019	0.021	16.801	0.537
		21	0.220	0.053	21.590	0.305
		22	0.026	0.057	21.657	0.359
		23	0.034	-0.020	21.775	0.413
		24	0.040	0.042	21.946	0.463
		25	0.028	-0.068	22.034	0.518

Dari correlogram diatas terlihat bahwa ada indikasi heteroskedastisitas, karena terdapat lag yang keluar dari batas kendali yang ada.



Gambar 4.10 Histogram residual model ARIMA (0,2,1)

Hipotesis

H_0 : residual normal

H_1 : residual tidak normal

Tingkat signifikansi $\alpha = 0.05$

Daerah kritis, tolak H_0 jika p-value < α

Keputusan

Karena nilai p-value = 0,3390 > $\alpha = 0.05$, terima H_0

Kesimpulan

Karena kita menerima H_0, artinya residual berdistribusi normal.

4.9.6 ARIMA (1,1,1) tanpa c d(log(data)) AR(1) MA(1)

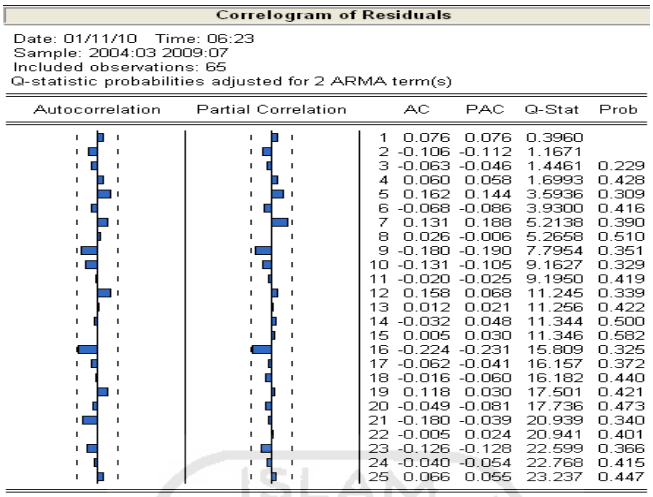
Tabel 4.21 Estimasi model ARIMA (1,1,1)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
AR(1)	0.482146	0.108927	4.426335	0.0000
MA(1)	-0.957683	0.034389	-27.84877	0.0000

Dari hasil output diatas, setelah nilai constan tidak dimasukkan dalam analisis diperoleh hasil bahwa dengan $\alpha = 5\%$, parameter AR(1) dan MA(1) sudah signifikan, terlihat dengan nilai probabilitasnya sebesar 0.0000 dan 0.0000.

Uji Residual :

Tabel 4.22 Plot *ACF* dan *PACF* dari model ARIMA (1,1,1) untuk estimasi *autokorelasi*



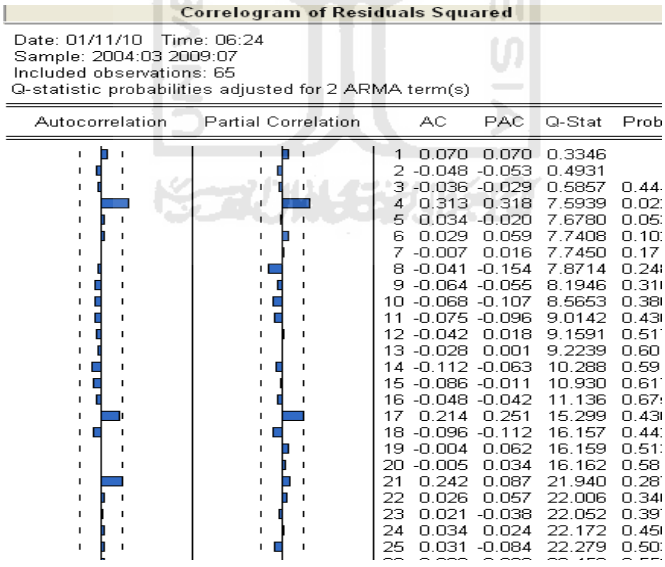
Correlogram of Residuals

Date: 01/11/10 Time: 06:23
Sample: 2004:03 2009:07
Included observations: 65
Q-statistic probabilities adjusted for 2 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.076	0.076	0.3960	
		2	-0.106	-0.112	1.1671	
		3	-0.063	-0.046	1.4461	0.229
		4	0.060	0.058	1.6993	0.428
		5	0.162	0.144	3.5936	0.309
		6	-0.068	-0.086	3.9300	0.416
		7	0.131	0.188	5.2138	0.390
		8	0.026	-0.006	5.2658	0.510
		9	-0.180	-0.190	7.7954	0.351
		10	-0.131	-0.105	9.1627	0.329
		11	-0.020	-0.025	9.1960	0.419
		12	0.158	0.068	11.245	0.339
		13	0.012	0.021	11.256	0.422
		14	-0.032	0.048	11.344	0.500
		15	0.005	0.030	11.346	0.582
		16	-0.224	-0.231	15.809	0.325
		17	-0.062	-0.041	16.157	0.372
		18	-0.016	-0.060	16.182	0.440
		19	0.118	0.030	17.501	0.421
		20	-0.049	-0.081	17.736	0.473
		21	-0.180	-0.039	20.939	0.340
		22	-0.005	0.024	20.941	0.401
		23	-0.126	-0.128	22.599	0.366
		24	-0.040	-0.054	22.768	0.415
		25	0.066	0.055	23.237	0.447

Dari correlogram diatas terlihat bahwa ada indikasi no autokorelasi, karena tidak terdapat lag yang keluar dari batas kendali yang ada.

Tabel 4.23 Plot *ACF* dan *PACF* dari model ARIMA (1,1,0) untuk estimasi *heteroskedastisitas*

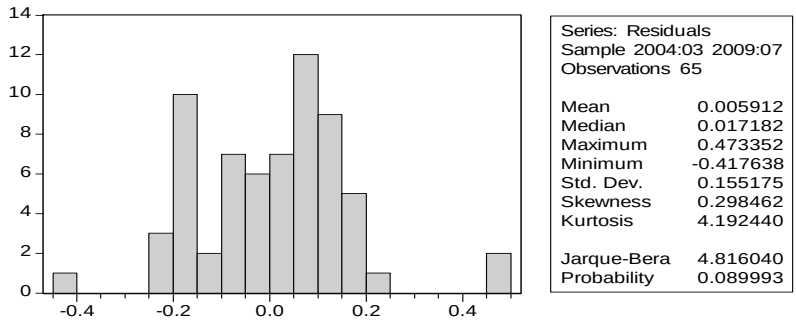


Correlogram of Residuals Squared

Date: 01/11/10 Time: 06:24
Sample: 2004:03 2009:07
Included observations: 65
Q-statistic probabilities adjusted for 2 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.070	0.070	0.3346	
		2	-0.048	-0.053	0.4931	
		3	-0.036	-0.029	0.5857	0.44
		4	0.313	0.318	7.5939	0.02
		5	0.034	-0.020	7.6780	0.05
		6	0.029	0.059	7.7408	0.10
		7	-0.007	0.016	7.7450	0.17
		8	-0.041	-0.154	7.8714	0.24
		9	-0.064	-0.055	8.1946	0.31
		10	-0.068	-0.107	8.5653	0.38
		11	-0.075	-0.096	9.0142	0.43
		12	-0.042	0.018	9.1591	0.51
		13	-0.028	0.001	9.2239	0.60
		14	-0.112	-0.063	10.288	0.59
		15	-0.086	-0.011	10.930	0.61
		16	-0.048	-0.042	11.136	0.67
		17	0.214	0.251	15.299	0.43
		18	-0.096	-0.112	16.157	0.44
		19	-0.004	0.062	16.159	0.51
		20	-0.005	0.034	16.162	0.58
		21	0.242	0.087	21.940	0.28
		22	0.026	0.057	22.006	0.34
		23	0.021	-0.038	22.052	0.39
		24	0.034	0.024	22.172	0.45
		25	0.031	-0.084	22.279	0.50

Dari correlogram diatas terlihat bahwa ada indikasi heteroskedastisitas, karena terdapat lag yang keluar dari batas kendali yang ada.



Gambar 4.11 Histogram residual model ARIMA (4,2,0)

Hipotesis

H_0 : residual normal

H_1 : residual tidak normal

Tingkat signifikansi $\alpha = 0.05$

Daerah kritis, tolak H_0 jika p-value < α

Keputusan

Karena nilai p-value = 0,0899 > $\alpha = 0.05$, terima H_0

Kesimpulan

Karena kita menerima H_0, artinya residual berdistribusi normal.

4.9.7 ARIMA (0,1,1) d(log(data)) C MA(1)

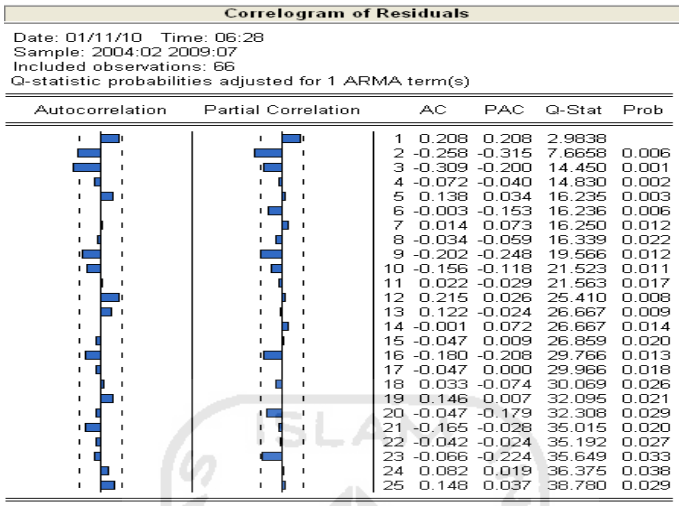
Tabel 4.24 Estimasi model ARIMA (0,1,1)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.002360	0.012043	0.195968	0.8453
MA(1)	-0.471125	0.110403	-4.267301	0.0001

Dari hasil output diatas, dapat diambil kesimpulan bahwa dengan menggunakan nilai $\alpha = 5\%$, maka untuk nilai constan dengan probabilitas sebesar 0.8453 tidak signifikan sehingga harus dilakukan pengujian ulang tanpa memasukkan nilai constan. Kemudian untuk parameter MA(1) sudah signifikan, terlihat dengan nilai probabilitasnya sebesar 0.0001 yang kurang dari nilai $\alpha = 5\%$

Uji Residual :

Tabel 4.25 Plot *ACF* dan *PACF* dari model ARIMA (0,1,1) untuk estimasi *autokorelasi*



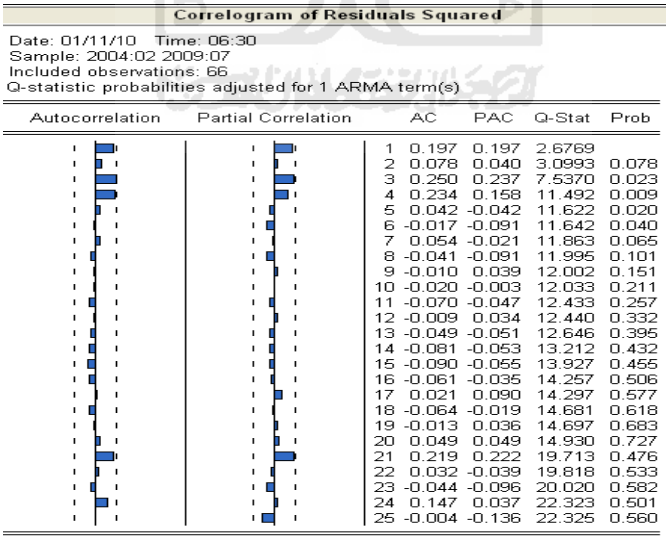
Correlogram of Residuals

Date: 01/11/10 Time: 06:28
Sample: 2004:02 2009:07
Included observations: 66
Q-statistic probabilities adjusted for 1 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.208	0.208	2.9838	
		2	-0.258	-0.315	7.6658	0.006
		3	-0.309	-0.200	14.450	0.001
		4	-0.072	-0.040	14.830	0.002
		5	0.138	0.034	16.235	0.003
		6	-0.003	-0.153	16.236	0.006
		7	0.014	0.073	16.250	0.012
		8	-0.034	-0.059	16.339	0.022
		9	-0.202	-0.248	19.566	0.012
		10	-0.156	-0.118	21.523	0.011
		11	0.022	-0.029	21.563	0.017
		12	0.215	0.026	25.410	0.008
		13	0.122	-0.024	26.667	0.009
		14	-0.001	0.072	26.667	0.014
		15	-0.047	0.009	26.859	0.020
		16	-0.180	-0.208	29.766	0.013
		17	-0.047	0.000	29.966	0.018
		18	0.033	-0.074	30.069	0.026
		19	0.146	0.007	32.095	0.021
		20	-0.047	-0.179	32.308	0.029
		21	-0.165	-0.028	35.015	0.020
		22	-0.042	-0.024	35.192	0.027
		23	-0.066	-0.224	35.649	0.033
		24	0.082	0.019	36.375	0.038
		25	0.148	0.037	38.780	0.029

Dari correlogram diatas terlihat bahwa ada indikasi autokorelasi, karena terdapat lag yang keluar dari batas kendali yang ada.

Tabel 4.26 Plot *ACF* dan *PACF* dari model ARIMA (0,1,1) untuk estimasi *heteroskedastisitas*



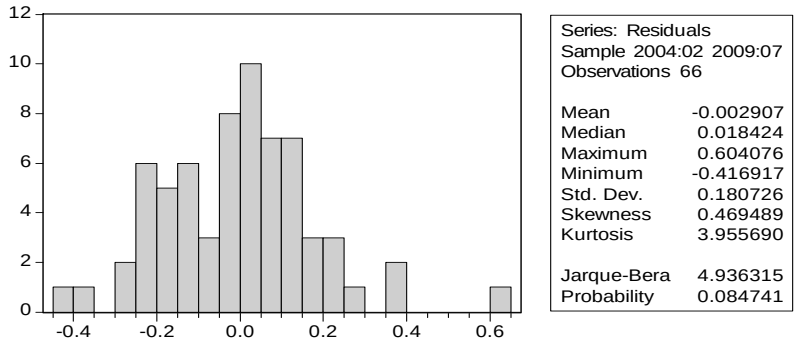
Correlogram of Residuals Squared

Date: 01/11/10 Time: 06:30
Sample: 2004:02 2009:07
Included observations: 66
Q-statistic probabilities adjusted for 1 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.197	0.197	2.6769	
		2	0.078	0.040	3.0993	0.078
		3	0.250	0.237	7.5370	0.023
		4	0.234	0.158	11.492	0.009
		5	0.042	-0.042	11.622	0.020
		6	-0.017	-0.091	11.642	0.040
		7	0.054	-0.021	11.863	0.065
		8	-0.041	-0.091	11.995	0.101
		9	-0.010	0.039	12.002	0.151
		10	-0.020	-0.003	12.033	0.211
		11	-0.070	-0.047	12.433	0.257
		12	-0.009	0.034	12.440	0.332
		13	-0.049	-0.051	12.646	0.395
		14	-0.081	-0.053	13.212	0.432
		15	-0.090	-0.055	13.927	0.455
		16	-0.061	-0.035	14.257	0.506
		17	0.021	0.090	14.297	0.577
		18	-0.064	-0.019	14.681	0.618
		19	-0.013	0.036	14.697	0.683
		20	0.049	0.049	14.930	0.727
		21	0.219	0.222	19.713	0.476
		22	0.032	-0.039	19.818	0.533
		23	-0.044	-0.096	20.020	0.582
		24	0.147	0.037	22.323	0.501
		25	-0.004	-0.136	22.325	0.560

.

Dari correlogram diatas terlihat bahwa ada indikasi heteroskedastisitas, karena terdapat lag yang keluar dari batas kendali yang ada.



Gambar 4.12 Histogram residual model ARIMA (3,2,0)

Hipotesis

H_0 : residual normal

H_1 : residual tidak normal

Tingkat signifikansi $\alpha = 0.05$

Daerah kritis, tolak H_0 jika p-value < α

Keputusan

Karena nilai p-value = 0,0847 > $\alpha = 0.05$, terima H_0

Kesimpulan

Karena kita menerima H_0, artinya residual berdistribusi normal.

4.9.8 ARIMA (0,1,1) tanpa c d(log(data)) MA(1)

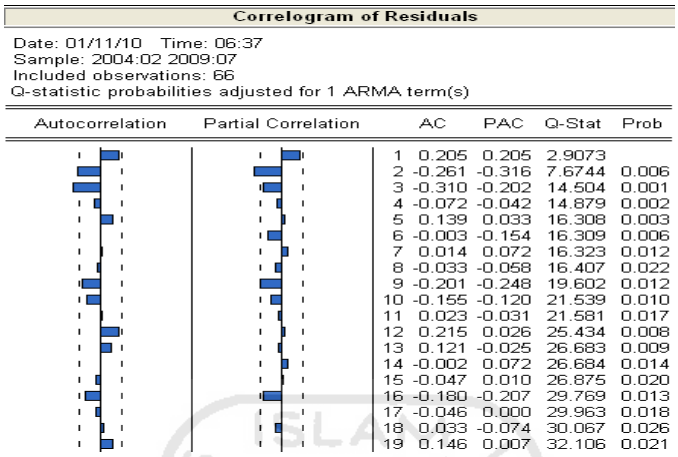
Tabel 4.27 Estimasi model ARIMA (0,1,1)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
MA(1)	-0.466934	0.109778	-4.253427	0.0001

Dari hasil output diatas, setelah nilai constan tidak dimasukkan dalam analisis diperoleh hasil bahwa dengan $\alpha = 5\%$, parameter MA(1) sudah signifikan, terlihat dengan nilai probabilitasnya sebesar 0.0001.

Uji Residual :

Tabel 4.28 Plot *ACF* dan *PACF* dari model ARIMA (0,1,1) untuk estimasi *autokorelasi*



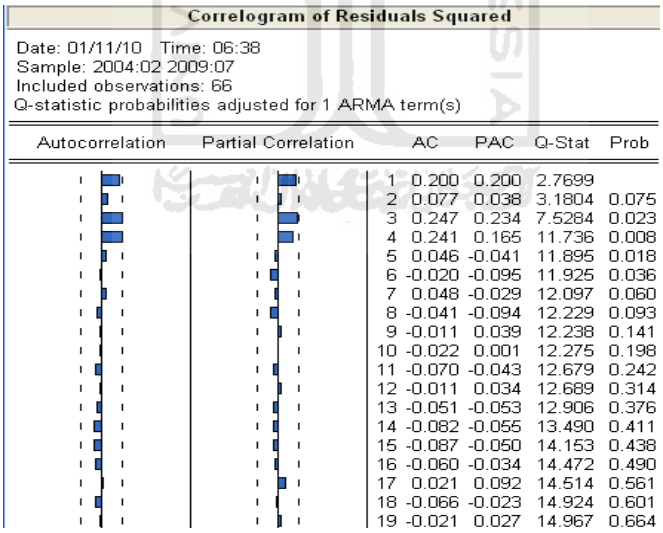
Correlogram of Residuals

Date: 01/11/10 Time: 06:37
Sample: 2004:02 2009:07
Included observations: 66
Q-statistic probabilities adjusted for 1 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.205	0.205	2.9073	
		2	-0.261	-0.316	7.6744	0.006
		3	-0.310	-0.202	14.504	0.001
		4	-0.072	-0.042	14.879	0.002
		5	0.139	0.033	16.308	0.003
		6	-0.003	-0.154	16.309	0.006
		7	0.014	0.072	16.323	0.012
		8	-0.033	-0.058	16.407	0.022
		9	-0.201	-0.248	19.602	0.012
		10	-0.155	-0.120	21.539	0.010
		11	0.023	-0.031	21.581	0.017
		12	0.215	0.026	25.434	0.008
		13	0.121	-0.025	26.683	0.009
		14	-0.002	0.072	26.684	0.014
		15	-0.047	0.010	26.875	0.020
		16	-0.180	-0.207	29.769	0.013
		17	-0.046	0.000	29.963	0.018
		18	0.033	-0.074	30.067	0.026
		19	0.146	0.007	32.106	0.021

Dari correlogram diatas terlihat bahwa ada indikasi autokorelasi, karena terdapat lag yang keluar dari batas kendali yang ada.

Tabel 4.29 Plot *ACF* dan *PACF* dari model ARIMA (0,1,1) untuk estimasi *heteroskedastisitas*

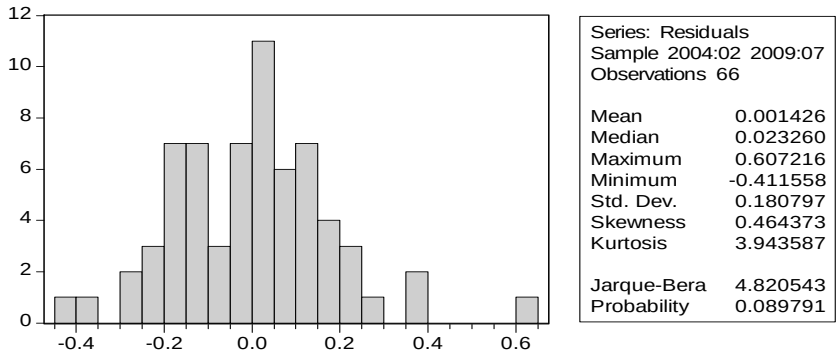


Correlogram of Residuals Squared

Date: 01/11/10 Time: 06:38
Sample: 2004:02 2009:07
Included observations: 66
Q-statistic probabilities adjusted for 1 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.200	0.200	2.7699	
		2	0.077	0.038	3.1804	0.075
		3	0.247	0.234	7.5284	0.023
		4	0.241	0.165	11.736	0.008
		5	0.046	-0.041	11.895	0.018
		6	-0.020	-0.095	11.925	0.036
		7	0.048	-0.029	12.097	0.060
		8	-0.041	-0.094	12.229	0.093
		9	-0.011	0.039	12.238	0.141
		10	-0.022	0.001	12.275	0.198
		11	-0.070	-0.043	12.679	0.242
		12	-0.011	0.034	12.689	0.314
		13	-0.051	-0.053	12.906	0.376
		14	-0.082	-0.055	13.490	0.411
		15	-0.087	-0.050	14.153	0.438
		16	-0.060	-0.034	14.472	0.490
		17	0.021	0.092	14.514	0.561
		18	-0.066	-0.023	14.924	0.601
		19	-0.021	0.027	14.967	0.664

Dari correlogram diatas terlihat bahwa ada indikasi heteroskedastisitas, karena terdapat lag yang keluar dari batas kendali yang ada.



Gambar 4.13 Histogram residual model ARIMA (2,2,0)

Hipotesis

H_0 : residual normal

H_1 : residual tidak normal

Tingkat signifikansi $\alpha = 0.05$

Daerah kritis, tolak H_0 jika p-value < α

Keputusan

Karena nilai p-value = 0,0897 > $\alpha = 0.05$, terima H_0

Kesimpulan

Karena kita menerima H_0, artinya residual berdistribusi normal.

4.9.9 ARIMA (1,1,0) d(log(data)) C AR(1)

Tabel 4.30 Estimasi model ARIMA (1,1,0)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.004735	0.020515	0.230795	0.8182
AR(1)	-0.084007	0.122324	-0.686760	0.4948

Dari hasil output diatas, dapat diambil kesimpulan bahwa dengan menggunakan nilai $\alpha = 5\%$, maka untuk nilai constan dengan probabilitas sebesar 0.8182 tidak signifikan sehingga harus dilakukan pengujian ulang tanpa memasukkan nilai constan. Kemudian untuk parameter AR(1) juga tidak signifikan, terlihat dengan nilai probabilitasnya sebesar 0.4948 yang lebih dari nilai $\alpha = 5\%$

4.9.10 ARIMA (1,1,0) tanpa c d(log(data)) AR(1)

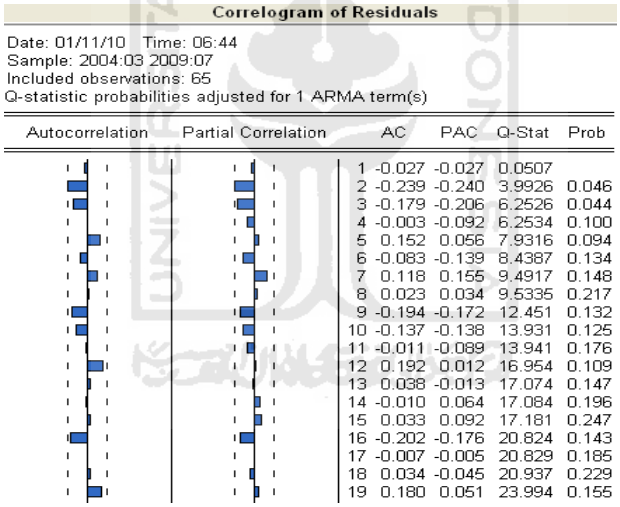
Tabel 4.31 Estimasi model ARIMA (1,1,0)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
AR(1)	-0.083615	0.121404	-0.688734	0.4935

Dari hasil output di atas, setelah nilai constan tidak dimasukkan dalam analisis diperoleh hasil bahwa dengan α = 5%, parameter AR(1) tidak signifikan, terlihat dengan nilai probabilitasnya sebesar 0.4935 yang lebih dari nilai α = 5%.

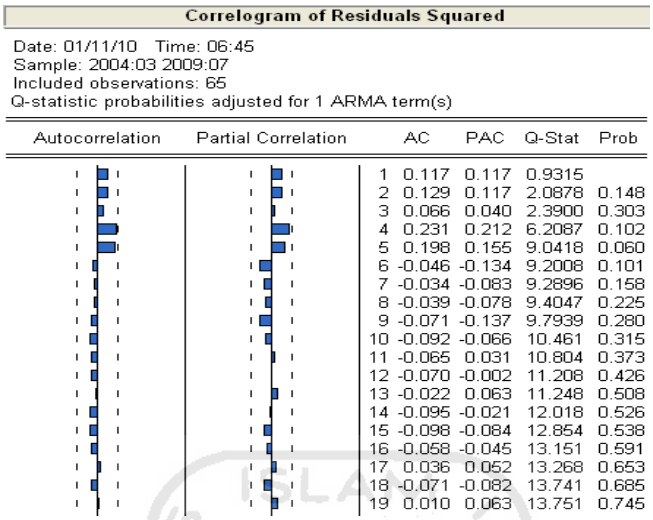
Uji Residual :

Tabel 4.32 Plot *ACF* dan *PACF* dari model ARIMA (1,1,0) untuk estimasi *autokorelasi*



Dari correlogram diatas terlihat bahwa ada indikasi autokorelasi, karena terdapat lag yang keluar dari batas kendali yang ada.

Tabel 4.33 Plot *ACF* dan *PACF* dari model ARIMA (1,1,0) untuk estimasi *hetroskedastisitas*

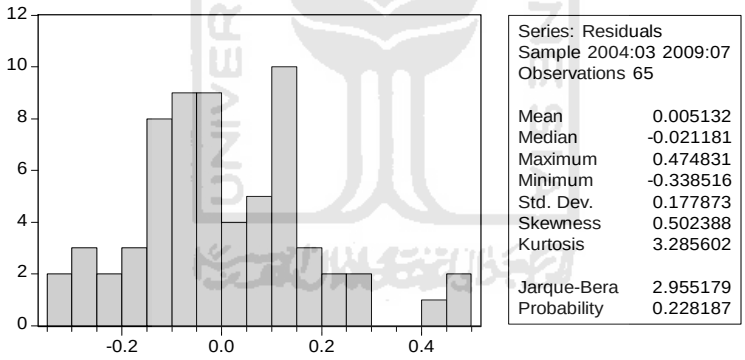


Correlogram of Residuals Squared

Date: 01/11/10 Time: 06:45
Sample: 2004:03 2009:07
Included observations: 65
Q-statistic probabilities adjusted for 1 ARMA term(s)

Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	0.117	0.117	0.9315	
		2	0.129	0.117	2.0878	0.148
		3	0.066	0.040	2.3900	0.303
		4	0.231	0.212	6.2087	0.102
		5	0.198	0.155	9.0418	0.060
		6	-0.046	-0.134	9.2008	0.101
		7	-0.034	-0.083	9.2896	0.158
		8	-0.039	-0.078	9.4047	0.225
		9	-0.071	-0.137	9.7939	0.280
		10	-0.092	-0.066	10.461	0.315
		11	-0.065	0.031	10.804	0.373
		12	-0.070	-0.002	11.208	0.426
		13	-0.022	0.063	11.248	0.508
		14	-0.095	-0.021	12.018	0.526
		15	-0.098	-0.084	12.854	0.538
		16	-0.058	-0.045	13.151	0.591
		17	0.036	0.052	13.268	0.653
		18	-0.071	-0.082	13.741	0.685
		19	0.010	0.063	13.751	0.745

Dari correlogram diatas terlihat bahwa ada indikasi heteroskedastisitas, karena terdapat lag yang keluar dari batas kendali yang ada.



Gambar 4.14 Histogram residual model ARIMA (1,2,0)

Hipotesis

H_0 : residual normal

H_1 : residual tidak normal

Tingkat signifikansi $\alpha = 0.05$

Daerah kritis, tolak H_0 jika p-value $< \alpha$

Keputusan

Karena nilai p-value = 0,2281 $> \alpha = 0.05$, terima H_0

Kesimpulan

Karena kita menerima H_0, artinya residual berdistribusi normal.

4.10 Pemilihan model terbaik

Pengujian overfitting model – model di atas :

Tabel 4.34 Tabel Pengujian Model yang Terbaik

Model	Parameter				Uji Asumsi		
	AR(1)	AR(2)	MA(1)	C	No Autokorelasi	Homoskedastisitas	Normal
ARIMA (2,1,1)	sig	Tdk	sig	tdk	Auto	Hetero	normal
ARIMA(2,1,1) tanpa c	sig	Tdk	sig	-	Auto	Hetero	Normal
ARIMA (2,1,0)	tdk	Sig	-	tdk	Auto	Homo	Normal
ARIMA (2,1,0) tanpa c	tdk	Sig	-	-	No Auto	Homo	Normal
ARIMA (1,1,1)	sig	-	sig	tdk	No auto	Hetero	Normal
ARIMA(1,1,1) tanpa c	sig	-	sig	-	No Auto	Hetero	Normal
ARIMA (0,1,1)	-	-	sig	tdk	Auto	Hetero	normal
ARIMA (0,1,1) tanpa c	-	-	sig	-	Auto	Hetero	Normal
ARIMA (1,1,0)	tdk	-	-	tdk	Auto	Homo	normal
ARIMA (1,1,0) tanpa c	tdk	-	-	-	Auto	Homo	Normal

4.11 Forecasting

Dari pengujian diatas, dapat dilihat model yang dapat digunakan untuk forecasting adalah model ARIMA (1,1,1) tanpa c. Hal ini karena semua nilai parameter dari model tersebut sudah signifikan, sehingga dapat digunakan untuk forecasting. Hasil dari forecasting untuk periode bulan Agustus 2009 adalah: 21554.05

Dari hasil pengujian dengan Jaringan Saraf Tiruan dan Analisis Runtun Waktu diperoleh hasil prediksi menggunakan JST adalah sebesar 21857 dan hasil dengan analisis runtun waktu diperoleh hasil 21554. Dari hasil pengujian menggunakan Jaringan Saraf Tiruan dan Analisis Runtun Waktu yang kemudian dibandingkan dengan data sebenarnya di lapangan sebesar 21711, maka kedua metode diatas dapat digunakan sebagai acuan untuk peramalan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pembahasan pada bab sebelumnya maka dapat disimpulkan bahwa Jaringan Saraf Tiruan juga dapat digunakan untuk memprediksi atau meramalkan data untuk periode selanjutnya. Dalam pembahasan ini, Jaringan Saraf Tiruan diterapkan untuk memprediksi jumlah penumpang bus di terminal bus Kabupaten Kebumen dengan prediksi jumlah penumpang untuk bulan Agustus 2009 adalah sebesar 21857 orang.

5.2 Saran

Dari hasil penerapan Jaringan Saraf Tiruan untuk peramalan jumlah penumpang bus tersebut, masih banyak kekurangan yang dapat menyebabkan error maupun hasil prediksinya kurang akurat. Untuk penerapan metode ini selanjutnya perlu pengamatan tentang model jaringan serta *weight* yang digunakan untuk proses kalkulasinya.

DAFTAR PUSTAKA

Andijasa dan Mistianingsih. 2010. *Penerapan Jaringan Syaraf Tiruan Untuk Memprediksi Jumlah Pengangguran di Provinsi Kalimantan Timur Dengan Menggunakan Algoritma Pembelajaran Backpropagation.* 5 hlm. http://informatikamulawarman.files.wordpress.com/.../08-jurnal-ilkom-unmul-v-5-1-0.pdf. 28 Agustus 2010. pekerjaan 19.57

http://www.id.wikipedia.org/wiki/Jaringan_saraf_tiruan

http://www.id.wikipedia.org/wiki/Kabupaten_Kebumen

Kusumadewi, Sri. 2003. *Artificial Intelligence (Teknik dan Aplikasinya)*. Graha Ilmu. Yogyakarta.

Makridakis, Spyros, Whellwright, S.C., McGee, V.E. 1999. *Metode dan Aplikasi Peramalan*. Jakarta : Binarupa Aksara.

Minsky dan Papert. 2010. *Training Hidden Units with Back Propagation.* 16 hlm. http://www.stanford.edu/group/pdplab/pdphandbook/handbookch6.html. 22 Oktober 2010. pekerjaan 06.22

Puspitaningrum, Diyah. 2006. *Pengantar Jaringan Saraf Tiruan*. ANDI OFFSET. Yogyakarta.

Siana Halim dan Adrian Michael Wibisono. *2000. Penerapan Jaringan Saraf Tiruan Untuk Peramalan.* 8 hlm. http://puslit.petra.ac.id/journals/industrial. 28 Agustus 2010. pekerjaan 20.03

Siang Jok, Jek. 2005. *Jaringan Saraf Tiruan dan Pemrogramannya Menggunakan MATLAB*. ANDI OFFSET. Yogyakarta.

LAMPIRAN

Pelatihan Pertama:

```
> library(AMORE)
> input<-matrix(0,48,1)
> input[,1]<-c(0.39090, 0.11666, 0.13489, 0.42832, 0.88333, 0.81062, 0.49120,
0.39834, 0.9, 0.75049, 0.59770, 0.54383, 0.46560, 0.39090, 0.29611, 0.41678,
0.53514, 0.40514, 0.36423, 0.46770, 0.43615, 0.53681, 0.46215, 0.40514,
0.29594, 0.1, 0.19503, 0.21643, 0.41703, 0.29294, 0.49761, 0.29583, 0.18157,
0.39464, 0.49797, 0.43722, 0.42433, 0.28589, 0.47375, 0.43003, 0.53514,
0.51224, 0.47995, 0.52364, 0.29864, 0.35875, 0.26727, 0.39090)
> target<-matrix(0,48,1)
> target[,1]<-c(0.46560, 0.39090, 0.29611, 0.41678, 0.53514, 0.40514, 0.36423,
0.46770, 0.43615, 0.53681, 0.46215, 0.40514, 0.29594, 0.1, 0.19503, 0.21643,
0.41703, 0.29294, 0.49761, 0.29583, 0.18157, 0.39464, 0.49797, 0.43722,
0.42433, 0.28589, 0.47375, 0.43003, 0.53514, 0.51224, 0.47995, 0.52364,
0.29864, 0.35875, 0.26727, 0.39090, 0.50975, 0.46923, 0.29576, 0.22714,
0.31961, 0.39449, 0.43854, 0.46531, 0.52649, 0.51345, 0.39197, 0.46492)
> net1 <- newff(n.neurons=c(1,2,1), learning.rate.global=0.2,
+ momentum.global=0.5, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid", method="ADAPTgdwm")
> net1
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2

$layers[[3]]
[1] 3


$neurons
$neurons[[1]]
$id
[1] 1

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3
```

```
$output.aims
[1] 1

$input.links
[1] -1

$weights
[1] 0.5282752

$bias
[1] 0.4848078

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01fcb9c4>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01fcb9c4>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.2

$method.dep.variables$momentum
```

[1] 0.5

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0

attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
[1] 2

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 2

$input.links
[1] -1

$weights
[1] 0.5393306

$bias
[1] 0.2530626

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{

```
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01fbbfd8>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01fbbfd8>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.2

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[3]]
$id
[1] 3

$type
[1] "output"

$activation.function
[1] 2
```

```
$output.links
[1] NA

$output.aims
[1] 1

$input.links
[1] 1 2

$weights
[1] 0.4475340 0.6086728

$bias
[1] -0.02323091

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01fa8390>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01fa8390>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0
```

```
$method.dep.variables$learning.rate
[1] 0.2

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"


$input
[1] 0

$output
[1] 0

$target
[1] 0

$deltaE
$deltaE$fname
[1] 0

$deltaE$f
function (arguments)
{
   prediction <- arguments[[1]]
   target <- arguments[[2]]
   residual <- prediction - target
   return(residual)
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
list()
```

```
attr(,"class")
[1] "MLPnet"
> result<-train(net1,input,target,error.criterium="LMS",
+ report=TRUE,show.step=300,n.shows=10)
index.show: 1 LMS 0.0111005794363061
index.show: 2 LMS 0.0110102445645546
index.show: 3 LMS 0.0109311098654860
index.show: 4 LMS 0.0108904663335761
index.show: 5 LMS 0.0108745588223424
index.show: 6 LMS 0.0108673663438209
index.show: 7 LMS 0.0108628839041544
index.show: 8 LMS 0.0108593856434366
index.show: 9 LMS 0.0108563453390957
index.show: 10 LMS 0.0108535787851419
> result
$net
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2

$layers[[3]]
[1] 3


$neurons
$neurons[[1]]
$id
[1] 1

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 1

$input.links
```

```
[1] -1

$weights
[1] 0.6273876

$bias
[1] 0.2775019

$v0
[1] 0.6277344

$v1
[1] 0.2336839

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01fcb9c4>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01fcb9c4>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.0009438861

$method.dep.variables$learning.rate
[1] 0.2

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0.0001256357
```

```
$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
[1] 2

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 2

$input.links
[1] -1

$weights
[1] 1.228723

$bias
[1] -0.4126226

$v0
[1] 0.5166169

$v1
[1] 0.2497239

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01fbbfd8>
```

```
$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01fbbfd8>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.004723357

$method.dep.variables$learning.rate
[1] 0.2

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0.0006362066

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[3]]
$id
[1] 3

$type
[1] "output"

$activation.function
[1] 2

$output.links
[1] NA
```

```
$output.aims
[1] 1

$input.links
[1] 1 2

$weights
[1] 0.246080 1.152328

$bias
[1] -1.163421

$v0
[1] 0.3963137

$v1
[1] 0.2392491

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01fa8390>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01fa8390>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.016414

$method.dep.variables$learning.rate
[1] 0.2

$method.dep.variables$momentum
```

```
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0.003744648 0.003056620

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"


$input
[1] 0

$output
[1] 0

$target
[1] 0

$deltaE
$deltaE$fname
[1] 1

$deltaE$f
function (arguments)
{
   prediction <- arguments[[1]]
   target <- arguments[[2]]
   residual <- prediction - target
   return(residual)
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
$other.elements$Stao
[1] NA


attr(,"class")
```

```
[1] "MLPnet"

$Merror
           [,1]
 [1,] 0.01110058
 [2,] 0.01101024
 [3,] 0.01093111
 [4,] 0.01089047
 [5,] 0.01087456
 [6,] 0.01086737
 [7,] 0.01086288
 [8,] 0.01085939
 [9,] 0.01085635
[10,] 0.01085358

> y<-sim(result$net1,input)
> y
           [,1]
 [1,] 0.3981261
 [2,] 0.3728592
 [3,] 0.3745041
 [4,] 0.4016210
 [5,] 0.4429995
 [6,] 0.4366560
 [7,] 0.4074912
 [8,] 0.3988208
 [9,] 0.4444325
[10,] 0.4313061
[11,] 0.4173812
[12,] 0.4123908
[13,] 0.4051026
[14,] 0.3981261
[15,] 0.3892974
[16,] 0.4005431
[17,] 0.4115832
[18,] 0.3994559
[19,] 0.3956372
[20,] 0.4052986
[21,] 0.4023524
[22,] 0.4117384
[23,] 0.4047805
[24,] 0.3994559
[25,] 0.3892816
[26,] 0.3713621
[27,] 0.3799760
[28,] 0.3819380
```

[29,] 0.4005665
[30,] 0.3890032
[31,] 0.4080889
[32,] 0.3892714
[33,] 0.3787457
[34,] 0.3984753
[35,] 0.4081225
[36,] 0.4024524
[37,] 0.4012483
[38,] 0.3883494
[39,] 0.4058633
[40,] 0.4017808
[41,] 0.4115832
[42,] 0.4094521
[43,] 0.4064419
[44,] 0.4105135
[45,] 0.3895322
[46,] 0.3951261
[47,] 0.3866250
[48,] 0.3981261

Pelatihan kedua:

```
> net2 <- newff(n.neurons=c(1,2,1), learning.rate.global=0.1,
+  momentum.global=0.5, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid", method="ADAPTgdwm")
> net2
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2

$layers[[3]]
[1] 3


$neurons
```

```
$neurons[[1]]
$id
[1] 1

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 1

$input.links
[1] -1

$weights
[1] 0.2769573

$bias
[1] 0.4506469

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
    a.sigmoid <- 1
    return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01b69f18>

$f1
function (v)
{
    a.sigmoid <- 1
    return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
        v))^2)
}
```

<environment: 0x01b69f18>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0

attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
[1] 2

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 2

$input.links
[1] -1

$weights
[1] 0.3566978

```
$bias
[1] 0.3306377

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01b61c08>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01b61c08>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0
```

```
attr(,"class")
[1] "neuron"

$neurons[[3]]
$id
[1] 3

$type
[1] "output"

$activation.function
[1] 2

$output.links
[1] NA

$output.aims
[1] 1

$input.links
[1] 1 2

$weights
[1]  0.4999403 -0.5467695

$bias
[1] -0.311444

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01b2a93c>

$f1
function (v)
{
```

```
    a.sigmoid <- 1
    return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
        v))^2)
}
<environment: 0x01b2a93c>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"


$input
[1] 0

$output
[1] 0

$target
[1] 0

$deltaE
$deltaE$fname
[1] 0

$deltaE$f
function (arguments)
{
```

```
    prediction <- arguments[[1]]
    target <- arguments[[2]]
    residual <- prediction - target
    return(residual)
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
list()

attr(,"class")
[1] "MLPnet"
> result<-train(net2,input,target,error.criterium="LMS",
+ report=TRUE,show.step=300,n.shows=10)
index.show: 1 LMS 0.0110993664363406
index.show: 2 LMS 0.0110113404720059
index.show: 3 LMS 0.0109262969043736
index.show: 4 LMS 0.0108687584459981
index.show: 5 LMS 0.0108426114424298
index.show: 6 LMS 0.0108331897576646
index.show: 7 LMS 0.0108297576415381
index.show: 8 LMS 0.0108282341365132
index.show: 9 LMS 0.0108273596595856
index.show: 10 LMS 0.0108267383454241
> result
$net
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2

$layers[[3]]
[1] 3


$neurons
$neurons[[1]]
$id
[1] 1
```

```
$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 1

$input.links
[1] -1

$weights
[1] 0.9520718

$bias
[1] 0.2212835

$v0
[1] 0.6440403

$v1
[1] 0.2292524

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01b69f18>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01b69f18>

$method
[1] "ADAPTgdwm"
```

$method.dep.variables
$method.dep.variables$delta
[1] -0.003996349

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0.0002685144

$method.dep.variables$former.bias.change
[1] 0

attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
[1] 2

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 2

$input.links
[1] -1

$weights
[1] -0.7552703

$bias
[1] 0.3744386

```
$v0
[1] 0.5199259

$v1
[1] 0.2496030

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01b61c08>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01b61c08>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0.004297466

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] -0.0002883665

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"
```

```
$neurons[[3]]
$id
[1] 3

$type
[1] "output"

$activation.function
[1] 2

$output.links
[1] NA

$output.aims
[1] 1

$input.links
[1] 1 2

$weights
[1]  1.032975 -1.020241

$bias
[1] -0.5604944

$v0
[1] 0.394257

$v1
[1] 0.2388184

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01b2a93c>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
```

<environment: 0x01b2a93c>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.01687562

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0.001959916 0.001607820

$method.dep.variables$former.bias.change
[1] 0

attr(,"class")
[1] "neuron"

$input
[1] 0

$output
[1] 0

$target
[1] 0

$deltaE
$deltaE$fname
[1] 1

$deltaE$f
function (arguments)
{
    prediction <- arguments[[1]]
    target <- arguments[[2]]
    residual <- prediction - target
    return(residual)

```
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
$other.elements$Stao
[1] NA


attr(,"class")
[1] "MLPnet"

$Merror
          [,1]
 [1,] 0.01109937
 [2,] 0.01101134
 [3,] 0.01092630
 [4,] 0.01086876
 [5,] 0.01084261
 [6,] 0.01083319
 [7,] 0.01082976
 [8,] 0.01082823
 [9,] 0.01082736
[10,] 0.01082674

> y2<-sim(result$net,input)
> y2
          [,1]
 [1,] 0.3952231
 [2,] 0.3678114
 [3,] 0.3696266
 [4,] 0.3989560
 [5,] 0.4430349
 [6,] 0.4362256
 [7,] 0.4052080
 [8,] 0.3959659
 [9,] 0.4445796
[10,] 0.4305124
[11,] 0.4157120
[12,] 0.4104144
[13,] 0.4026663
[14,] 0.3952231
[15,] 0.3857428
```

[16,] 0.3978056
[17,] 0.4095566
[18,] 0.3966445
[19,] 0.3925584
[20,] 0.4028750
[21,] 0.3997361
[22,] 0.4097215
[23,] 0.4023234
[24,] 0.3966445
[25,] 0.3857258
[26,] 0.3661546
[27,] 0.3756286
[28,] 0.3777682
[29,] 0.3978306
[30,] 0.3854255
[31,] 0.4058435
[32,] 0.3857148
[33,] 0.3742838
[34,] 0.3955965
[35,] 0.4058792
[36,] 0.3998427
[37,] 0.3985584
[38,] 0.3847197
[39,] 0.4034761
[40,] 0.3991264
[41,] 0.4095566
[42,] 0.4072926
[43,] 0.4040917
[44,] 0.4084204
[45,] 0.3859961
[46,] 0.3920105
[47,] 0.3828557
[48,] 0.3952231

Pelatihan Ketiga:

```
> net3 <- newff(n.neurons=c(1,2,1), learning.rate.global=0.01,
+ momentum.global=0.5, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid", method="ADAPTgdwm")
> net3
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2

$layers[[3]]
[1] 3


$neurons
$neurons[[1]]
$id
[1] 1

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 1

$input.links
[1] -1

$weights
[1] 0.4505978

$bias
[1] 0.4291132

$v0
[1] 0
```

```
$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x0168a4d0>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x0168a4d0>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.01

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
[1] 2
```

```
$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 2

$input.links
[1] -1

$weights
[1] -0.6230671

$bias
[1] -0.005967374

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01673bcc>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01673bcc>

$method
```

[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.01

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0

attr(,"class")
[1] "neuron"

$neurons[[3]]
$id
[1] 3

$type
[1] "output"

$activation.function
[1] 2

$output.links
[1] NA

$output.aims
[1] 1

$input.links
[1] 1 2

$weights
[1] 0.5783356 -0.2331131

$bias
[1] 0.2674810

```
$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01638a0c>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01638a0c>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.01

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"
```

```
$input
[1] 0

$output
[1] 0

$target
[1] 0

$deltaE
$deltaE$fname
[1] 0

$deltaE$f
function (arguments)
{
   prediction <- arguments[[1]]
   target <- arguments[[2]]
   residual <- prediction - target
   return(residual)
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
list()

attr(,"class")
[1] "MLPnet"
> result<-train(net3,input,target,error.criterium="LMS",
+ report=TRUE,show.step=300,n.shows=10)
index.show: 1 LMS 0.0110191956570256
index.show: 2 LMS 0.0110087124510327
index.show: 3 LMS 0.0109981845142511
index.show: 4 LMS 0.0109876655784154
index.show: 5 LMS 0.0109772124552196
index.show: 6 LMS 0.0109668835387313
index.show: 7 LMS 0.0109567372187331
index.show: 8 LMS 0.0109468302796451
index.show: 9 LMS 0.0109372163634174
index.show: 10 LMS 0.0109279445714502
```

```
> result
$net
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2

$layers[[3]]
[1] 3


$neurons
$neurons[[1]]
$id
[1] 1

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 1

$input.links
[1] -1

$weights
[1] 0.4561591

$bias
[1] 0.3967849

$v0
[1] 0.6399366

$v1
[1] 0.2304177

$f0
```

```
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x0168a4d0>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x0168a4d0>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.000663162

$method.dep.variables$learning.rate
[1] 0.01

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 4.320408e-06

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
[1] 2

$type
[1] "hidden"
```

```
$activation.function
[1] 2

$output.links
[1] 3

$output.aims
[1] 2

$input.links
[1] -1

$weights
[1] -0.82493

$bias
[1] 0.05571253

$v0
[1] 0.4337155

$v1
[1] 0.2456064

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01673bcc>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01673bcc>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
```

[1] 0.003568362

$method.dep.variables$learning.rate
[1] 0.01

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] -2.32671e-05

$method.dep.variables$former.bias.change
[1] 0

attr(,"class")
[1] "neuron"

$neurons[[3]]
$id
[1] 3

$type
[1] "output"

$activation.function
[1] 2

$output.links
[1] NA

$output.aims
[1] 1

$input.links
[1] 1 2

$weights
[1] 0.1710714 -0.8635806

$bias
[1] -0.1630680

$v0
[1] 0.394488

```
$v1
[1] 0.2388672

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01638a0c>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01638a0c>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.01682389

$method.dep.variables$learning.rate
[1] 0.01

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0.0001875204 0.0001285655

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"


$input
[1] 0
```

```
$output
[1] 0

$target
[1] 0

$deltaE
$deltaE$fname
[1] 1

$deltaE$f
function (arguments)
{
   prediction <- arguments[[1]]
   target <- arguments[[2]]
   residual <- prediction - target
   return(residual)
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
$other.elements$Stao
[1] NA


attr(,"class")
[1] "MLPnet"

$Merror
           [,1]
 [1,] 0.01101920
 [2,] 0.01100871
 [3,] 0.01099818
 [4,] 0.01098767
 [5,] 0.01097721
 [6,] 0.01096688
 [7,] 0.01095674
 [8,] 0.01094683
 [9,] 0.01093722
[10,] 0.01092794
```

```
> y3<-sim(result$net,input)
> y3
          [,1]
 [1,] 0.3945726
 [2,] 0.3818631
 [3,] 0.3827092
 [4,] 0.3962949
 [5,] 0.4165899
 [6,] 0.4134535
 [7,] 0.3991768
 [8,] 0.3949154
 [9,] 0.4173019
[10,] 0.4108236
[11,] 0.4040131
[12,] 0.4015746
[13,] 0.3980056
[14,] 0.3945726
[15,] 0.3901910
[16,] 0.3957643
[17,] 0.4011797
[18,] 0.3952286
[19,] 0.3933422
[20,] 0.3981017
[21,] 0.3966547
[22,] 0.4012556
[23,] 0.3978475
[24,] 0.3952286
[25,] 0.3901832
[26,] 0.3810902
[27,] 0.3855014
[28,] 0.3864950
[29,] 0.3957758
[30,] 0.3900442
[31,] 0.3994696
[32,] 0.3901781
[33,] 0.3848764
[34,] 0.3947449
[35,] 0.3994860
[36,] 0.3967038
[37,] 0.3961115
[38,] 0.3897175
[39,] 0.3983788
[40,] 0.3963735
[41,] 0.4011797
[42,] 0.4001371
[43,] 0.3986625
```

[44,] 0.4006564
[45,] 0.3903083
[46,] 0.3930891
[47,] 0.3888542
[48,] 0.3945726

Pelatihan Keempat:

```
> net4 <- newff(n.neurons=c(1,3,1), learning.rate.global=0.1,
+ momentum.global=0.4, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid", method="ADAPTgdwm")
> net4
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2 3

$layers[[3]]
[1] 4


$neurons
$neurons[[1]]
$id
```

```
[1] 1

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 1

$input.links
[1] -1

$weights
[1] -0.5214052

$bias
[1] -0.5035302

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01a89d30>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01a89d30>
```

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.4

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0

attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
[1] 2

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 2

$input.links
[1] -1

$weights
[1] -0.1172197

$bias

```
[1] -0.1560024

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x018acafc>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x018acafc>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.4

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
```

```
[1] "neuron"

$neurons[[3]]
$id
[1] 3

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 3

$input.links
[1] -1

$weights
[1] 0.2650663

$bias
[1] -0.2034504

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x017817d4>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
```

```
      v))^2)
}
<environment: 0x017817d4>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.4

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[4]]
$id
[1] 4

$type
[1] "output"

$activation.function
[1] 2

$output.links
[1] NA

$output.aims
[1] 1

$input.links
[1] 1 2 3
```

```
$weights
[1] 0.2450736 0.4359519 0.4414996

$bias
[1] -0.2817161

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01a7f144>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01a7f144>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.4

$method.dep.variables$former.weight.change
[1] 0 0 0

$method.dep.variables$former.bias.change
```

```
[1] 0


attr(,"class")
[1] "neuron"


$input
[1] 0

$output
[1] 0

$target
[1] 0

$deltaE
$deltaE$fname
[1] 0

$deltaE$f
function (arguments)
{
   prediction <- arguments[[1]]
   target <- arguments[[2]]
   residual <- prediction - target
   return(residual)
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
list()

attr(,"class")
[1] "MLPnet"
> result<-train(net4,input,target,error.criterium="LMS",
+ report=TRUE,show.step=300,n.shows=10)
index.show: 1 LMS 0.0110994675701486
index.show: 2 LMS 0.0110303057821422
index.show: 3 LMS 0.0109547564619149
index.show: 4 LMS 0.0108926190948405
index.show: 5 LMS 0.0108555655637460
```

```
index.show: 6 LMS 0.0108383534485945
index.show: 7 LMS 0.0108312754262704
index.show: 8 LMS 0.0108283099231956
index.show: 9 LMS 0.0108268841792264
index.show: 10 LMS 0.0108260476891620
> result
$net
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2 3

$layers[[3]]
[1] 4


$neurons
$neurons[[1]]
$id
[1] 1

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 1

$input.links
[1] -1

$weights
[1] -0.842979

$bias
[1] -0.3974077

$v0
[1] 0.3259536
```

```
$v1
[1] 0.2197079

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01a89d30>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01a89d30>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0.003151834

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.4

$method.dep.variables$former.weight.change
[1] -0.0001767013

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
```

```
[1] 2

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 2

$input.links
[1] -1

$weights
[1] 0.1219117

$bias
[1] -0.1813609

$v0
[1] 0.4666101

$v1
[1] 0.2488851

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x018acafc>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x018acafc>
```

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.0004360884

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.4

$method.dep.variables$former.weight.change
[1] 2.413672e-05

$method.dep.variables$former.bias.change
[1] 0

attr(,"class")
[1] "neuron"

$neurons[[3]]
$id
[1] 3

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 3

$input.links
[1] -1

$weights
[1] 1.033161

$bias

```
[1] -0.3275639

$v0
[1] 0.5189366

$v1
[1] 0.2496414

$f0
function (v)
{
    a.sigmoid <- 1
    return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x017817d4>

$f1
function (v)
{
    a.sigmoid <- 1
    return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
        v))^2)
}
<environment: 0x017817d4>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.00423838

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.4

$method.dep.variables$former.weight.change
[1] 0.0002363740

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
```

```
[1] "neuron"

$neurons[[4]]
$id
[1] 4

$type
[1] "output"

$activation.function
[1] 2

$output.links
[1] NA

$output.aims
[1] 1

$input.links
[1] 1 2 3

$weights
[1] -0.8483537  0.1036179  1.0040202

$bias
[1] -0.7197575

$v0
[1] 0.3941039

$v1
[1] 0.238786

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01a7f144>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
```

```
      v))^2)
}
<environment: 0x01a7f144>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.01690989

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.4

$method.dep.variables$former.weight.change
[1] 0.0008307314 0.0011761844 0.0012982944

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"


$input
[1] 0

$output
[1] 0

$target
[1] 0

$deltaE
$deltaE$fname
[1] 1

$deltaE$f
function (arguments)
{
   prediction <- arguments[[1]]
   target <- arguments[[2]]
```

```
   residual <- prediction - target
   return(residual)
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
$other.elements$Stao
[1] NA


attr(,"class")
[1] "MLPnet"

$Merror
          [,1]
 [1,] 0.01109947
 [2,] 0.01103031
 [3,] 0.01095476
 [4,] 0.01089262
 [5,] 0.01085557
 [6,] 0.01083835
 [7,] 0.01083128
 [8,] 0.01082831
 [9,] 0.01082688
[10,] 0.01082605

> y4<-sim(result$net,input)
> y4
          [,1]
 [1,] 0.3949129
 [2,] 0.3674491
 [3,] 0.3692639
 [4,] 0.3986576
 [5,] 0.4426781
 [6,] 0.4359229
 [7,] 0.4049279
 [8,] 0.3956580
 [9,] 0.4442069
[10,] 0.4302375
[11,] 0.4154526
[12,] 0.4101467
[13,] 0.4023791
```

[14,] 0.3949129
[15,] 0.3854045
[16,] 0.3975036
[17,] 0.4092872
[18,] 0.3963388
[19,] 0.3922399
[20,] 0.4025884
[21,] 0.3994401
[22,] 0.4094524
[23,] 0.4020351
[24,] 0.3963388
[25,] 0.3853874
[26,] 0.3657933
[27,] 0.3752698
[28,] 0.3774125
[29,] 0.3975286
[30,] 0.3850863
[31,] 0.4055651
[32,] 0.3853764
[33,] 0.3739235
[34,] 0.3952875
[35,] 0.4056009
[36,] 0.3995469
[37,] 0.3982587
[38,] 0.3843787
[39,] 0.4031911
[40,] 0.3988285
[41,] 0.4092872
[42,] 0.4070179
[43,] 0.4038085
[44,] 0.4081484
[45,] 0.3856584
[46,] 0.3916903
[47,] 0.3825101
[48,] 0.3949129

Pelatihan Kelima:

```
> net5 <- newff(n.neurons=c(1,3,1), learning.rate.global=0.1,
+ momentum.global=0.5, error.criterium="LMS", Stao=NA,
+ hidden.layer="sigmoid", output.layer="sigmoid", method="ADAPTgdwm")
> net5
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2 3

$layers[[3]]
[1] 4


$neurons
$neurons[[1]]
$id
[1] 1

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 1
```

```
$input.links
[1] -1

$weights
[1] -0.397081

$bias
[1] 0.4932445

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01a23f18>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01a23f18>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
```

```
[1] 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
[1] 2

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 2

$input.links
[1] -1

$weights
[1] 0.09283296

$bias
[1] -0.5338356

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
```

```
<environment: 0x018d8490>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x018d8490>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[3]]
$id
[1] 3

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4
```

```
$output.aims
[1] 3

$input.links
[1] -1

$weights
[1] -0.1266459

$bias
[1] 0.2822449

$v0
[1] 0

$v1
[1] 0

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01853684>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01853684>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1
```

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0

$method.dep.variables$former.bias.change
[1] 0

attr(,"class")
[1] "neuron"

$neurons[[4]]
$id
[1] 4

$type
[1] "output"

$activation.function
[1] 2

$output.links
[1] NA

$output.aims
[1] 1

$input.links
[1] 1 2 3

$weights
[1] 0.4481588 -0.3049654 -0.5236728

$bias
[1] -0.3792364

$v0
[1] 0

$v1
[1] 0

$f0
function (v)

```
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01723eb4>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01723eb4>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0 0 0

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"


$input
[1] 0

$output
[1] 0

$target
```

```
[1] 0

$deltaE
$deltaE$fname
[1] 0

$deltaE$f
function (arguments)
{
   prediction <- arguments[[1]]
   target <- arguments[[2]]
   residual <- prediction - target
   return(residual)
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
list()

attr(,"class")
[1] "MLPnet"
> result<-train(net5,input,target,error.criterium="LMS",
+ report=TRUE,show.step=400,n.shows=10)
index.show: 1 LMS 0.0110652766655848
index.show: 2 LMS 0.0109324843654636
index.show: 3 LMS 0.010856178211219
index.show: 4 LMS 0.0108350128140282
index.show: 5 LMS 0.0108301248614665
index.show: 6 LMS 0.0108284976019143
index.show: 7 LMS 0.0108276139932975
index.show: 8 LMS 0.0108269563466561
index.show: 9 LMS 0.010826391766739
index.show: 10 LMS 0.0108258796989045
> result
$net
$layers
$layers[[1]]
[1] -1

$layers[[2]]
[1] 1 2 3
```

```
$layers[[3]]
[1] 4


$neurons
$neurons[[1]]
$id
[1] 1

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 1

$input.links
[1] -1

$weights
[1] 0.3728586

$bias
[1] 0.4657969

$v0
[1] 0.648222

$v1
[1] 0.2280302

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01a23f18>

$f1
function (v)
```

```
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01a23f18>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.002490672

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0.000166772

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[2]]
$id
[1] 2

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 2
```

```
$input.links
[1] -1

$weights
[1] -0.275218

$bias
[1] -0.5633243

$v0
[1] 0.3383228

$v1
[1] 0.2238605

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x018d8490>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x018d8490>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0.001026653

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
```

```
[1] -6.923705e-05

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[3]]
$id
[1] 3

$type
[1] "hidden"

$activation.function
[1] 2

$output.links
[1] 4

$output.aims
[1] 3

$input.links
[1] -1

$weights
[1] -1.105062

$bias
[1] 0.3894732

$v0
[1] 0.4895417

$v1
[1] 0.2498906

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
```

```
<environment: 0x01853684>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01853684>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] 0.005197591

$method.dep.variables$learning.rate
[1] 0.1

$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] -0.0003489047

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"

$neurons[[4]]
$id
[1] 4

$type
[1] "output"

$activation.function
[1] 2

$output.links
[1] NA
```

```
$output.aims
[1] 1

$input.links
[1] 1 2 3

$weights
[1]  0.6483955 -0.2722465 -1.2347186

$bias
[1] -0.1485786

$v0
[1] 0.3943915

$v1
[1] 0.2388468

$f0
function (v)
{
   a.sigmoid <- 1
   return(1/(1 + exp(-a.sigmoid * v)))
}
<environment: 0x01723eb4>

$f1
function (v)
{
   a.sigmoid <- 1
   return(a.sigmoid * exp(-a.sigmoid * v)/(1 + exp(-a.sigmoid *
      v))^2)
}
<environment: 0x01723eb4>

$method
[1] "ADAPTgdwm"

$method.dep.variables
$method.dep.variables$delta
[1] -0.01684551

$method.dep.variables$learning.rate
[1] 0.1
```

```
$method.dep.variables$momentum
[1] 0.5

$method.dep.variables$former.weight.change
[1] 0.001978503 0.001040017 0.001517830

$method.dep.variables$former.bias.change
[1] 0


attr(,"class")
[1] "neuron"


$input
[1] 0

$output
[1] 0

$target
[1] 0

$deltaE
$deltaE$fname
[1] 1

$deltaE$f
function (arguments)
{
   prediction <- arguments[[1]]
   target <- arguments[[2]]
   residual <- prediction - target
   return(residual)
}
<environment: namespace:AMORE>

$deltaE$Stao
[1] NA


$other.elements
$other.elements$Stao
[1] NA
```

```
attr(,"class")
[1] "MLPnet"

$Merror
           [,1]
 [1,] 0.01106528
 [2,] 0.01093248
 [3,] 0.01085618
 [4,] 0.01083501
 [5,] 0.01083012
 [6,] 0.01082850
 [7,] 0.01082761
 [8,] 0.01082696
 [9,] 0.01082639
[10,] 0.01082588

> y5<-sim(result$net,input)
> y5
           [,1]
 [1,] 0.3954235
 [2,] 0.3687336
 [3,] 0.3704734
 [4,] 0.3991196
 [5,] 0.4434445
 [6,] 0.4365480
 [7,] 0.4053371
 [8,] 0.3961579
 [9,] 0.4450095
[10,] 0.4307680
[11,] 0.4158497
[12,] 0.4105382
[13,] 0.4028055
[14,] 0.3954235
[15,] 0.3860977
[16,] 0.3979792
[17,] 0.4096800
[18,] 0.3968294
[19,] 0.3927931
[20,] 0.4030132
[21,] 0.3998936
[22,] 0.4098450
[23,] 0.4024643
[24,] 0.3968294
[25,] 0.3860811
[26,] 0.3671494
[27,] 0.3762559
```

[28,] 0.3783279
[29,] 0.3980039
[30,] 0.3857872
[31,] 0.4059709
[32,] 0.3860703
[33,] 0.3749563
[34,] 0.3957927
[35,] 0.4060065
[36,] 0.3999994
[37,] 0.3987253
[38,] 0.3850969
[39,] 0.4036115
[40,] 0.3992886
[41,] 0.4096800
[42,] 0.4074172
[43,] 0.4042246
[44,] 0.4085439
[45,] 0.3863457
[46,] 0.3922532
[47,] 0.3832764
[48,] 0.3954235

Pengujian Data ke 55 (Bulan Juli 2008):

```
> input<-matrix(0,19,1)
> input[,1]<-c( 0.438544)
> y<-sim(net5,input)
> y
          [,1]
 [1,] 0.3712812
```

Prediksi Data ke-68 (Bulan Agustus 2009):

```
>  input<-matrix(0,1,1)
> input[,1]<-c(0.51346)
> y<-sim(net5,input)
> y
         [,1]
[1,] 0.3706909
```