PERAMALAN PENJUALAN SEMEN MENGGUNAKAN BACKPROPAGATION NEURAL NETWORK DAN RECURRENT NEURAL NETWORK

(Studi kasus di PT Semen Indonesia (Persero) Tbk)

Skripsi

disusun sebagai salah satu syarat

untuk memperoleh gelar Sarjana Sains

Program Studi Matematika

Oleh

Aisyah Fany Achmalia

4111414005

JURUSAN MATEMATIKA

FAKULTAS ILMU PENGETAHUAN ALAM DAN MATEMATIKA

UNIVERSITAS NEGERI SEMARANG

2019

PERNYATAAN

Saya menyatakan bahwa skripsi ini bebas plagiat dan apabila di kemudian hari terbukti terdapat plagiat dalam skripsi ini, maka saya akan bersedia menerima sanksi sesuai ketentuan perundang-undangan.

Semarang, Maret 2019

Aisyah Fany Achmalia

4111414005

PENGESAHAN

Skripsi yang berjudul

Peramalan Penjualan Semen Menggunakan *Backpropagation Neural Network* dan *Recurrent Neural Network* (Studi kasus di PT Semen Indonesia (Persero) Tbk)

Disusun oleh

Nama : Aisyah Fany Achmalia

NIM : 4111414005

Telah dipertahankan di hadapan sidang Panitia Ujian Skripsi FMIPA Unnes pada

Hari : Senin

Tanggal : 18 Maret 2019

Panitia,

Ketua

Prof. Dr. Sudarmin, M.Si.

NIP 196601231992031003

Sekretaris

Drs. Arief Agoestanto, M.Si.

NIP 196807221993031005

Ketua Penguji

Dr. Scolastika Mariani, M.Si.

NIP 196502101991022001

Anggota Penguji/

Pembimbing I

Dr. Walid, S.Pd., M.Si.

NIP 197408192001121001

Anggota Penguji/

Pembimbing II

Drs. Sugiman, M.Si.

NIP 196401111989011001

MOTTO

Jika kamu ridho atas apa yang menjadi bagianmu, sungguh kamu ada dalam kenikmatan hidup. Tapi jika kamu tidak ridho atas apa yang telah menjadi bagianmu sungguh kamu akan berada dalam kesedihan hidup (HR. Al Imam Abdullah Al-Haddad)

Yaitu orang-orang yang beriman dan hati mereka menjadi tenteram dengan mengingat Allah. Ingatlah, hanya dengan mengingat Allah-lah hati menjadi tenteram (QS. Ar Ra'd: 28)

Lakukanlah kebaikan sekecil apapun, karena engkau tidak pernah tahu kebaikan mana yang akan membawamu ke surga (HR. Imam Hasan Al-Basri)

Tidak ada suatu rezeki yang Allah berikan kepada seorang hamba yang lebih luas baginya dari pada sabar (HR. Al Hakim)

PERSEMBAHAN

Untuk Bapak, Ibu, Adik, Saudara, Sahabat,
Teman, dan Almamater

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya, sehingga skripsi dengan judul "Peramalan Penjualan Semen Menggunakan *Backpropagation Neural Network* dan *Recurrent Neural Network* (Studi kasus di PT Semen Indonesia (Persero) Tbk)" dapat diselesaikan dengan baik. Penyusunan skripsi ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Sains.

Penyusunan skripsi ini dapat terselesaikan tidak lepas dari bantuan dan bimbingan berbagai pihak. Oleh karena itu, pada kesempatan ini penulis dengan rendah hati mengucapkan terima kasih kepada:

1. Prof. Dr. Fathur Rokhman, M.Hum, Rektor Universitas Negeri Semarang.
2. Prof. Dr. Sudarmin, M.Si., Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
3. Drs. Arief Agoestanto, M.Si., Ketua Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
4. Drs. Mashuri, M.Si., Ketua Prodi Matematika Ketua Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
5. Dr. Walid S.Pd., M.Si., Dosen Pembimbing I yang telah sabar dan ikhlas memberikan bimbingan, arahan, nasihat, motivasi, dan saran-saran yang sangat berguna selama penyusunan skripsi ini.
6. Drs. Sugiman M.Si., Dosen Pembimbing II yang telah sabar dan ikhlas memberikan bimbingan, arahan, nasihat, motivasi, dan saran-saran yang sangat berguna selama penyusunan skripsi ini.
7. Dr. Scolastika Mariani, M.Si., Dosen Penguji dan Dosen Wali sekaligus orang tua yang telah memberikan penilaian dan saran dalam perbaikan skripsi ini, serta arahan dan bimbingannya selama penulis menempuh perkuliahan.

8. Dosen Jurusan Matematika Universitas Negeri Semarang yang telah membekali penulis dengan berbagai ilmu selama mengikuti perkuliahan sampai akhir penulisan skripsi ini.
9. Bapak dan Ibu tercinta, Bapak Akhmad dan Ibu Yuliana, yang telah membimbing dan memberikan dukungan penuh secara moril maupun materil serta kasih sayang, ridho, nasihat, dan doa yang selalu menyertai setiap langkah penulis.
10. Keluarga besar penulis, khususnya Afif Aji Prasetya, Alfi Hasna Afiqa, Frisha Laksmi Harimurti, Mas Udin, De Ana, De Mego, Tante Ika, Dek Faqih, Dek Ina, Dek Dani, Mas Ragil yang senantiasa memberi doa, nasihat, dan dukungan yang tiada putusnya.
11. Teman-teman seperjuangan di Universitas Negeri Semarang, khususnya Tiffani Dita Permata Putri, S.Si., Leni Lestifahmawati Ningsih, S.Si., Dewi Ariyanti, Fitra Sukma Amorizki, Frida Anggraeni Setyowati, Itsnaini M. Fadlilah, S.Si., Ernia Umaya Sari, S.Si., Destantya Devi Masita, S.H., Bertania Rizky Triayuni, S.H., Risang Aji Prakoso, S.H., Jeslin Eka Putri, S.H., sebagai teman seperjuangan yang hebat.
12. Teman-teman Numthai, Muhammad Abi Mashkun, Afi Kholidah, Aji Prasetyo, Ridho Hakiki Prasetyo, S,Sn., Charisma Fitra Andriyan, Devia Nur Safitri sebagai teman seperjuangan mengumpulkan pundi-pundi kekayaan dan pencari jati diri yang hebat.
13. Teman-teman Matematika Unnes 2014 yang selalu berjuang bersama dan rekan yang hebat.
14. Keluarga Mbois Ilakes sebagai teman-teman yang hebat yang selalu memberi dukungan dan semangat.
15. Kakak-kakak dan adik-adik ketemu gede, khususnya Mas Agustaf, Mbak Reni, Nabila, Biru, Yohan, Arif, Variana, Ajeng, Puput yang senantiasa memberi dukungan dan semangat.
16. Semua pihak yang telah memberi dukungan dan bantuan dalam penyelesaian skripsi ini.

Hanya ucapan syukur, terimakasih, dan doa, semoga jasa-jasa yang telah diberikan tercatat sebagai amal baik dan mendapatkan balasan dari Allah SWT. Penulis menyadari bahwa dalam penyusunan skripsi ini masih terdapat banyak kekurangan. Oleh karena itu, penulis mengharapkan saran dan kritik yang bersifat membangun dari pembaca. Semoga skripsi ini bermanfaat bagi semua pihak yang membutuhkan.

Semarang, Maret 2019

Penulis

ABSTRAK

Achmalia, Aisyah Fany. 2019. *Peramalan Penjualan Semen Menggunakan Backpropagation Neural Network dan Recurrent Neural Network (Studi Kasus di PT Semen Indonesia (Persero) Tbk)*. Skripsi. Jurusan Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang. Pembimbing: Dr. Walid, S.Pd, M.Si., dan Drs. Sugiman, M.Si.

Kata Kunci: *Backpropagation Neural Network*, Penjualan, Peramalan, *Recurrent Neural Network*

Backpropagation Neural Netwrok (BPNN) adalah *Neural Network* (NN) yang bergerak maju dan tidak memiliki loop dimana aliran sinyalnya dari neuron *input* ke neuron *output*, sedangkan *Recurrent Neural Network* (RNN) adalah model NN yang arsitekturnya memiliki minimal satu *feedback loop*, sehingga dapat menyimpan data dalam struktur jaringannya. Dalam penelitian ini dilakukan peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk dengan menggunakan BPNN dan RNN tipe Elman. Tujuan penelitian ini adalah untuk memperoleh pemodelan BPNN dan RNN tipe Elman untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk, serta hasil peramalan dengan menggunakan model terbaik. *Input* jaringan dalam penelitian ini diperoleh berdasarkan lag-lag yang signifikan pada plot PACF. Pembentukan model terbaik dalam penelitian ini menggunakan beberapa variasi pada neuron tersembunyi, fungsi aktivasi, algoritma pelatihan, dan parameter-parameter pelatihan. Pemilihan model jaringan terbaik dengan melihat MSE dan MAPE terkecil pada tahap pengujian. Data yang digunakan adalah *volume* penjualan semen di PT Semen Indonesia (Persero) Tbk pada Bulan Januari 2006 sampai dengan Bulan Desember 2018.

Hasil dari penelitian ini menghasilkan bahwa model BPNN terbaik adalah model BPNN (9-5-1) dengan algoritma pelatihan *Levenberg-Marquardt* dengan inisialisasi Mu yang digunakan adalah 0,02 dan fungsi aktivasi yang digunakan adalah logsig, sedangkan model RNN tipe Elman terbaik adalah model RNN tipe Elman (9-5-1) dengan algoritma pelatihan *gradient descent* dengan momentum dan *adaptive learning rate* dengan momentum yang digunakan adalah 0,2, *learning rate* yang digunakan adalah 0,2, dan fungsi aktivasi yang digunakan adalah logsig. Model terbaik untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk adalah model BPNN (9-5-1) dengan hasil peramalan untuk Bulan April 2018 sebesar 2479607 ton, Bulan Mei 2018 sebesar 2344701 ton, Bulan Juni 2018 sebesar 2045132 ton, Bulan Juli 2018 sebesar 2486581 ton, Bulan Agustus 2018 sebesar 2674669 ton, Bulan September 2018 sebesar 2379005 ton, Bulan Oktober 2018 sebesar 2834896 ton, Bulan November 2018 sebesar 2501668 ton, dan Bulan Desember 2018 sebesar 2918820 ton.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi dari hari ke hari semakin canggih dan berkembang dengan pesat. Perkembangan teknologi saat ini tidak dapat dipisahkan dari kehidupan manusia. Perkembangan pada zaman sekarang ini cenderung untuk mengembangkan teknologi yang cerdas dengan memiliki kemampuan untuk berpikir dan mengambil keputusan layaknya manusia. Kecerdasan teknologi diharapkan mampu membantu berbagai persoalan dalam kehidupan sehari-hari dengan cepat dan akurat.

Banyak kecerdasan buatan yang dapat diterapkan dalam banyak bidang dalam kehidupan. Para ahli mencoba untuk mengadaptasi otak manusia ke dalam sistem komputer sehingga diharapkan di masa yang akan datang kecerdasan buatan tersebut dapat mendekati kerja otak manusia. Penerapan kecerdasan buatan yang sering diaplikasikan dalam berbagai persoalan dikehidupan salah satunya adalah jaringan syaraf tiruan.

Jaringan Syaraf Tiruan (JST) atau yang sering dikenal dengan *Neural Network* (NN) adalah sistem pemrosesan informasi yang memiliki karakteristik mirip dengan jaringan syaraf biologis (Fausett, 1994: 3). NN merupakan salah satu dari sistem informasi yang didesain dengan menirukan kerja otak manusia dalam menyelesaikan suatu masalah dengan melakukan proses belajar melalui perubahan

bobot sinapsisnya. NN telah dikembangkan sebagai generalisasi model matematika dari kognisi manusia atau syaraf biologi. NN ditandai dengan pola hubungan antara neuron (arsitektur), algoritma untuk menentukan bobot penghubung (pelatihan, atau belajar, algoritma), dan fungsi aktivasi (Fausett, 1994: 3). NN bermanfaat untuk pengenalan pola, *signal processing*, pengklasifikasian dan peramalan (Siang, 2004: 5).

Setiap perusahaan dituntut untuk merencanakan produksinya seefektif mungkin agar dapat memaksimalkan keuntungan. Salah satu alat untuk membuat perencanaan produksi yang baik adalah dengan melakukan estimasi terhadap pasar potensial yang dapat dikuasai oleh perusahaan. Estimasi penjualan atau peramalan penjualan pada hakekatnya adalah perkiraan yang dapat digunakan sebagai target penjualan perusahaan di masa mendatang. Menurut Arsyad (1994: 7), peramalan menjadi *input* bagi proses perencanaan dan pengambilan keputusan. Peramalan menunjukkan perkiraan yang akan terjadi pada suatu keadaan tertentu dan perencanaan menggunakan peramalan tersebut untuk membantu para pengambil keputusan dalam memilih alternatif terbaik jika penjualan yang diramalkan menurun.

Menurut Nafarin (2007: 96), peramalan penjualan merupakan faktor penting dalam perencanaan perusahaan karena peramalan penjualan menentukan anggaran penjualan dan anggaran penjualan menentukan anggaran produk, anggaran biaya pabrik, anggaran beban usaha, anggaran kas, anggaran laba rugi, dan anggaran neraca. Peramalan penjualan merupakan bagian fungsi manajemen sebagai salah satu kontributor keberhasilan sebuah perusahaan. Ketika penjualan

diprediksi dengan akurat maka pemenuhan permintaan konsumen dapat diusahakan tepat waktu, kerjasama perusahaan dengan relasi tetap terjaga dengan baik, kepuasan konsumen terpenuhi, perusahaan dapat mengatasi hilangnya penjualan atau kehabisan stok, dan mencegah pelanggan lari ke kompetitor. Perusahaan juga dapat menentukan keputusan kebijakan rencana produksi, persediaan barang, investasi aktiva dan *cash flow*. Oleh karena kemanfaatannya, maka tidak ada perusahaan yang dapat menghindar dari kegiatan memperkirakan atau meramalkan penjualan untuk keperluan perencanaan aktivitas-aktivitas yang harus dilakukan.

PT Semen Indonesia (Persero) Tbk adalah produsen semen tebesar di Indonesia dan memiliki anak perusahaan yaitu PT Semen Gresik, PT Semen Padang, PT Semen Tonasa, dan *Thang Long Cement Company* yang tergabung dalam Semen Indonesia *Group*. Saat ini kapasitas terpasang PT Semen Indonesia (Persero) Tbk sebesar 29 juta ton semen per tahun dan menguasai sekitar 42% pangsa pasar domestik (Wikipedia, 2018).

Persaingan bisnis semen di Indonesia semakin kompetitif seiring dengan gencarnya penambahan pabrik baru dan masuknya investor baru ke bisnis ini. Sebagai perusahaan produsen semen terkemuka di Indonesia, PT Semen Indonesia (Persero) Tbk harus selalu berupaya menjadi yang terbaik untuk mempertahankan posisinya dengan terus meningkatkan penjualan produknya seiring dengan ketatnya persaingan bisnis (Portal BUMN, 2013). Berdasarkan laporan Asosiasi Semen Indonesia (ASI) di Bulan Januari sampai dengan Bulan November 2017 tercatat permintaan semen domestik mencapai 60,55 juta ton atau naik 7,8% dibandingkan periode yang sama tahun 2016 (Kontan, 2017). Saat ini permintaan semen curah

mengalami peningkatkan. Jika sebelumnya porsi penjualan semen curah nasional hanya 20%, saat ini porsinya sudah mencapai 25% (Portal BUMN, 2018). Ketua Umum ASI Widodo Santoso menyatakan kenaikan permintaan semen nasional bukan hanya terjadi di pasar domestik, tapi permintaan semen juga melonjak di pasar ekspor (Semen Indonesia, 2017). Oleh karena itu, untuk memenuhi permintaan semen yang terus meningkat diperlukan perencanaan dan pengawasan manajemen perusahaan yang baik.

Salah satu keberhasilan suatu perusahaan ditentukan oleh perencanaan dalam manajemennya. Perencanaan dalam manajemen perusahaan harus meliputi disegala bidang, salah satunya adalah bidang penjualan dengan menyusun rencana penjualan. Rencana penjualan salah satunya berbentuk peramalan penjualan, dengan dilakukannya peramalan penjualan yang akurat dapat meningkatkan penjualan produk dan mendatangkan laba bagi perusahaan.

Peramalan adalah suatu teknik untuk meramalkan keadaan di masa yang akan datang melalui pengujian keadaan di masa lalu. Dasarnya meramalkan sama halnya dengan memprediksi atau memperkirakan suatu hal, kejadian atau peristiwa masa datang yang berdasar data pada masa lalu (*time series* atau runtun waktu) hingga saat ini. Meramalkan masalah yang terkait dengan pembentukan model dan metode yang dapat digunakan untuk menghasilkan prakiraan yang akurat. Pemodelan bertujuan untuk mendapatkan model statistik yang tepat untuk mempresentasikan perilaku dari data runtun waktu jangka panjang (Walid & Alamsyah, 2017). Metode peramalan sangat banyak dan seringkali memerlukan

asumsi-asumsi yang harus dipenuhi, namun terdapat juga model yang tidak memerlukan asumsi-asumsi salah satunya adalah NN (Kusumadewi, 2014).

NN adalah model non parametrik yang mempunyai bentuk fungsional yang fleksibel, mengandung beberapa parameter yang tidak dapat ditafsirkan jika dalam model parametrik (Walid, dkk, 2015). NN muncul sebagai model kuantitatif yang penting untuk peramalan bisnis. NN merupakan algoritma yang serba guna untuk aplikasi peramalan dalam hal itu tidak hanya dapat untuk menemukan non linier dalam suatu masalah, tetapi NN juga dapat memodelkan proses linier (Zhang, 2004: 2). NN bekerja dengan mensimulasikan sejumlah besar unit pemrosesan sederhana yang saling berhubungan yang menyerupai versi abstrak dari neuron. Unit pemrosesan disusun berlapis-lapis. Biasanya ada tiga bagian dalam NN, yaitu lapisan *input* dengan neuron yang mewakili bidang *input*, satu atau lebih lapisan tersembunyi, dan lapisan *output* dengan satu atau lebih neuron yang mewakili bidang *output*. Neuron terhubung dengan berbagai bobot. Data *input* disampaikan ke lapisan pertama, dan nilai-nilai disebarkan dari satu neuron ke neuron lain pada lapisan berikutnya. Akhirnya, hasilnya dikirimkan dari lapisan *output* (Bahadir, 2016).

Berdasarkan kemampuan belajar yang dimilikinya, maka NN dapat dilatih untuk mempelajari dan menganalisa pola data masa lalu dan berusaha mencari suatu formula atau fungsi yang akan menghubungkan pola data masa lalu dengan *output* yang diinginkan pada saat ini (Salman & Prasetio, 2010). Salah satu kelebihan dari model NN dalam algoritma peramalan yaitu dapat digunakan untuk meramalkan data *time series* non linier (Hikmah, 2017).

Menurut Valipor, dkk (2013), dengan meningkatkan target *error* pada NN memungkinkan untuk mencegah variasi pelatihan yang parah dan tidak menguntungkan dan pada gantinya membatalkan pelatihan di dalam jaringan. Lebih banyak neuron di lapisan tersembunyi menyebabkan lebih banyak derajat kebebasan dalam jaringan. Semakin banyak variabel dioptimalkan, waktu perpanjangan pelatihan, dan matriks bobot dan vektor bias menjadi lebih tinggi. Selain itu, jumlah neuron yang lebih tinggi menghasilkan kemungkinan peningkatan penemuan jawaban dan banyak kesempatan untuk mencegah jatuh di minimum lokal.

Menurut Suhada (2009), peramalan dengan menggunakan pendekatan NN memberikan nilai MSE yang sangat kecil mendekati nol. Hal ini berarti bahwa NN memiliki akurasi yang sangat tinggi dalam melakukan peramalan terhadap suatu model sistem. Menurut Walid, dkk (2015), peramalan menggunakan NN menghasilkan hasil yang jauh lebih baik dibandingan dengan model ARIMA (*Autoregressive Integrated Moving Average*). Menurut Kohzadi, dkk (1996), peramalan dengan NN jauh lebih akurat dibandingkan dengan model tradisional ARIMA. Alasan mengapa model NN lebih baik daripada model ARIMA karena data mengandung perilaku non linier yang tidak dapat sepenuhnya ditangkap oleh model ARIMA linier. Namun, jika dibandingkan dengan algoritma peramalan tradisional seperti model ARIMA atau model regresi, ada banyak faktor pemodelan yang perlu dipertimbangkan dalam NN (Zhang & Hu, 1998).

NN dibagi menjadi dua kelompok, yaitu *Feedforward Neural Network* (FFNN) dan *Recurrent/feedback Neural Network* (RNN) (Puspitaningrum, 2006:

10). FFNN dapat juga disebut dengan *Backpropagation Neural Network* (BPNN) (Yang, dkk, 2013). Menurut Puspitaningrum (2006: 10), FFNN adalah NN yang bergerak maju dan tidak memiliki loop dimana aliran sinyalnya dari neuron *input* ke neuron *output*, sedangkan RNN adalah NN berulang atau umpan balik yang dicirikan dengan adanya loop-loop koneksi balik dimana jalur sinyal loop tertutup dari neuron kembali ke dirinya sendiri.

BPNN pada dasarnya adalah fungsi pemetaan dari neuron *input* ke neuron *output* tanpa mengetahui korelasi antar data. Setelah mempelajari *trend* data dari data historis, BPNN dapat digunakan secara efektif untuk meramalkan data baru (Ren, dkk, 2013). BPNN adalah model NN yang diakui secara kritis karena kapasitas pemodelan non parametrik dan non liniernya, kemampuan beradaptasi yang kuat, dan kemampuan komputasi paralel. Standar BPNN terdiri dari lapisan *input*, satu atau lebih lapisan tersembunyi, dan lapisan *output*. Secara umum, BPNN dengan satu lapisan tersembunyi dapat menghasilkan akurasi yang diinginkan untuk aplikasi peramalan runtun waktu (Wang, dkk, 2018). BPNN adalah suatu kelas dari FFNN dengan aturan pembelajaran terawasi. Proses pembelajaran terawasi adalah proses membandingkan setiap perkiraan jaringan dengan jawaban benar yang diketahui dan menyesuaikan berat berdasarkan *error* perkiraan yang dihasilkan untuk meminimalkan fungsi *error* (Kaastra & Boyd, 1996). Pendekatan BPNN memberikan alternatif kompetitif untuk prosedur yang ada untuk pembelajaran serta peramalan data independen (Chakraborty, dkk, 1992). Perkembangan terbaru dalam belajar NN menunjukkan bahwa FFNN adalah struktur pemetaan non linier yang dapat mendekati fungsi arbitrer apapun. Oleh karena itu, model non linier

seperti itu lebih unggul dari model tradisional ARIMA untuk peramalan *time series* (Kohzadi, dkk, 1996).

RNN memiliki karakteristik yang unik, yaitu arsitekturnya memiliki minimal satu *feedback loop*, sehingga dapat menyimpan data dalam struktur jaringannya dan performasi dari RNN dalam melakukan prediksi bergantung pada bobot dan arsitekturnya (Hardiantho, dkk, 2011). Keunikan RNN yang lain adalah adanya koneksi umpan balik yang membawa informasi gangguan (*noise*) pada saat *input* sebelumnya yang akan diakomodasikan bagi *input* berikutnya (Salman & Prasetio, 2010). Penelitian ini menggunakan RNN tipe Elman. Standar RNN tipe Elman terdiri dari empat lapisan, yaitu lapisan *input*, lapisan konteks, lapisan tersembunyi, dan lapisan *output*. Lapisan konteks membuat salinan dari lapisan tersembunyi yang dihasilkan pada langkah waktu sebelumnya, menyimpan catatan operasi jaringan sebelumnya. Jaringan ini memiliki koneksi paralel yang sangat besar, tidak hanya antara lapisan tersembunyi dan *output* tetapi juga antara lapisan tersembunyi dan *input* dan neuron konteks. Koneksi-diri membuat jaringan menjadi peka terhadap historis *input*, sehingga memungkinkannya untuk melakukan pemetaan *input* dan pola target non linier yang bervariasi waktu (Wang, dkk, 2018). Tidak seperti pada BPNN, pada RNN tipe Elman fungsi aktivasi dapat berupa sembarang fungsi, baik yang kontinu maupun yang diskontinu (Kusumadewi, 2004). RNN tipe Elman dapat menjelaskan efek berurutan dari model AR (*Autoregressive*) dan MA (*Moving Average*) secara bersamaan untuk meramalkan beberapa runtun waktu musiman dan membandingkan akurasi prakiraan dengan menggunakan model ARIMA musiman (Walid, dkk, 2015).

Penelitian yang sudah dilakukan dengan menggunakan algoritma BPNN maupun algoritma RNN untuk menyelesaikan masalah peramalan antara lain penelitian yang dilakukan oleh Kurniawan, dkk (2017) yang menggunakan FFNN *Backpropagation* untuk meramalkan harga saham, penelitian yang dilakukan oleh Misriati (2016) yang menggunakan BPNN untuk meramalkan jumlah pengunjung wisata mancanegara ke Lombok, penelitian yang dilakukan oleh Salman & Prasetio (2010) yang menggunakan RNN dengan algoritma pelatihan *gradient descent adaptive learning rate* untuk pendugaan curah hujan berdasarkan peubah ENSO (*El-Nin Southern Oscilation*), penelitian yang dilakukan oleh Pakaja, dkk (2012) yang menggunakan BPNN untuk meramalkan penjualan mobil, penelitian yang dilakukan oleh Rizal & Hartati (2017) untuk memprediksi kunjungan wisatawan di pulau Lombok dengan menggunakan RNN *Extended Kalman Filter*, dan Udin, dkk (2017) yang menggunakan BPNN dan RNN dengan algoritma pelatihan *levenberg marquardt neural network* untuk meramalkan kapasitas baterai *lead acid* pada mobil listrik.

Berdasarkan penelitian-penelitian yang telah dilakukan maka perlu adanya pengembangan metode pada NN yang dapat memberikan hasil peramalan yang lebih akurat. Oleh karena itu, akan dilakukan penelitian dengan membandingkan BPNN dengan RNN tipe Elman untuk menghitung peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk.

1.2 Rumusan Masalah

Berdasarkan latar belakang maka rumusan masalah yang akan dikaji dalam penelitian ini adalah:

1. Bagaimana pemodelan *Backpropagation Neural Network* (BPNN) untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk?
2. Bagaimana pemodelan *Recurrent Neural Network* (RNN) untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk?
3. Berapakah hasil peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk untuk sembilan periode berikutnya berdasarkan pada model terbaik?

1.3 Batasan Masalah

Batasan masalah dalam penulisan penelitian ini dimaksudkan untuk mempersempit ruang lingkup permasalahan yang akan dikaji lebih lanjut agar tidak meluas, maka pada penulisan skripsi ini, permasalahan yang akan dikaji dibatasi pada:

1. Data yang digunakan adalah data *volume* penjualan semen pada Semen Indonesia *Group* yang meliputi total penjualan di pasar domestik dan total penjualan di pasar ekspor pada Bulan Januari 2006 sampai dengan Bulan Desember 2018 yang dikeluarkan oleh *website* resmi PT Semen Indonesia (Persero) Tbk.
2. Mencari nilai *error* minimum untuk mendapatkan model terbaik dengan MSE dan MAPE.
3. Yang dimaksud dengan persoalan pada rumusan masalah adalah menemukan persamaan dari model terpilih pada *Backpropagation Neural Network* (BPNN) dan *Recurrent Neural Network* (RNN).

1.4 Tujuan

Berdasarkan rumusan masalah, maka tujuan penulisan skripsi ini adalah:

1. Untuk mengetahui pemodelan *Backpropagation Neural Network* (BPNN) untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk.
2. Untuk mengetahui pemodelan *Recurrent Neural Network* (RNN) untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk.
3. Untuk mengetahui hasil peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk untuk sembilan periode berikutnya berdasarkan model terbaik.

1.5 Manfaat

Manfaat yang diharapkan dari penyusunan skripsi ini adalah:

1. Bagi Peneliti
   1) Mengembangkan dan mengaplikasikan pengetahuan dan keilmuan di bidang matematika.
   2) Mengetahui implementasi *Backpropagation Neural Network* (BPNN) dan *Recurrent Neural Network* (RNN) untuk meramalkan data penjualan semen di PT Semen Indonesia (Persero) Tbk.
   3) Menambah dan memperkaya pengetahuan mengenai model *Backpropagation Neural Network* (BPNN) dan *Recurrent Neural Network* (RNN) serta penerapannya pada peramalan data deret berkala.
2. Bagi Mahasiswa
   1) Menambah pengetahuan mengenai model *Backpropagation Neural Network* (BPNN) dan *Recurrent Neural Network* (RNN).
   2) Memberikan suatu algoritma alternatif untuk melakukan peramalan menggunakan model *Backpropagation Neural Network* (BPNN) dan *Recurrent Neural Network* (RNN).

3. Bagi Jurusan
    1) Menambah khasanah untuk keilmuan dibidang statistik, khususnya *Neural Network* (NN).
    2) Menambah kajian pustaka bagi pihak perpustakaan sebagai bahan bacaan bagi mahasiswa yang membutuhkan dalam memahami lebih lanjut mengenai implementasi *Neural Network* (NN) untuk menyelesaikan masalah peramalan.
4. Bagi Perusahaan

    Hasil penelitian ini dapat digunakan sebagai salah satu alternatif pengambilan keputusan bagi perusahaan dalam meramalkan penjualan sehingga ketika penjualan diprediksi dengan akurat maka pemenuhan permintaan konsumen dapat diusahakan tepat waktu, kerjasama perusahaan dengan relasi tetap terjaga dengan baik, kepuasan konsumen terpenuhi, perusahaan dapat mengatasi hilangnya penjualan atau kehabisan stok, mencegah pelanggan lari ke kompetitor.

1.6 Sistematika Penulisan

Secara garis besar, penulisan skripsi ini terdiri dari tiga bagian utama, yaitu bagian awal, bagian isi, dan bagian akhir. Untuk memberikan gambaran yang jelas tentang skripsi ini dan memudahkan pembaca dalam menelaah skripsi ini, maka skripsi ini disusun secara sistematis sebagai berikut.

1. Bagian awal terdiri atas halaman judul, pernyataan keaslian tulisan, halaman pengesahan, motto dan persembahan, kata pengantar, abstrak, daftar isi, daftar tabel, daftar gambar, dan daftar lampiran.

2. Bagian isi terdiri atas judul bab dan bagian-bagiannya. Bagian isi terdiri atas lima bab, yaitu:

   A. Bab 1 Pendahuluan

      Bab pendahuluan menyajikan gagasam pokok yang terdiri atas latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

   B. Bab 2 Tinjauan Pustaka

      Bab tinjauan pustaka berisi kajian teori yang mendasari pemecahan dari permasalahan yang disajikan. Tinjauan pustaka ini berisi tentang konsep dasar *neural network, backpropagation neural network, recurrent neural network, mean square error* (MSE), *mean percentage error* (MAPE), Matlab, penjualan, peramalan, peramalan penjualan, penelitian yang relevan, dan kerangka berpikir.

   C. Bab 3 Algoritma Penelitian

      Bab algoritma penelitian menyajikan gagasan pokok tentang prosedur dan langkah-langkah yang dilakukan dalam penelitian. Bab ini terdiri atas studi pustaka, pengumpulan data, spesifikasi perangkat, pemecahan masalah, dan penarikan kesimpulan.

   D. Bab 4 Hasil Penelitian dan Pembahasan

      Bab hasil penelitian dan pembahasan berisi hasil analisis data dan pembahasan dari hasil penelitian yang telah dilakukan.

E. Bab 5 Penutup

Bab penutup berisi tentang simpulan dari hasil pembahasan dan saran untuk pengembangan penelitian selanjutnya.

3. Bagian Akhir

Bagian akhir skripsi terdiri atas daftar pustaka dan lampiran. Daftar pustaka berisi semua bahan kepustakaan yang digunakan sebagai rujukan langsung dalam penulisan skripsi. Lampiran berisi kelengkapan skripsi.

BAB II

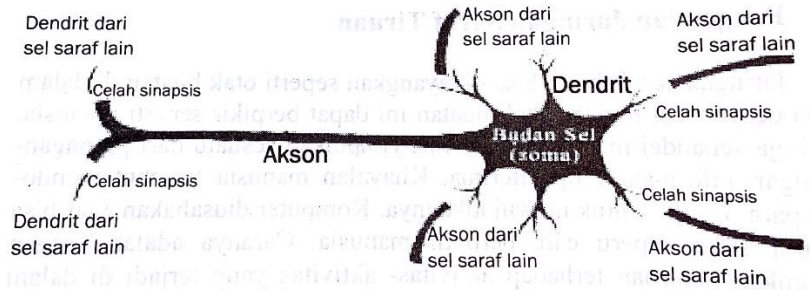
TINJAUAN PUSTAKA

2.1 *Neural Network*

2.1.1 Jaringan Syaraf Biologi

Menurut Puspitaningrum (2006: 2-3), jaringan syaraf biologi merupakan kumpulan dari sel-sel syaraf (neuron). Neuron mempunyai tugas mengolah informasi. Komponen-komponen utama dari sebuah neuron dapat dikelompokkan menjadi 3 bagian, yaitu:

1. Dendrit, bertugas untuk menerima informasi.
2. Badan sel (soma), berfungsi sebagai tempat pengolahan informasi.
3. Akson (neurit), mengirimkan impuls-impuls ke sel syaraf lainnya.



(Sumber: Puspitaningrum, 2006: 2)

Gambar 2.1. Sel Syaraf Biologi

Perhatikan Gambar 2.1. Sebuah neuron menerima impuls-impuls sinyal dari neuron yang lain melalui dendrit dan mengirimkan sinyal yang dibagikan oleh badan sel melalui akson. Akson dari sel syaraf biologi ini bercabang-cabang dan berhubungan dengan dendrit dari sel syaraf lainnya dengan cara mengirimkan

impuls melalui sinapsis. Sinapsis adalah neuron fungsional antara dua buah sel syaraf, katakanlah A dan B, dimana yang satu adalah serabut akson dari neuron A dan satunya lagi dendrit dari neuron B. Kekuatan sinapsis ini bisa menurun atau meningkat tergantung kepada seberapa besar tingkat propagasi (penyiaran) sinyal yang diterimanya.

Adapun cara belajar *neural network* yaitu ke dalam *neural network* di-*input*-kan informasi yang sebelumnya telah diketahui hasil *output*-nya. *Input* informasi ini dilakukan lewat node-node atau neuron-neuron *input*. Bobot-bobot antar penghubung dalam suatu arsitektur diberi nilai awal dan kemudian *neural network* dijalankan. Bobot-bobot ini bagi jaringan digunakan untuk belajar dan mengingat suatu informasi. Pengaturan bobot dilakukan secara terus-menerus dan dengan menggunakan kriteria tertentu sampai diperoleh *output* yang diharapkan. Tabel 2.1 memperlihatkan keanalogan antara *neural network* dengan jaringan syaraf biologi.

Tabel 2.1. Keanalogan *Neural Network* Terhadap Jaringan Syaraf Biologi

Neural Network	**Jaringan Syaraf Biologi**
Node atau neuron	Badan sel (soma)
Input	Dendrit
Output	Akson
Bobot	Sinapsis

Hal yang ingin dicapai dengan melatih atau mengajari *neural network* adalah untuk mencapai keseimbangan antara kemampuan memorisasi dan generalisasi. Kemampuan memorisasi adalah kemampuan *neural network* untuk

memanggil kembali secara sempurna sebuah pola yang telah dipelajari, sedangkan kemampuan generalisasi adalah kemampuan *neural network* untuk menghasilkan respon yang bisa diterima terhadap pola-pola *input* yang serupa (namun tidak identik) dengan pola-pola yang sebelumnya telah dipelajari. Hal ini sangat bermanfaat bila pada suatu saat ke dalam *neural network* itu di-*input*-kan informasi baru yang belum pernah dipelajari, maka *neural network* itu masih akan tetap dapat memberikan tanggapan yang baik, memberikan *output* yang paling mendekati.

2.1.2 Sejarah *Neural Network*

Sejarah perkembangan *neural network* secara garis besar telah dimulai pada tahun 1940 dengan mengasosiasikan cara kerja otak manusia dengan logika numerik yang diadaptasi peralatan komputer. Perkembangan selanjutnya mengalami banyak tahapan. Tahun 1943, McCulloch dan Pitts meletakan konsep dasar *neural network* secara matematis, model matematis lebih mudah dipahami, diimplementasikan dan dikembangkan, sedangkan di tahun 1949, Donald Hebb memperkenalkan metode pembelajaran Hebbian. Metode pembelajaran memungkinkan informasi keadaan disimpan di antara *neural network* untuk proses selanjutnya. Tahun 1952, Ashby dalam buku *the origin of adaptive behavior* memperkenalkan ide pembelajaran adaptif. Komputerisasi dengan menggunakan teknik *neural network* pertama kali dibuat dan di uji coba pada tahun 1954. Tahun 1958, Frank Rosenblatt memperkenalkan struktur *perceptron neural network*. Struktur ini merupakan dasar dari mesin *neural network* yang dikembangkan hingga sekarang. Tahun 1960, mesin ADALINE *neural network* dibuat dan dilatih dengan metode pembelajaran *Least Mean Square* oleh Widrow dan Hoff.

ADALINE diaplikasikan untuk peramalan cuaca, pencocokan pola, dan kendali adaptif (Muis, 2006: 1-2).

Tahun 1974, Werbos memperkenalkan algoritma *backpropagation* untuk melatih *perceptron* dengan banyak lapisan. Kohonen mengembangkan metode pembelajaran *neural network* yang tidak terawasi (*unsupervised learning*) pada tahun 1982 untuk pemetaan. Tahun 1986, Rumelhart menciptakan algoritma belajar yang dikenal sebagai perambatan balik. Algoritma ini diterapkan pada *perceptron* yang memiliki banyak lapisan (*multilayer perceptron*) maka dapat dibuktikan bahwa pemilahan pola-pola yang tidak linear dapat diselesaikan. Tahun 1988 merupakan tahun dimana *radial basis function* mulai dikembangkan (Hermawan, 2006: 19).

2.1.3 Definisi *Neural Network*

Menurut Fausett (1994: 3), Jaringan Syaraf Tiruan (JST) atau yang sering dikenal dengan *Neural Network* (NN) adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan syaraf biologi. NN dibentuk sebagai generalisasi model matematika dari jaringan syaraf biologi, dengan asumsi sebagai berikut.

1. Pemrosesan informasi terjadi pada banyak elemen sederhana yang disebut neuron.
2. Sinyal dikirimkan diantara neuron-neuron melalui penghubung-penghubung.
3. Setiap penghubung antar neuron memiliki bobot yang akan memperkuat atau memperlemah sinyal (bobot yang bernilai positif akan memperkuat sinyal, sedangkan bobot yang bernilai negatif akan memperlemah sinyal).

4. Setiap neuron menggunakan fungsi aktivasi (biasanya bukan fungsi linier) terhadap *input* (jumlah sinyal *input* yang terboboti) untuk menentukan sinyal *output*.

Karakteristik NN ditentukan oleh pola penghubung antara neuron-neuron (yang disebut arsitektur), metode penentuan bobot pada penghubung (yang disebut pelatihan, atau pembelajaran, algoritma), dan fungsi aktivasi. NN terdiri dari sejumlah besar elemen pemrosesan sederhana yang disebut neuron, unit, sel, atau node. Setiap neuron terhubung ke neuron lain melalui penghubung komunikasi langsung, masing-masing dengan bobot terkait. Bobot menunjukkan informasi yang digunakan oleh jaringan untuk memecahkan masalah. Setiap neuron memiliki keadaan internal, yang disebut aktivasi atau aktivitasnya, yang merupakan fungsi dari *input* yang telah diterimanya. Biasanya neuron mengirimkan aktivasi sebagai sinyal ke beberapa neuron lain. Penting untuk dicatat bahwa neuron hanya dapat mengirim satu sinyal pada satu waktu, meskipun sinyal itu disiarkan ke beberapa neuron lain.

Menurut Siswanto (2010: 175), NN adalah sistem pengolahan informasi yang didasari filosofi struktur perilaku syaraf makhluk hidup. NN mempunyai kelebihan memecahkan masalah teknis. Pertama, NN tak perlu pemrograman tentang hubungan *input* dan *output*. Melainkan akan mempelajari sendiri respon yang diinginkan dengan cara pelatihan. Ini sangat penting guna menghilangkan sebagian besar biaya pemrograman. Kedua, NN dapat memperbaiki respon dengan belajar. Itu karena NN didesain untuk mengevaluasi dan beradaptasi terhadap kriteria-kriteria respon yang baru. Ketiga, NN bekerja sebagai penjumlahan semua

sinyal *input* dengan *input* tidak harus sama. Ini artinya NN akan dapat mengenali seseorang meski orang tersebut sudah berbeda dengan saat dikenali pertama kali atau NN akan mengenali suatu kata meski kata itu diucapkan oleh orang yang berbeda-beda. Semua ini tentunya sangat sulit dikerjakan oleh teknik komputer digital biasa.

Menurut Kusumadewi (2004: 49), NN merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena NN ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pelatihan.

Menurut Puspitaningrum (2006: 1), NN bisa dibayangkan seperti otak buatan di dalam cerita-cerita fiksi ilmiah. Otak buatan ini dapat berpikir seperti manusia, dan juga sepandai manusia dalam menyimpulkan sesuatu dari potongan-potongan informasi yang diterima.

Menurut Hermawan (2006: 2), NN merupakan salah satu sistem pemrosesan informasi yang didesain dengan menirukan cara kerja otak manusia dalam menyelesaikan suatu masalah dengan melakukan proses belajar melalui perubahan bobot sinapsisnya. NN mampu mengenali kegiatan dengan berbasis pada data masa lalu. Data masa lalu akan dipelajari oleh NN sehingga mempunyai kemampuan untuk memberi keputusan terhadap data yang belum pernah dipelajari.

2.1.4 Arsitektur Jaringan

Pembagian arsitektur NN bisa dilihat dari kerangka kerja dan skema interkoneksi. Kerangka kerja NN bisa dilihat dari jumlah lapisan (*layer*) dan jumlah

neuron pada setiap lapisan. Lapisan-lapisan penyusun NN dapat dibagi menjadi tiga, yaitu (Puspitaningrum, 2006: 9):

1. Lapisan masukan (*input layer*)

   Node-node di dalam lapisan *input* disebut neuron-neuron *input*. Neuron-neuron *input* menerima *input* dari dunia luar. *Input* yang dimasukkan merupakan penggambaran dari suatu masalah.

2. Lapisan tersembunyi (*hidden layer*)

   Node-node di dalam lapisan tersembunyi disebut neuron-neuron tersembunyi. *Output* dari lapisan ini tidak secara langsung dapat diamati.

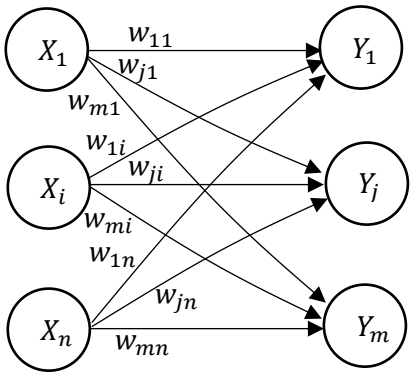
3. Lapisan keluaran (*output layer*)

   Node-node pada lapisan *output* disebut neuron-neuron *output*. *Output* dari lapisan ini merupakan *output* NN terhadap suatu permasalahan.

Menurut Fausett (1994: 12), pengaturan neuron ke dalam lapisan dan pola koneksi di dalam dan di antara lapisan disebut arsitektur jaringan. Banyak NN memiliki lapisan *input* dimana aktivasi setiap neuron sama dengan sinyal *input* eksternal. Beberapa arsitektur jaringan yang sering dipakai dalam NN, antara lain (Siang, 2004: 24):

1. Jaringan lapisan tunggal (*single layer network*)

   Sekumpulan *input* neuron dihubungkan langsung dengan sekumpulan *output*-nya pada jaringan ini. Beberapa model hanya ada sebuah neuron pada lapisan *output*.

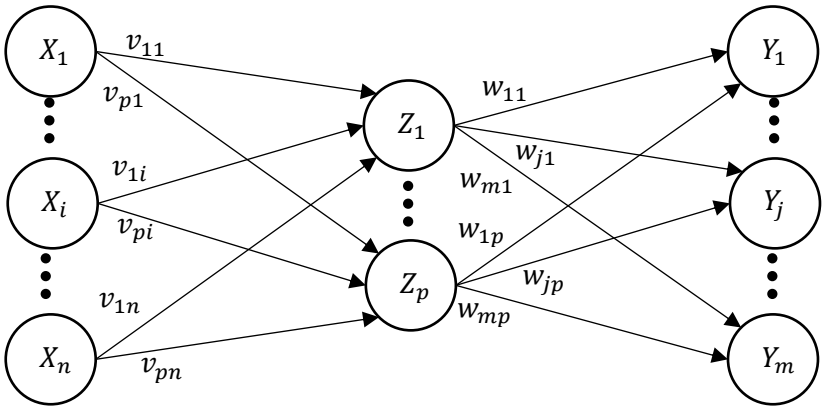


Gambar 2.2. Jaringan Lapisan Tunggal

Gambar 2.2 menunjukkan arsitektur jaringan dengan n neuron *input* $(X_1, X_2, \dots, X_n)$ dan m buah neuron *output* $(Y_1, Y_2, \dots, Y_m)$. Perhatikan bahwa dalam jaringan ini semua neuron *input* dihubungkan dengan semua neuron *output*, meskipun dengan bobot yang berbeda-beda. Tidak ada neuron *input* yang dihubungkan dengan neuron *input* lainnya, demikian pula dengan neuron *output*. Besaran w_{ji} menyatakan bobot hubungan antara neuron ke-i dalam *input* dengan neuron ke-j dalam *output*. Bobot-bobot ini saling independen. Selama proses pelatihan, bobot-bobot tersebut akan dimodifikasi untuk meningkatkan keakuratan hasil.

2. Jaringan lapisan jamak (*multi-layer network*)

   Jaringan lapisan jamak merupakan perluasan dari lapisan tunggal. Dalam jaringan ini, selain neuron *input* dan *output*, ada neuron-neuron lain (sering disebut lapisan tersembunyi). Dimungkinkan pula ada beberapa lapisan tersembunyi. Sama seperti pada neuron *input* dan *output*, neuron-neuron dalam satu lapisan tidak saling berhubungan.



Gambar 2.3. Jaringan Lapisan Jamak

Gambar 2.3 adalah jaringan dengan n buah *input* $(X_1, X_2, \dots, X_n)$, sebuah lapisan tersembunyi yang terdiri dari p buah neuron $(Z_1, \dots, Z_p)$, dan m buah neuron *output* $(Y_1, Y_2, \dots, Y_m)$. Jaringan lapisan jamak dapat menyelesaikan masalah yang lebih kompleks dibandingkan dengan lapisan tunggal, meskipun kadangkala proses pelatihan lebih kompleks dan lama.

3. Jaringan *reccurent*

   Model jaringan *reccurent* mirip dengan jaringan lapisan tunggal atau ganda. Hanya saja, ada neuron *output* yang memberikan sinyal pada neuron *input* (sering disebut *feedback loop*).

2.1.5 Algoritma Pelatihan *Neural Network*

Salah satu bagian terpenting dari konsep NN adalah terjadinya proses pelatihan (*training*) atau pembelajaran (*learning*). Tujuan utama dari proses pelatihan adalah melakukan pengaturan terhadap bobot-bobot yang ada pada NN, sehingga diperoleh bobot akhir yang tepat sesuai dengan pola data yang dilatih. Selama proses pelatihan akan terjadi perbaikan bobot-bobot berdasarkan algoritma tertentu. Nilai bobot akan bertambah jika informasi yang diberikan oleh neuron yang bersangkutan tersampaikan, sebaliknya jika informasi tidak disampaikan oleh

suatu neuron ke neuron yang lain, maka nilai bobot yang menghubungkan keduanya akan dikurangi. Saat pelatihan dilakukan pada *input* yang berbeda, maka nilai bobot akan diubah secara dinamis hingga mencapai suatu nilai yang cukup seimbang. Apabila nilai ini telah tercapai mengindikasikan bahwa tiap-tiap *input* telah berhubungan dengan *output* yang diharapkan (Kusumadewi & Hartati, 2010: 84). Proses pelatihan berlangsung pada beberapa iterasi (dalam NN dikenal sebagai *epoch*) dan berhenti setelah jaringan menemukan bobot yang sesuai dimana nilai *error* yang diinginkan telah tercapai atau jumlah iterasi telah mencapai nilai maksimal yang ditetapkan sebelumnya. Berdasarkan permasalahan yang ditangani, proses pelatihan pada NN dapat dibedakan menjadi dua, yaitu (Warsito, 2009: 4-5):

1. Pelatihan terawasi (*supervised training*)

   Metode pelatihan disebut terawasi jika target yang diharapkan dari *input* yang diterima jaringan telah diketahui sebelumnya. Metode pelatihan ini mengarahkan jaringan untuk dapat mengenali pola *input*, dengan cara memetakan pola *input* ke pola *output* yang diinginkan. Tujuan dari pelatihan terawasi adalah untuk menghasilkan pemetaan dari vektor *input* ke nilai *output*, sehingga akan diperoleh *output* yang sesuai dengan targetnya. Pengubahan nilai bobot (proses pelatihan) dilakukan untuk mencapai tujuan ini. Setiap *input* jaringan akan menghitung nilai *output*-nya dan selanjutnya *output* tersebut akan dibandingkan dengan vektor target yang bersesuaian. Perbedaan antara *output* yang dihasilkan dengan target *output* disebut sebagai *error*. *Error* tersebut akan dikembalikan ke dalam jaringan sebagai acuan untuk menentukan bobot

koneksi yang baru. Bobot koneksi antar neuron akan diperbaiki berdasarkan algoritma pelatihan tertentu, sampai seluruh rangkaian pelatihan menghasilkan *error* yang sedikit (minimal).

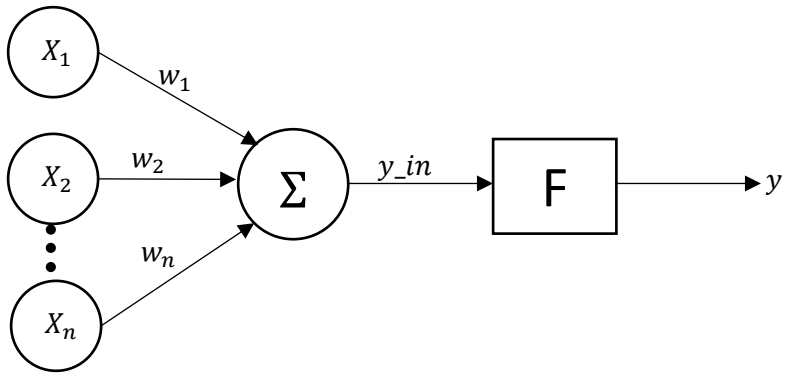
2. Pelatihan tak terawasi (*unsupervised learning*)

   Pelatihan tak terawasi diterapkan pada permasalahan dimana *output* jaringan tidak diketahui sebelumnya. Pengubahan nilai bobot pada metode ini dilakukan dengan sendirinya tanpa menggunakan *output* yang diinginkan. Tujuannya adalah mengelompokkan neuron-neuron dengan ciri-ciri yang hampir sama ke dalam satu kelompok, dengan memanfaatkan sifat keteraturan dalam himpunan data. Pelatihan tak terawasi tidak membutuhkan vektor target untuk *output*-nya, sehingga tidak ada pembandingan yang dilakukan dengan target yang ditentukan sebelumnya. Rangkaian pelatihan hanya berisi vektor *input* saja. Jaringan akan menempatkan *input* ke dalam beberapa kelompok. Selanjutnya, dari kelompok-kelompok tersebut ditentukan karakteristik kelompoknya, sehingga apabila jaringan menerima *input* yang baru dapat ditentukan ke dalam kelompok dimana *input* tersebut berada. Jaringan memperbaiki bobot koneksi antar neuron sedemikian sehingga pola-pola *input* yang mirip ditempatkan ke dalam kelompok yang sama. Kemudian jaringan akan membuat pola-pola *input* sebagai wakil bagi setiap kelompok yang terbentuk.

2.1.6 Fungsi Aktivasi

Menurut Siang (2004: 26), dalam NN, fungsi aktivasi dipakai untuk menentukan *output* suatu neuron. Menurut Puspitaningrum (2006: 14), fungsi aktivasi adalah fungsi yang menggambarkan hubungan antara tingkat aktivasi

internal (*summation function*) yang mungkin berbentuk *linear* atau non-*linear*. Gambar 2.8 menunjukkan NN sederhana dengan fungsi aktivasi F (Kusumadewi & Hartati, 2010: 72-73).



Gambar 2.4. Fungsi Aktivasi pada *Neural Network* Sederhana

Gambar 2.4 menunjukkan sebuah neuron akan mengolah n *input* $(X_1, X_2, \dots, X_n)$ yang masing-masing memiliki bobot $w_1, w_2, \dots, w_n$, dengan rumus:

$$y\_in = \sum_{i=1}^{n} x_i w_i \tag{2.1}$$

Kemudian fungsi aktivasi F akan mengaktivasi y_in menjadi *output* jaringan y. Untuk NN dengan jumlah neuron pada lapisan *output* sebanyak m buah, maka proses pengolahan data pada neuron ke-j adalah:

$$y\_in_j = \sum_{i=1}^{N} x_i w_{ij}; j = 1, \dots, m \tag{2.2}$$

dengan w_{ij} adalah bobot yang menghubungkan *input* ke-i menuju neuron ke-j.

Adakalanya NN tidak mampu mengakomodasi informasi yang ada melalui data-data *input* maupun melalui bobot-bobotnya. Untuk mengakomodasi hal tersebut, maka biasanya pada NN ditambahkan bias yang senantiasa bernilai 1 (Gambar 2.5). Bias berfungsi untuk mengubah nilai nilai ambang (*threshold*)

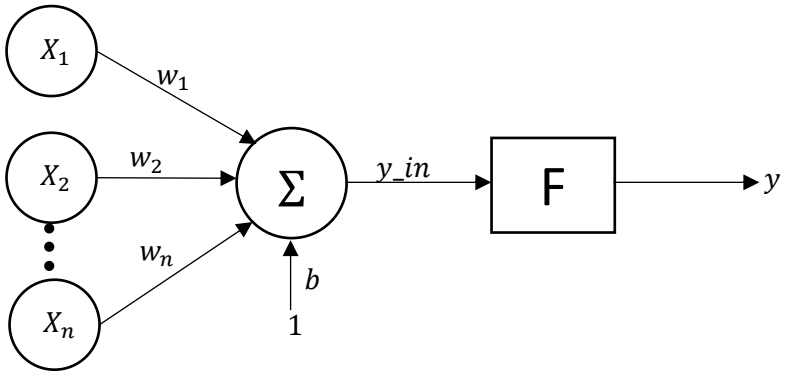
menjadi sama dengan 0, sedangkan *threshold* berperan sebagai penimbang dalam suatu hubungan dari sebuah neuron tertentu. Pengaruh bias terhadap neuron ditunjukkan dengan bobot bias, b. Apabila pada NN dilengkapi dengan bias, maka proses komputasi neuron menjadi:

$$y\_in = \sum_{i=1}^{N} x_i w_i + b \tag{2.3}$$

Untuk NN dengan jumlah neuron pada lapisan *output* sebanyak m buah, maka proses pengolahan data pada neuron ke-j adalah:

$$y\_in_j = \sum_{i=1}^{N} x_i w_{ij} + b_j; j = 1, \dots, m \tag{2.4}$$

dengan w_{ij} adalah bobot yang menghubungkan *input* ke-i menuju neuron ke-j, dan b_j adalah bobot bias yang menuju ke neuron ke-j.



Gambar 2.5. Fungsi Aktivasi pada *Neural Network* Sederhana dengan Bias

Ada beberapa fungsi aktivasi yang sering digunakan dalam NN, antara lain (Kusumadewi, 2004: 51-62):

1. Fungsi undak biner (*hard limit*)

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

   Fungsi undak biner (*hard limit*) dirumuskan sebagai:

$$y = \begin{cases} 0, \text{ jika } x < 0 \\ 1, \text{ jika } x \geq 0 \end{cases} \tag{2.5}$$

   Fungsi aktivasi undak biner pada Matlab dikenal dengan nama `hardlim`. *Syntax* untuk fungsi tersebut adalah:

```
Y = hardlim(x)
```

2. Fungsi bipolar (*symmetric hard limit*)

   Fungsi bipolar sebenarnya hampir sama dengan fungsi undak biner, hanya saja *output* yang dihasilkan berupa 1, 0, atau -1.

   Fungsi *symetric hard limit* dirumuskan sebagai:

$$y = \begin{cases} 1, \text{ jika } x \geq 0 \\ -1, \text{ jika } x < 0 \end{cases} \tag{2.6}$$

   Fungsi aktivasi bipolar pada Matlab dikenal dengan nama `hardlims`. *Syntax* untuk fungsi tersebut adalah:

```
Y = hardlims(x)
```

3. Fungsi *linear* (identitas)

   Fungsi linear memiliki *output* yang sama dengan nilai *input*-nya.

   Fungsi *linear* dirumuskan sebagai:

$$y = x \tag{2.7}$$

Fungsi aktivasi *linear* pada Matlab dikenal dengan nama `purelin`. *Syntax* untuk fungsi tersebut adalah:

```
Y = purelin(x)
```

4. Fungsi *saturating linear*

   Fungsi ini akan bernilai 0 jika *input*-nya kurang dari $-\frac{1}{2}$, dan akan bernilai 1 jika *input*-nya lebih dari $\frac{1}{2}$. Sedangkan jika nilai *input* terletak antara $-\frac{1}{2}$ dan $\frac{1}{2}$, maka *output*-nya akan bernilai sama dengan nilai *input* ditambah $\frac{1}{2}$.

   Fungsi *saturating linear* dirumuskan sebagai:

$$y = \begin{cases} 1; & \text{jika } x \geq 0{,}5 \\ x + 0{,}5; & \text{jika } -0{,}5 \leq x \leq 0{,}5 \\ 0; & \text{jika } x \leq -0{,}5 \end{cases} \tag{2.8}$$

   Fungsi aktivasi identitas pada Matlab dikenal dengan nama `satlin`. *Syntax* untuk fungsi tersebut adalah:

```
Y = satlin(x)
```

5. Fungsi *symmetric saturating linear*

   Fungsi ini akan bernilai -1 jika *input*-nya kurang dari -1, dan akan bernilai 1 jika *input*-nya lebih dari 1. Sedangkan jika nilai *input* terletak antara -1 dan 1, maka *output*-nya akan bernilai sama dengan nilai *input*-nya.

   Fungsi *symetric saturating linear* dirumuskan sebagai:

$$y = \begin{cases} 1; & \text{jika } x \geq 1 \\ x; & \text{jika } -1 \leq x \leq 1 \\ -1; & \text{jika } x \leq -1 \end{cases} \tag{2.9}$$

   Fungsi aktivasi *symmetric saturating linear* pada Matlab dikenal dengan nama `satlins`. *Syntax* untuk fungsi tersebut adalah:

```
Y = satlins(x)
```

6. Fungsi sigmoid biner

   Fungsi ini digunakan untuk NN yang dilatih dengan menggunakan metode *backpropagation*. Fungsi sigmoid biner memiliki nilai pada *range* 0 sampai 1. Oleh karena itu, fungsi ini sering digunakan untuk NN yang membutuhkan nilai *output* yang terletak pada interval 0 sampai 1. Namun, fungsi ini bisa juga digunakan oleh NN yang nilai *output*-nya 0 atau 1.

   Fungsi sigmoid biner dirumuskan sebagai:

$$y = f(x) = \frac{1}{1 + e^{-\sigma x}} \tag{2.10}$$

   dengan: $f'(x) = \sigma f(x)[1 - f(x)]$.

   Fungsi aktivasi sigmoid biner pada Matlab dikenal dengan nama `logsig`. *Syntax* untuk fungsi tersebut adalah:

```
Y = logsig(x)
```

7. Fungsi sigmoid bipolar

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

   Fungsi sigmoid bipolar bipolar dirumuskan sebagai:

$$y = f(x) = \frac{1 - e^{-\sigma x}}{1 + e^{-\sigma x}} \tag{2.11}$$

   dengan: $f'(x) = \frac{\sigma}{2}[1 + f(x)][1 - f(x)]$.

   Fungsi ini sangat dekat dengan fungsi *hyperbolic tangent*. Keduanya memiliki *range* antara -1 sampai 1. Untuk fungsi *hyperbolic tagent*, dirumuskan sebagai:

$$y = f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.12)$$

Atau

$$y = f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad (2.13)$$

dengan: $f'(x) = [1 + f(x)][1 - f(x)]$.

Fungsi aktivasi sigmoid bipolar pada Matlab dikenal dengan nama `tansig`. *Syntax* untuk fungsi tersebut adalah:

```
Y = tansig(x)
```

2.2 *Backpropagation Neural Network*

2.2.1 Konsep Dasar *Backpropagation Neural Network*

Menurut Fausset (1994: 12), *Feedforward Neural Network* (FFNN) merupakan jaringan dimana sinyal mengalir dari neuron *input* ke neuron *output*, dalam arah maju. Algoritma pelatihan *Backpropagation* (BP) atau propagasi balik, pertama kali dirumuskan oleh Werbos dan dipopulerkan oleh Rumelhart bersama McClelland untuk digunakan pada NN. Algoritma ini didesain untuk operasi pada NN *feedforward* lapisan jamak. Proses pelatihan pada BP didasarkan pada interkoneksi yang sederhana, yaitu jika *output* memberikan hasil yang salah, maka bobot dikoreksi supaya *error*-nya dapat diperkecil dan tanggapan NN selanjutnya diharapkan akan lebih mendekati nilai yang benar. BP berkemampuan untuk memperbaiki bobot pada lapisan tersembunyi (Purnomo & Kurniawan, 2006: 32-33).

Model FFNN dengan algoritma *Backpropagation Neural Network* (BPNN) untuk melatih jaringan sehingga diperoleh bobot-bobot optimal yang meminimalkan *error*. Istilah BPNN berhubungan dengan metode untuk

penghitungan gradien fungsi *error* dalam kaitannya dengan bobot-bobot untuk suatu jaringan umpan maju (*feedforward*). Pelatihan menggunakan BPNN meliputi tiga tahap yaitu *feedforward* dari pola *input*, penghitungan *error*, dan penyesuaian bobot-bobot (Warsito, 2009: 37).

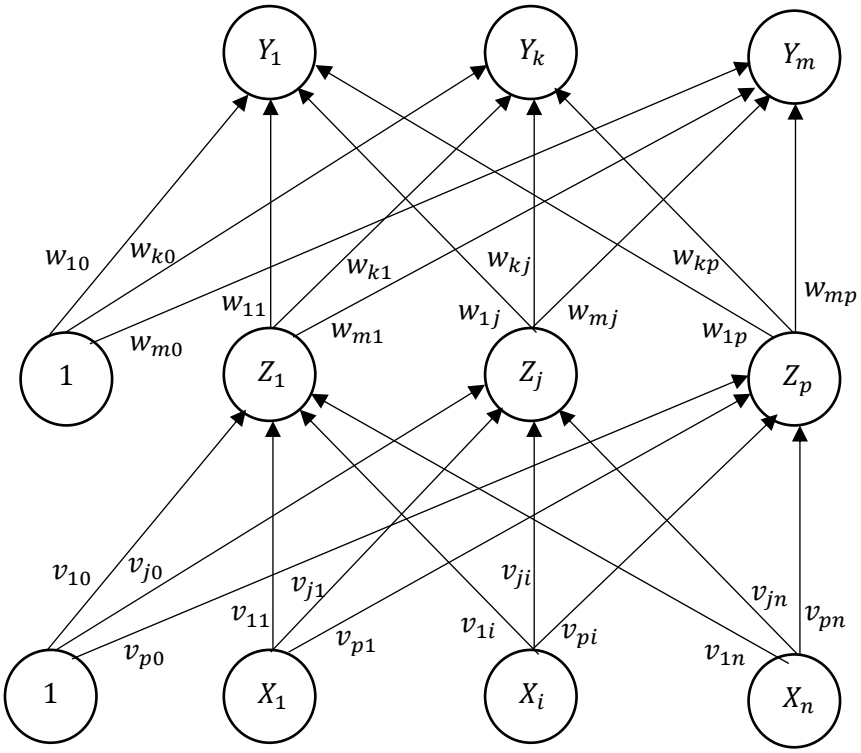
Istilah BP atau propagasi balik atau penyiaran kembali diambil dari cara kerja jaringan ini, yaitu bahwa gradien *error* neuron-neuron tersembunyi diturunkan dari penyiaran kembali *error* yang disosiasikan dengan neuron-neuron *output*. Hal ini karena nilai target untuk neuron-neuron tersembunyi tidak diberikan. Algoritma ini menurunkan gradien untuk meminimkan penjumlahan *error* kuadrat *output* jaringan (Puspitaningrum, 2006: 125).

BPNN merupakan algoritma pelatihan yang terawasi dan biasanya digunakan oleh *perceptron* dengan lapisan jamak untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyinya. BPNN menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error* ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu (Kusumadewi, 2004: 93). BPNN melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respon yang benar terhadap pola *input* yang serupa (tapi tidak sama) dengan pola yang dipakai selama pelatihan (Siang, 2004: 97).

2.2.2 Arsitektur *Backpropagation Neural Network*

BPNN memiliki beberapa neuron yang ada dalam satu atau lebih lapisan tersembunyi. Gambar 2.6 adalah arsitektur BPNN dengan *n input* (ditambah sebuah

bias), sebuah lapisan tersembunyi yang terdiri dari p neuron (ditambah sebuah bias), serta m buah neuron *output*. v_{ji} merupakan bobot garis dari neuron *input* X_i ke neuron lapisan tersembunyi Z_j (v_{j0} merupakan bobot garis yang menghubungkan bias di neuron *input* ke neuron lapisan tersembunyi Z_j). w_{kj} merupakan bobot dari neuron lapisan tersembunyi Z_j ke neuron *output* Y_k (w_{k0} merupakan bobot dari bias di lapisan tersembunyi ke neuron *output* Y_k) (Siang, 2004: 98).



Gambar 2.6. Arsitektur *Backpropagation Neural Network*

Selama *feedforward*, setiap neuron *input* (X_i) menerima sinyal *input* dan mengirimkannya ke setiap neuron tersembunyi ($Z_1, \dots, Z_p$). Setiap neuron tersembunyi menghitung aktivasi dan mengirimkan sinyal (z_j) ke setiap neuron *output*. Setiap neuron *output* (Y_k) menghitung aktivasi (y_k) untuk membentuk respons jaringan untuk pola *input* yang diberikan.

Selama pelatihan, setiap neuron *output* membandingkan perhitungan aktivasi y_k dengan nilai targetnya t_k untuk menentukan *error* terkait untuk pola tersebut dengan neuron itu. Berdasarkan *error* ini, faktor δ_k ($k = 1, \dots, m$) dihitung. δ_k digunakan untuk menyalurkan *error* pada neuron *output* Y_k kembali ke semua neuron di lapisan sebelumnya (neuron tersembunyi terhubung ke Y_k). Itu juga digunakan (nanti) untuk memperbaiki bobot diantara lapisan *output* dan lapisan tersembunyi. Dengan cara yang sama, faktor δ_j ($j = 1, \dots, p$) dihitung untuk setiap neuron tersembunyi Z_j. Itu tidak perlu untuk menyebarkan *error* kembali ke lapisan *input*, tetapi δ_j digunakan untuk memperbaiki bobot-bobot diantara lapisan tersembunyi dan lapisan *input*.

Setelah semua faktor δ telah ditentukan, bobot-bobot untuk semua lapisan disesuaikan secara bersamaan. Penyesuaian bobot w_{kj} (dari neuron tersembunyi Z_j ke neuron *output* Y_k) didasarkan pada faktor δ_k dan aktivasi z_j dari neuron tersembunyi Z_j. Penyesuaian bobot v_{ji} (dari neuron *input* X_i ke neuron tersembunyi Z_j) didasarkan pada faktor δ_j dan aktivasi x_i dari neuron *input* X_i (Fausett, 1994: 291).

2.2.3 Fungsi Aktivasi *Backpropagation Neural Network*

Fungsi aktivasi untuk BPNN harus memiliki beberapa karakteristik penting, yaitu: harus kontinu, terdiferensiasi, dan monoton tidak menurun. Selanjutnya, untuk efisiensi komputasi, diharapkan turunannya mudah dikomputasi. Untuk fungsi aktivasi yang paling banyak digunakan, nilai dari turunan (pada nilai tertentu dari variabel dependen) dapat dinyatakan dalam bentuk nilai fungsi (pada nilai dari variabel independen). Salah satu fungsi aktivasi yang paling umum adalah fungsi

simoid biner yang memiliki *range* $(0,1)$ dan disefinisikan sebagai (Fausett, 1994: 292-293):

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.14}$$

dengan: $f'(x) = f(x)[1 - f(x)]$.

Fungsi aktivasi yang umum lainnya adalah sigmoid bipolar yang memiliki *range* $(-1,1)$ dan didefinisikan sebagai:

$$f(x) = \frac{2}{1 + e^{-x}} - 1 \tag{2.15}$$

dengan: $f'(x) = \frac{1}{2}[1 + f(x)][1 - f(x)]$.

Catatan bahwa fungsi sigmoid bipolar terkait erat dengan fungsi

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.16}$$

Fungsi sigmoid memiliki nilai maksimum sama dengan 1. Maka untuk pola targetnya lebih dari 1, pola *input* dan *output* harus terlebih dahulu ditransformasi sehingga semua polanya memiliki *range* yang sama seperti sigmoid yang dipakai. Alternatif lain adalah menggunakan fungsi aktivasi sigmoid hanya pada lapisan yang bukan lapisan *output*. Fungsi aktivasi yang dipakai pada lapisan *output* adalah fungsi identitas dan didefinisikan sebagai (Siang, 2004: 99-100):

$$f(x) = x \tag{2.17}$$

2.2.4 Algoritma Pelatihan *Backpropagation Neural Network*

Salah satu fungsi aktivasi yang didefinisikan di bagian 2.2.3 dapat digunakan dalam algoritma BPNN umumnya yang akan diberikan pada algoritma pelatihan BPNN. Bentuk data (terutama nilai target) merupakan faktor penting

dalam memilih fungsi yang sesuai. Pemilihan bobot awal juga sangat memengaruhi NN dalam mencapai minimum global atau lokal terhadap nilai *error*, serta cepat tidaknya proses pelatihan menuju kekonvergenan. Apabila nilai bobot awal terlalu besar, maka *input* ke setiap lapisan tersembunyi atau lapisan *output* akan jatuh pada daerah dimana turunan fungsi sigmoidnya akan sangat kecil. Sebaliknya, apabila nilai bobot awal terlalu kecil, maka *input* ke setiap lapisan tersembunyi atau lapisan *output* akan sangat kecil, yang akan menyebabkan proses pelatihan akan berjalan sangat lambat. Biasanya bobot awal diinisialisasi secara acak dengan nilai antara $-0{,}5$ sampai 0,5 atau -1 sampai 1 (Kusumadewi, 2004: 97). Algoritma pelatihan dari BPNN sebagai berikut (Fausett, 1994: 292-296).

Langkah 0. Inisialisasi bobot (tetapkan untuk nilai acak kecil).

Langkah 1. Selama kondisi berhenti salah, lakukan langkah 2-9.

Langkah 2. Untuk setiap pasang pelatihan, lakukan langkah 3-8.

Feedforward:

Langkah 3. Setiap neuron *input* $(X_i, i = 1,2, \dots, n)$ menerima sinyal *input* x_i dan menyebarkan sinyal tersebut ke semua neuron pada lapisan atas (neuron tersembunyi).

Langkah 4. Setiap neuron tersembunyi $(Z_j, j = 1,2, \dots, p)$ menjumlahkan bobot sinyal *input*,

$$z\_in_j = v_{j0} + \sum_{i=1}^{n} x_i v_{ji} \tag{2.18}$$

menerapkan fungsi aktivasi untuk menghitung sinyal *output*,

$$z_j = f(z\_in_j) \tag{2.19}$$

dan mengirimkan sinyal tersebut ke semua neuron di lapisan atas (neuron *output*).

Langkah 5. Setiap neuron *output* $(Y_k, k = 1,2, \dots, m)$ menjumlahkan bobot sinyal *input*,

$$y\_in_k = w_{k0+} \sum_{i=1}^{p} z_j w_{kj} \tag{2.20}$$

dan menerapkan fungsi aktivasi untuk menghitung sinyal *output*,

$$y = f(y\_in_k) \tag{2.21}$$

Backpropagation of error:

Langkah 6. Setiap neuron *output* $(Y_k, k = 1,2, \dots, m)$ menerima sebuah pola target yang sesuai dengan pola *input* pelatihan, hitung informasi *error*,

$$\delta_k = (t_k - y_k) f'(y\_in_k) \tag{2.22}$$

hitung koreksi bobot (yang nanti digunakan untuk memperbaiki w_{kj}),

$$\Delta w_{kj} = \alpha \delta_k z_j \tag{2.23}$$

hitung koreksi bias (yang nanti digunakan untuk memperbaiki w_{k0}),

$$\Delta w_{k0} = \alpha \delta_k \tag{2.24}$$

dan mengirimkan δ_k ke neuron pada lapisan bawah.

Langkah 7. Setiap neuron tersembunyi $(Z_j, j = 1,2, \dots, p)$ menjumlahkan *input* delta dari neuron pada lapisan atas (lapisan *output*).

$$\delta\_in_j = \sum_{i=1}^{m} \delta_k w_{kj} \tag{2.25}$$

Kalikan dengan turunan dari fungsi aktivasi untuk menghitung informasi *error*,

$$\delta_j = \delta\_in_j f'(z\_in_j) \quad (2.26)$$

hitung koreksi bobot (yang nanti digunakan untuk memperbaiki v_{ji}),

$$\Delta v_{ji} = \alpha \delta_j x_i \quad (2.27)$$

dan hitung koreksi bias (yang nanti digunakan untuk memperbaiki v_{j0}),

$$\Delta v_{0j} = \alpha \delta_j \quad (2.28)$$

Memperbaiki bobot dan bias:

Langkah 8. Setiap neuron *output* $(Y_k, k = 1,2, \dots, m)$ memperbaiki bias dan bobot $(j = 0, \dots, p)$:

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj} \quad (2.29)$$

$$w_{k0}(baru) = w_{k0}(lama) + \Delta w_{k0} \quad (2.30)$$

Setiap neuron tersembunyi $(Z_j, j = 1,2, \dots, p)$ memperbaiki bias dan bobot $(i = 0, \dots, n)$:

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \quad (2.31)$$

$$v_{j0}(baru) = v_{j0}(lama) + \Delta v_{j0} \quad (2.32)$$

Langkah 9. Kondisi tes berhenti.

Keterangan:

z_in_j : sinyal *input* yang diterima oleh neuron tersembunyi dari neuron *input*

v_{j0} : bobot bias pada lapisan tersembunyi, $j = 1,2, \dots, p$

v_{ji} : bobot dari neuron *input* ke-i menuju neuron tersembunyi ke-j

z_j : sinyal *output* yang telah diaktivasi pada neuron tersembunyi

y_in_k : sinyal *input* yang diterima oleh neuron *output* dari neuron tersembunyi

w_{kj} : bobot dari neuron tersembunyi ke-j menuju neuron *output* ke-k

w_{k0} : bobot bias pada lapisan *output*, $k = 1,2,\dots,m$

y_k : sinyal *output* yang telah diaktivasi pada neuron *output*

t_k : target yang ingin dicapai

α : laju percepatan

Bentuk umum model BPNN secara sistematis dapat dituliskan sebagai berikut.

$$y_t = f\left(\sum_{j=1}^{p} w_{kj}.f\left(v_{j0} + \sum_{i=1}^{n} X_i v_{ji}\right) + w_{k0}\right) \quad (2.33)$$

dengan:

y_t : variabel *output*

w_{kj} : bobot dari neuron tersembunyi ke-j menuju neuron *output* ke-k

v_{j0} : bobot bias pada lapisan tersembunyi, $j = 1,2,\dots,p$

X_i : neuron *input* ke-i, $i = 1,2,\dots,n$

v_{ji} : bobot dari neuron *input* ke-i menuju neuron tersembunyi ke-j

w_{k0} : bobot bias pada lapisan *output*, $k = 1,2,\dots,m$

2.2.5 Algoritma Pelatihan *Backpropagation Neural Network* yang Lebih Cepat

Algoritma BPNN sederhana seringkali terlalu lama untuk keperluan praktis. Algoritma pelatihan dilakukan dalam rangka melakukan pengaturan bobot, sehingga pada akhir pelatihan akan diperoleh bobot-bobot yang baik. Menurut Kusumadewi (2004: 116), prinsip dasar dari algoritma BPNN sederhana adalah memperbaiki bobot-bobot dengan arah yang membuat fungsi kinerja menjadi turun

dengan cepat. Terdapat dua cara algoritma pelatihan sederhana *gradient descent* pada Matlab, yaitu:

1. *Incremental mode*

   Perhitungan gradien dan perbaikan nilai bobot-bobot pada *incremental mode* dilakukan pada setiap pengoperasian *input* data. Untuk menggunakan pelatihan *backpropagation* dengan *incremental mode* digunakan fungsi `adapt`. Fungsi ini menggunakan objek jaringan, kumpulan data *input* dan target sebagai *input* pelatihan, dan akan menghasilkan objek jaringan terlatih, *output* jaringan, *error* yang terjadi dan bobot-bobot akhir sebagai nilai *output*. Terdapat dua fungsi pelatihan untuk bobot-bobot yang menggunakan *gradient descent* pada *incremental mode*, yaitu *gradient descent* (`learngd`) dan *gradient descent* dengan momentum (`learngdm`).

2. *Batch mode*

   Perhitungan gradien dan perbaikan nilai bobot-bobot pada *batch mode* dilakukan setelah pengoperasian semua *input* data. Untuk menggunakan pelatihan *backpropagation* dengan *batch mode* digunakan fungsi `train`. Fungsi ini akan menggunakan objek jaringan, kumpulan data *input* dan target sebagai *input* pelatihan, dan akan menghasilkan objek jaringan telatih, bobot-bobot akhir, dan informasi selama pelatihan (*epoch* dan fungsi kinerja) sebagai nilai *output*. Terdapat beberapa fungsi pelatihan untuk bobot-bobot yang menggunakan *gradient descent* pada *batch mode*, yaitu *gradient descent* (`traingd`) dan *gradient descent* dengan momentum (`traingdm`).

Proses pelatihan kedua pelatihan pada *incremental mode* dan *batch mode* biasanya akan berjalan cukup lambat. Oleh karena itu, untuk lebih mempercepat proses pelatihan, proses pelatihan akan diperbaiki dengan dua alternatif, yaitu dengan menggunakan teknik heuristik dan dengan menggunakan teknik optimasi numeris (*conjugate gradient*). Pelatihan pada Matlab untuk kedua algoritma tersebut disediakan dalam *batch mode*.

1. Perbaikan dengan teknik heuristik

   Teknik ini merupakan pengembangan dari suatu analisis kinerja pada *gradient descent* standar. Terdapat dua algoritma pelatihan perbaikan dengan teknik heuristik yang disediakan oleh Matlab, yaitu:

   1) *Gradient descent* dengan *adaptive learning rate* (`traingda`)

      Fungsi `traingda` akan memperbaiki bobot-bobot berdasarkan *gradient descent* dengan *learning rate* yang bersifat adaptif. Selama proses pelatihan pada *gradient descent* standar (`trangd`), *learning rate* akan terus bernilai konstan. Apabila *learning rate* terlalu tinggi, maka algoritma menjadi tidak stabil. Sebaliknya, jika *learning rate* terlalu kecil maka algoritma akan sangat lama dalam mencapai kekonvergenan. Akan sangat sulit untuk menentukan berapa nilai *learning rate* yang optimal sebelum proses pelatihan berlangsung. Kenyataannya, nilai *learning rate* yang optimal ini akan terus berubah selama proses pelatihan seiring dengan berubahnya nilai fungsi kinerja. Nilai *learning rate* pada *gradient descent* dengan *adaptive learning rate* akan diubah selama proses pelatihan untuk menjaga agar algoritma ini senantiasa stabil selama proses pelatihan.

2) *Gradient descent* dengan momentum dan *adaptive learning rate* (`traingdx`)

   Fungsi `traingdx` akan memperbaiki bobot-bobot berdasarkan *gradient descent* dengan *learning rate* yang bersifat adaptif seperti `traingda`, dan juga dengan menggunakan momentum seperti `traindgm`.

3) *Resilent backpropagation* (`trainrp`)

   NN yang dibangun dengan struktur *multilayer* biasanya menggunakan fungsi aktivasi sigmoid. Fungsi aktivasi ini akan membawa *input* dengan *range* yang tak terbatas ke nilai *output* dengan *range* yang terbatas, yaitu antara 0 sampai 1. Salah satu karakteristik dari fungsi sigmoid adalah gradiennya akan mendekati 0 apabila *input* yang diberikan sangat banyak. Gradien yang mendekati 0 ini berimplikasi pada rendahnya perubahan bobot. Apabila bobot-bobot tidak cukup mengalami perubahan, maka algoritma akan sangat lambat untuk mendekati nilai optimum. Algoritma *resilent backpropagation* berusaha untuk mengeliminasi besarnya efek dari turunan parsial dengan cara hanya menggunakan tanda turunannya saja dan mengabaikan besarnya nilai turunan. Tanda turunan ini akan menentukan arah perbaikan bobot-bobot. Besarnya perubahan setiap bobot akan ditentukan oleh suatu faktor yang diatur pada parameter `delt_inc` atau `delt_dec`. Apabila gradien fungsi kinerja berubah tanda dari satu iterasi berikutnya, maka bobot akan berkurang sebesar `delt_dec`. Sebaliknya, apabila gradien fungsi kinerja tidak berubah tanda dari satu iterasi ke iterasi berikutnya, maka bobot akan bertambah sebesar `delt_inc`. Apabila

gradien fungsi kinerja sama dengan 0, maka perubahan bobot sama dengan perubahan bobot sebelumnya.

2. Perbaikan dengan teknik optimasi numeris (*conjugate gradient)*

   Terdapat beberapa algoritma pada perbaikan dengan teknik optimasi numeris, yaitu:

   1) *Conjugate gradient*

      Seperti pada *gradient descent*, algoritma *conjugate gradient* juga menggunakan gradien dari fungsi kinerja untuk menentukan pengaturan bobot-bobot dalam rangka meminimumkan fungsi kinerja. Pengaturan bobot pada algoritma *gradient descent* dilakukan melalui dalam arah turun (gradien negatif), sedangkan pada algoritma *conjugate gradient* pengaturan bobot tidak selalui dengan arah menurun, tapi disesuaikan dengan arah konjugasinya. Menurut Warsito (2009: 74), *conjugate gradient* menggunakan pendekatan penemuan bobot optimal sepanjang arah gradien dengan fungsi *line search* untuk mencari arah gradien fungsi kinerja. Fungsi *line search* tersebut digunakan untuk menempatkan titik minimum (a), sehingga dapat meminimumkan fungsi kinerja selama arah pencarian. Terdapat beberapa *line search* yang dapat digunakan, yaitu *golden section search* (`srchgol`), *brent's search* (`srchbre`), *hybrid bisection-cubic search* (`srchhyb`), dan *charalambous' search* (`srchcha`). Terdapat beberapa algoritma pencarian untuk menghitung parameter pada arah pencarian, yaitu Fletcher-Reeves (`traincgf`), Polak-Ribiére (`traincgp`),

Powell-Beale Restarts (`traincgb`), dan *scaled conjugate gradient* (`trainscg`).

2) Quasi Newton

Algoritma Newton merupakan salat satu alternatif *conjugate gradient* yang bisa mendapatkan nilai optimum lebih cepat. Algoritma Newton memang berjalan lebih cepat, namun algoritma ini sangat kompleks, memerlukan waktu dan memori yang cukup besar karena pada setiap iterasinya harus menghitung turunan kedua perbaikan dari algoritma ini sering dikenal dengan nama algoritma Quesi-Newton atau algoritma *Secant*. Matriks Hessian pada algoritma ini yang berisi turunan-turunan kedua diganti dengan menghitung suatu fungsi untuk gradien. Algoritma perubahan bobot dengan Quasi-Newton, terdiri dari BFGS dan *one step secant*. BFGS diperkenalkan oleh Broyden, Fletcher, Goldfarb, dan Shanno. Algoritma *one step secant* adalah meode yang menjembatani antara algoritma Quasi-Newton dengan *gradient conjugate*. Algoritma ini tidak menyimpan matriks Hessian secara lengkap, dengan asumsi bahwa setiap iterasi matriks Hessian sebelumnya merupakan matriks identitas, sehingga pencarian arah baru dapat dihitung tanpa harus menghitung invers matriks.

3) Levenberg-Marquardt (`trainlm`)

Seperti halnya algoritma Quasi-Newton, algoritma Levenberg-Marquardt dirancang dengan menggunakan pendekatan turunan kedua tanpa harus menghitung matriks Hessian.

Parameter-parameter pada proses pelatihan turut menentukan keberhasilan proses pelatihan pada algoritma BPNN. Parameter pelatihan dipilih berdasarkan pada penentuan model pelatihan yang digunakan. Beberapa parameter pelatihan tersebut antara lain (Kusumadewi, 2004: 134):

1. Maksimum *epoch*

   Maksimum *epoch* adalah jumlah *epoch* maksimum yang dilakukan selama proses pelatihan. Iterasi akan dihentikan apabila nilai *epoch* melebihi maksimum *epoch*. Nilai *default* untuk makimum *epoch* adalah 10.

2. Kinerja tujuan

   Kinerja tujuan adalah target nilai fungsi kinerja. Iterasi akan dihentikan apabila nilai fungsi kinerja kurang dari atau sama dengan kinerja tujuan. Nilai *default* untuk kinerja tujuan adalah 0.

3. *Learning rate*

   Learning rate adalah laju pelatihan. Semakin besar nilai *learning rate* akan berimplikasi pada semakin besarnya langkah pelatihan. Jika *learning rate* diatur terlalu besar, maka proses pelatihan akan menjadi tidak stabil. Sebaliknya, jika *learning rate* diatur terlalu kecil, maka proses pelatihan akan konvergen dalam jangka waktu yang sangat lama. Nilai *default* untuk *learning rate* adalah 0,01.

4. Maksimum kegagalan

   Maksimum kegagalan diperlukan apabila pada proses pelatihan disertai validitas (optional). Maksimum kegagalan adalah ketidakvalitan terbesar yang diperbolehkan. Apabila gradien pada iterasi ke-k lebih besar daripada gradien iterasi ke-$(k-1)$, maka kegagalannya akan bertambah 1. Iterasi akan

dihentikan apabila jumlah kegagalan lebih dari maksimum kegagalan. Nilai *default* untuk maksimum kegagalan adalah 5.

5. Gradien minimum

   Gradien minimum adalah akar dari jumlah kuadrat semua gradien (bobot *input*, bobot lapisan, bobot bias) terkecil yang diperbolehkan. Iterasi akan dihentikan apabila nilai akar jumlah kuadrat semua gradien ini kurang dari gradien minimum. Nilai *default* untuk gradien minimum adalah 10^{-10}.

6. Momentum

   Momentum adalah perubahan bobot yang baru dengan dasar bobot sebelumnya. Besarnya momentum antara 0 sampai 1. Apabila besarnya momentum sama dengan 0, maka perubahan bobot hanya akan dipengaruhi oleh gradiennya, sedangkan apabila besarnya momentum sama dengan 1, maka perubahan bobot akan sama dengan perubahan bobot sebelumnya. Nilai *default* untuk momentum adalah 0,9.

7. Jumlah *epoch* yang akan ditunjukkan kemajuannya

   Menunjukkan berapa jumlah *epoch* berselang yang akan ditunjukkan kemajuannya. Nilai *default* untuk jumlah *epoch* yang akan ditunjukkan adalah 25.

8. Waktu maksimum untuk pelatihan

   Menunjukkan waktu maksimum yang diijinkan untuk melakukan pelatihan. Iterasi akan dihentikan apabila waktu pelatihan melebihi waktu maksimum. Nilai *default* untuk waktu maksimum adalah tak terbatas (inf).

2.2.6 Prosedur Membangun Jaringan *Backpropagation Neural Network*

Prosedur membangun jaringan BPNN memerlukan beberapa langkah. Langkah-langkah untuk membangun jaringan BPNN sebagai berikut.

1. Penentuan *input* dan target jaringan

   Penentuan *input* dan target jaringan dilakukan dengan melihat lag-lag yang signifikan pada plot ACF dan PACF dari masing-masing variabel. Jika pada plot ACF dan PACF terdapat garis yang melewati selang kepercayaan (garis putus-putus merah) berarti lag tersebut telah signifikan. Data *input* merupakan variabel data masa lalu $(y_{t-1}, y_{t-2}, \dots, y_{t-k})$, sedangkan data target (y_t) merupakan variabel data masa kini. Misalkan, lag-lag yang signifikan pada plot PACF misal terdapat pada lag 1, lag 2, dan lag 7 atau dapat dapat dinyatakan bahwa y_t dipengaruhi oleh y_{t-1}, y_{t-2}, dan y_{t-7}, maka *input* jaringan terdiri atas y_{t-1}, y_{t-2}, dan y_{t-7}.

2. Pembagian data

   Pembagian data dilakukan dengan membagi data menjadi dua bagian, yaitu bagian pertama untuk data latih (*training*) yang digunakan pada tahap pelatihan jaringan dan bagian kedua untuk data uji (*testing*) yang digunakan pada tahap pengujian, dengan komposisi pembagian data sebagai berikut.

   1) 80% untuk data latih dan 20% untuk data uji.
   2) 75% untuk data latih dan 25% untuk data uji.
   3) 60% untuk data latih dan 40% untuk data uji.
   4) 50% untuk data latih dan 50% untuk data uji.

3. Normalisasi data

Sebelum dilakukan pelatihan, menurut Kusumadewi (2004: 183), perlu dilakukan penskalaan pada *input* dan target sedemikian sehingga data-data *input* dan target tersebut masuk dalam suatu *range* tertentu yang disebut dengan normalisasi data. Hal ini dimaksudkan agar data yang diproses sesuai dengan fungsi aktivasi yang digunakan jika data *input* dan target belum sesuai dengan *range* pada fungsi aktivasi. Warsito (2009: 80-81) menyatakan bahwa penskalaan data *input* dan target dengan proses normalisasi akan membawa data ke bentuk normal dengan *mean* sama dengan 0 dan standar deviasi sama dengan 1. Salah satu cara untuk normalisasi data kontinu adalah dengan menggunakan rumus sebagai berikut (Purnomo & Kurniawan, 2006: 32).

$$Y' = \frac{Y_t - Y_{min}}{Y_{maks} - Y_{min}} \tag{2.34}$$

dengan:

Y' : hasil normalisasi

Y_t : data aktual pada *input* maupun taget

Y_{min} : nilai minimum data aktual

Y_{maks} : nilai maksimum data aktual

4. Tahap Pembentukan Model

Model jaringan dibentuk melalui tahap pelatihan dan tahap pengujian.

1) Tahap Pelatihan

Tahap pelatihan merupakan langkah untuk melatih suatu NN, yaitu dengan cara melakukan pengaturan bobot, sehingga akan diperoleh bobot-bobot yang baik pada akhir pelatihan. Menurut Kusumadewi (2004: 116), selama

proses pelatihan, bobot-bobot diatur secara iteratif untuk meminimumkan fungsi kinerja jaringan. Fungsi kinerja jaringan yang sering digunakan adalah MSE. Fungsi ini mengambil rata-rata kuadrat *error* yang terjadi antara *output* jaringan dan target. Warsito (2009: 54) menyatakan bahwa proses pelatihan berlangsung pada beberapa iterasi dan akan berhenti setelah jaringan menemukan bobot yang sesuai dimana nilai *error* (selisih *output* dengan target) yang diinginkan telah tercapai atau jumlah iterasi telah mencapai nilai maksimal yang telah ditetapkan. Hasil dari proses pelatihan, yaitu diperoleh pembaruan bobot jaringan. Tahap pelatihan ini menghasilkan *output* yang sama atau paling tidak mirip dengan target. Hasil pelatihan jaringan akan digunakan untuk menguji jaringan pada tahap pengujian. Algoritma pelatihan dari BPNN ditunjukkan pada bagian 2.2.4. Berikut langkah-langkah pembentukan model terbaik untuk membangun jaringan BPNN.

a. Penentuan jaringan yang optimum

   Proses untuk memperoleh struktur jaringan yang optimum, yaitu dengan menentukan banyaknya lapisan tersembunyi, banyaknya neuron pada lapisan tersembunyi, dan fungsi aktivasi. Neuron tersembunyi berpengaruh pada banyaknya iterasi dan tingginya *error*. Jika banyaknya neuron tersembunyi semakin besar, maka semakin sedikit iterasi yang dibutuhkan untuk mencapai konvergensi dan semakin kecil *error* yang dihasilkan. Sebaliknya, jika banyaknya neuron tersembunyi diatur semakin kecil, maka semakin banyak iterasi yang dibutuhkan

untuk mencapai konvergensi dan semakin besar *error* yang dihasilkan. Fungsi aktivasi digunakan pada lapisan pertama (lapisan tersembunyi) dan lapisan kedua (lapisan *output*). Fungsi aktivasi yang dipilih harus memenuhi syarat kontinu, terdiferensial, dan monoton tidak turun (Fausset, 1994: 192). Fungsi aktivasi untuk BPNN ditunjukkan pada bagian 2.2.3.

b. Penentuan algoritma pelatihan

Terdapat dua cara algoritma pelatihan sederhana *gradient descent* pada Matlab, yaitu *incremental mode* dan *batch mode*. Proses pelatihan kedua pelatihan pada *incremental mode* dan *batch mode* biasanya akan berjalan cukup lambat. Oleh karena itu, untuk lebih mempercepat proses pelatihan, proses pelatihan akan diperbaiki dengan dua alternatif, yaitu dengan menggunakan teknik heuristik dan dengan menggunakan teknik optimasi numeris (*conjugate gradient*). Pelatihan pada Matlab untuk kedua algoritma tersebut disediakan dalam *batch mode*.

c. Penentuan parameter pelatihan

Parameter-parameter pada proses pelatihan turut menentukan keberhasilan proses pelatihan pada algoritma BPNN. Parameter pelatihan dipilih berdasarkan pada algoritma pelatihan yang digunakan.

2) Tahap pengujian

Bobot-bobot yang telah dihasilkan pada tahap pelatihan akan dilakukan pengujian terhadap suatu pola *input* yang belum pernah dilatih sebelumnya, yaitu data uji. Pengujian dilakukan melalui dua tahap, yaitu pengujian

terhadap data yang dilatih dan pengujian pada data uji. Tahap ini akan menentukan hasil keputusan NN dan dari tahap ini akan didapatkan *output* jaringan.

5. Denormalisasi data

Setelah proses pelatihan selesai dilakukan, maka hasil *output* jaringan yang ternormalisasi dikembalikan lagi seperti semula yang disebut dengan denormalisasi data.

6. Tahap penentuan model

Tahap penentuan model dilakukan untuk melihat apakah model yang dipilih sudah cukup baik untuk diramalkan. Tahap penentuan model terdiri dari penentuan model terbaik dan uji kesesuaian model.

1) Penentuan model terbaik

Menurut Kusumadewi (2004: 196), kinerja dari suatu NN setelah dilakukan pelatihan dapat diukur dengan melihat *error* hasil pelatihan, validasi, dan pengujian terhadap sekumpulan data *input* baru. Salah satu cara yang dapat digunakan untuk evaluasi ini dengan menggunakan analisis regresi terhadap respon jaringan dan target yang diharapkan. Penentuan model terbaik dilakukan dengan cara melihat plot dan dengan membandingkan nilai MSE dan MAPE jaringan. Model terbaik adalah model dengan MSE dan MAPE terkecil dengan jaringan paling sederhana.

2) Uji kesesuaian model

Uji kesesuaian model dilakukan untuk melihat apakah model terbaik yang telah diperoleh layak atau tidak untuk digunakan sebagai model peramalan.

Model dianggap baik apabila nilai *error* dari hasil pelatihan data latih bersifat acak yang artinya proses *white noise* terpenuhi. Pengujian ini dapat dilihat dari plot *regression* pelatihan yang dihasilkan. Jika nilai koefisien korelasi mendekati atau sama dengan 1 maka model jaringan layak digunakan untuk peramalan.

2.3 *Recurrent Neural Network*

2.3.1 Konsep Dasar *Recurrent Neural Network*

Menurut Purnomo & Kurniawan (2006: 64), *Recurrent Neural Network* (RNN) adalah NN dengan fasilitas umpan balik menuju neuron itu sendiri maupun neuron yang lain, sehingga aliran informasi dari *input* mempunyai arah banyak (*multidirectional*). *Output* NN tidak hanya tergantung pada *input* saat itu saja, tetapi juga tergantung pada kondisi *input* NN untuk waktu sebelumnya. Kondisi ini dimaksudkan untuk menampung kejadian sebelumnya diikutkan pada proses komputasi berikutnya. Hal ini penting untuk problematika yang cukup rumit dan tanggapan *output* NN berkaitan dengan variasi waktu (*time-varying*), sehingga NN memiliki *sense* terhadap waktu dan memori kondisi sebelumnya. Hal ini erat dengan suatu cabang ilmu yang disebut automaton (*finite automata*/*finite state machine*). Pada prinsipnya RNN sama dengan BPNN dengan tambahan neuron konteks yang hanya menerima *input* internal (*input* balik dari lapisan tersembunyi atau *output*). Conor, dkk (1994) menyatakan bahwa RNN didasarkan pada penyaringan *outlier* dari data dan estimasi parameter dari data yang difilter.

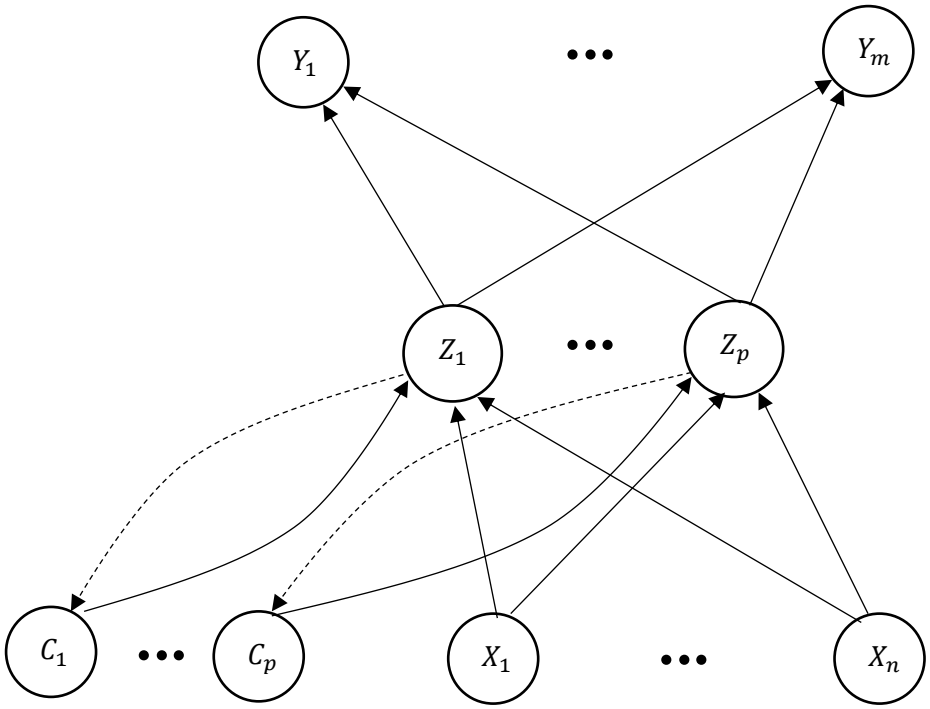
RNN dapat dianggap sebagai jaringan "*recurrent* sebagian", dimana sebagian besar koneksi hanya *feedforward*. Sekelompok neuron tertentu menerima

sinyal umpan balik (*feedback*) dari langkah waktu sebelumnya. Neuron itu diketahui sebagai neuron konteks (Fausett, 1994: 374).

RNN adalah jaringan yang mengakomodasi *output* jaringan untuk menjadi *input* pada jaringan itu lagi dalam rangka menghasilkan *output* jaringan berikutnya. Ada dua macam RNN pada *toolbox* Matlab, yaitu Jaringan Elman dan Jaringan Hopfield (Kusumadewi, 2004).

2.3.2 Arsitektur *Recurrent Neural Network*

Arsitektur untuk RNN sederhana seperti yang ditunjukkan pada Gambar 2.7 (Fausett, 1994: 375).



Gambar 2.7. Arsitektur *Recurrent Neural Network* Sederhana

Aktivasi dari neuron-neuron konteks pada waktu t adalah aktivasi (sinyal *output*) dari neuron-neuron tersembunyi pada langkah waktu sebelumnya. Bobot dari neuron konteks ke neuron tersembunyi dilatih dengan cara yang sama persis dengan bobot dari neuron *input* ke neuron tersembunyi. Dengan demikian, pada

setiap langkah waktu, algoritma pelatihan adalah sama seperti untuk BPNN sederhana.

Model umum jaringan RNN dengan satu lapisan tersembunyi, dengan q neuron *input* dan p neuron pada lapisan tersembunyi adalah (Sani, 2014):

$$y_t = f^o\left(\beta_0 + \sum_{j=1}^{p}\left(\beta_j f^h\left(\gamma_{j0} + \sum_{j=1}^{p} c_j\,\gamma_{dj} + \sum_{i=1}^{n} X_i\gamma_{ji}\right)\right)\right) \qquad (2.35)$$

dengan:

y_t : variabel *output*

β_0 : bobot bias pada lapisan *output*

β_j : bobot dari neuron tersembunyi ke-j menuju neuron *output*, $j = 1,2,\dots,p$

γ_{ji} : bobot dari neuron *input* ke-i menuju neuron tersembunyi ke-j

γ_{dj} : bobot *delay* atau neuron konteks menuju neuron tersembunyi

γ_{j0} : bobot bias pada lapisan tersembunyi, $j = 1,2,\dots,p$

c_j : sinyal yang diterima neuron konteks yang merupakan hasil dari aktivasi sinyal *output* pada lapisan tersembunyi

X_i : neuron *input*, $i = 1,2,\dots,n$

$f^h(x)$: fungsi aktivasi pada lapisan tersembunyi

$f^o(x)$: fungsi aktivasi pada lapisan *output*

2.3.3 Jaringan Elman

Jaringan Elman terdiri dari atas $N1$ lapisan tersembunyi. Lapisan pertama memiliki bobot-bobot yang diperoleh dari lapisan *input*. Seperti halnya NN yang lain, setiap lapisan akan menerima bobot dari lapisan sebelumnya. Semua lapisan

kecuali lapisan terakhir memiliki satu bobot *recurrent*, semua lapisan memiliki bias. Bobot-bobot awal pada setiap lapisan diinisialisasi dengan fungsi `initnw`. Performansi juga akan diukur berdasarkan fungsi kinerja khusus.

Jaringan Elman biasanya menggunakan fungsi aktivasi sigmoid bipolar untuk lapisan tersembunyi (*recurrent*) dan *linear* untuk lapisan *output*. Tidak seperti pada BPNN, pada jaringan Elman ini, fungsi aktivasi dapat berupa sembarang fungsi, baik yang kontinu maupun yang diskontinu. Neuron pada setiap jaringan, ditetapkan cukup banyak, demikian pula jumlah lapisan tersembunyi disesuaikan dengan kompleksitas permasalahan. *Delay* yang terjadi pada waktu sebelumnya $(t-1)$ dapat digunakan untuk waktu saat ini (t). Sehingga apabila dipilih dua RNN dengan bobot-bobot awal yang sama dan diberikan pada *input* yang sama, bisa jadi akan menghasilkan *output* jaringan yang berbeda. Seperti halnya pada BPNN, proses pelatihan pada jaringan Elman dapat dilakukan dengan dua cara, yaitu dengan fungsi `train` dan `adapt` (Kusumadewi, 2004: 327-328).

2.3.4 Jaringan Hopfield

Jaringan Hopfield akan menyimpan sekumpulan titik-titik pada posisi yang seimbang sedemikian rupa sehingga apabila diberikan suatu kondisi awal, maka jaringan akan merespon untuk mendesain suatu titik. Jaringan ini akan bekerja secara rekursif, sedemikian hingga *output* jaringan akan dikirim kembali sebagai *input* jaringan dalam setiap kali operasi.

Jaringan Hopfield dites dengan satu atau lebih vektor *input* yan diberikan sebagai kondisi awal jaringan. Setelah *input* diberikan, jaringan akan merespon untuk menghasilkan suatu *output* yang selanjutnya akan dikirim kembali menjadi

input. Proses ini dilakukan secara terus-menerus. Setiap vektor *input* akan mendekati satu titik keseimbangan yang terdekat. Dasarnya, algoritma Hopfield akan mencoba untuk menstabilkan *output* jaringan, sesuai dengan nilai target yang diberikan oleh pengguna (Kusumadewi, 2004: 333).

2.3.5 Prosedur Membangun Jaringan *Recurrent Neural Network* Tipe Elman

Penelitian pada skripsi ini menggunakan RNN tipe Elman. RNN tipe Elman memiliki tiga lapisan, yaitu lapisan *input*, lapisan tersembunyi, dan lapisan *output*, dimana lapisan *input* dan lapisan *output* ditandai dengan koneksi *feedforward*, lapisan tersembunyi mengandung *recurrent*. Langkah-langkah untuk membangun jaringan RNN tipe Elman sebagai berikut.

1. Penentuan *input* dan target jaringan

   Penentuan *input* dan target jaringan dilakukan dengan melihat lag-lag yang signifikan pada plot ACF dan PACF dari masing-masing variabel. Jika pada plot ACF dan PACF terdapat garis yang melewati selang kepercayaan (garis putus-putus merah) berarti lag tersebut telah signifikan.

2. Pembagian data

   Pembagian data dilakukan dengan membagi data menjadi dua bagian, yaitu bagian pertama untuk data latih (*training*) yang digunakan pada tahap pelatihan jaringan dan bagian kedua untuk data uji (*testing*) yang digunakan pada tahap pengujian, dengan komposisi pembagian data sebagai berikut.

   1) 80% untuk data latih dan 20% untuk data uji.
   2) 75% untuk data latih dan 25% untuk data uji.
   3) 60% untuk data latih dan 40% untuk data uji.

4) 50% untuk data latih dan 50% untuk data uji.

3. Normalisasi data

Sebelum dilakukan pelatihan, menurut Kusumadewi (2004: 183), perlu dilakukan penskalaan pada *input* dan target sedemikian sehingga data-data *input* dan target tersebut masuk dalam suatu *range* tertentu yang disebut dengan normalisasi data. Hal ini dimaksudkan agar data yang diproses sesuai dengan fungsi aktivasi yang digunakan jika data *input* dan target belum sesuai dengan *range* pada fungsi aktivasi. Warsito (2009: 80-81) menyatakan bahwa penskalaan data *input* dan target dengan proses normalisasi akan membawa data ke bentuk normal dengan *mean* sama dengan 0 dan standar deviasi sama dengan 1. Salah satu cara untuk normalisasi data kontinu adalah dengan menggunakan rumus sebagai berikut (Purnomo & Kurniawan, 2006: 32).

$$Y' = \frac{Y_t - Y_{min}}{Y_{maks} - Y_{min}} \tag{2.36}$$

dengan:

Y' : hasil normalisasi

Y_t : data aktual pada *input* maupun taget

Y_{min} : nilai minimum data aktual

Y_{maks} : nilai maksimum data aktual

4. Tahap Pembentukan Model

Model jaringan dibentuk melalui tahap pelatihan dan tahap pengujian.

1) Tahap pelatihan

Proses pelatihan pada RNN tipe Elman menggunakan algoritma pelatihan *backpropagation*. Hasil dari proses pelatihan, yaitu diperoleh pembaruan

bobot jaringan. Tahap pelatihan ini menghasilkan *output* yang sama atau paling tidak mirip dengan target. Hasil pelatihan jaringan akan digunakan untuk menguji jaringan pada tahap pengujian. Langkah-langkah pelatihan dengan *backpropagation* yang digunakan pada RNN tipe Elman sebagai berikut (Laily, dkk, 2018).

Langkah 0. Inisialisasi bobot dengan menggunakan nilai acak kecil

Langkah 1. Lakukan langkah-langkah berikut jika *epoch* $<$ maksimum *epoch* dan MSE $<$ target *error*

Feedforward:

Langkah 3. Setiap neuron *input* $(X_i, i = 1,2, \dots, n)$ menerima sinyal x_i dan menyebarkan sinyal tersebut ke semua neuron pada lapisan selanjutnya (neuron tersembunyi).

Langkah 4. Sinyal yang diterima oleh lapisan tersembunyi dari neuron *input* dirumuskan dengan persamaan:

$$z\_in_{(1)j} = \gamma_{j0} + \sum_{i=1}^{n} x_i \gamma_{ji} \tag{2.37}$$

menerapkan fungsi aktivasi untuk menghitung sinyal *output*,

$$z_{(1)j} = f\left(z\_in_{(1)j}\right) \tag{2.38}$$

Langkah 5. Setiap neuron konteks $\left(c_j, j = 1,2, \dots, p\right)$ menerima sinyal dan meneruskan sinyal ke lapisan tersembunyi. Sinyal diterima oleh neuron konteks adalah hasil dari aktivasi sinyal *output* pada lapisan tersembunyi yang didefinisikan pada persamaan:

$$c_j = z_{(1)j} \tag{2.39}$$

kemudian neuron konteks mengirim kembali sinyal ke neuron tersembunyi yang didefinisikan pada persamaan:

$$z\_in_{(2)j} = \sum_{j=1}^{p} c_j \gamma_{dj} \tag{2.40}$$

Langkah 6. Setiap neuron tersembunyi $(Z_j, j = 1,2, \dots, p)$ menjumlahkan bobot sinyal *input* yang diperoleh dari lapisan *input* dan neuron konteks. Penjumlahan dari sinyal tersebut dirumuskan pada persamaan:

$$z\_in_j = z\_in_{(1)j} + z\_in_{(2)j} = \gamma_{j0} + \sum_{i=1}^{n} x_i \gamma_{ji} + \sum_{j=1}^{p} c_j \gamma_{dj} \tag{2.41}$$

sinyal *ouput* (z_in_j) dihitung dengan menggunakan fungsi aktivasi yang didefinisikan pada persamaan:

$$z_j = f(z\_in_j) \tag{2.42}$$

lalu sinyal tersebut dikirim ke lapisan *output*.

Langkah 7. Setiap neuron *output* $(Y_k k = 1,2, \dots, m)$ menjumlahkan bobot sinyal *input* dari lapisan tersembunyi yang didefinisikan pada persamaan:

$$y\_in_k = \beta_0 + \sum_{j=1}^{p} \beta_j z_j \tag{2.43}$$

dan menerapkan fungsi aktivasi untuk menghitung sinyal *output* (y_in_k),

$$y = f(y\_in_k) \tag{2.44}$$

Backpropagation of error:

Langkah 8. Neuron *output* $(Y_k, k = 1,2, \dots, m)$ menerima sebuah pola target yang sesuai dengan pola *input* pelatihan, hitung informasi *error*. Perhitungan nilai *error* didefinisikan pada persamaan:

$$\delta_k = (t_k - y_k) f'(y\_in_k) \tag{2.45}$$

hitung koreksi bobot (yang nanti digunakan untuk memperbaiki β_j),

$$\Delta\beta_j = \alpha\delta_k z_j \tag{2.46}$$

hitung koreksi bias (yang nanti digunakan untuk memperbaiki β_0),

$$\Delta\beta_0 = \alpha\delta_k \tag{2.47}$$

dan mengirimkan δ_k ke neuron pada lapisan bawah.

Langkah 9. Setiap neuron tersembunyi $(Z_j, j = 1,2, \dots, p)$ menjumlahkan *input* delta dari neuron pada lapisan atas (lapisan *output*) yang dirumuskan pada persamaan:

$$\delta\_in_j = \sum_{i=1}^{m} \delta_k \beta_j \tag{2.48}$$

kalikan dengan turunan dari fungsi aktivasi untuk menghitung informasi *error*,

$$\delta_j = \delta\_in_j f'(z\_in_j) \tag{2.49}$$

hitung koreksi bobot (yang nanti digunakan untuk memperbaiki γ_{ji} dan γ_{dj}),

$$\Delta\gamma_{ji} = \alpha\delta_j x_i \quad (2.50)$$

$$\Delta\gamma_{dj} = \alpha\delta_j c_j \quad (2.51)$$

dan hitung koreksi bias (yang nanti digunakan untuk memperbaiki γ_{j0}),

$$\Delta\gamma_{j0} = \alpha\delta_j \quad (2.52)$$

Memperbaiki bobot dan bias:

Langkah 10. Setiap neuron *output* $(Y_k, k = 1, \dots, m)$ memperbaiki bias dan bobot $(j = 0, \dots, p)$:

$$\beta_j(baru) = \beta_j(lama) + \Delta\beta_j \quad (2.53)$$

$$\beta_0(baru) = \beta_0(lama) + \Delta\beta_0 \quad (2.54)$$

Setiap neuron tersembunyi $(Z_j, j = 1, \dots, p)$ memperbaiki bias dan bobot $(i = 0, \dots, n)$:

$$\gamma_{ji}(baru) = \gamma_{ji}(lama) + \Delta\gamma_{ji} \quad (2.55)$$

$$\gamma_{dj}(baru) = \gamma_{dj}(lama) + \Delta\gamma_{dj} \quad (2.56)$$

$$\gamma_{j0}(baru) = \gamma_{j0}(lama) + \Delta\gamma_{j0} \quad (2.57)$$

Langkah 11. Kondisi tes berhenti.

Keterangan:

$z\_in_{(1)j}$: sinyal *input* yang diterima oleh neuron tersembunyi dari neuron *input*

γ_{j0} : bobot bias pada lapisan tersembunyi

γ_{ji} : bobot dari neuron *input* ke-i menuju neuron tersembunyi ke-j, $j = 1,2, \dots, p$

$z_{(1)j}$: sinyal *output* yang telah diaktivasi pada neuron tersembunyi

c_j : sinyal yang diterima neuron konteks yang merupakan hasil dari aktivasi sinyal *output* pada lapisan tersembunyi

$z\_in_{(2)j}$: sinyal *input* yang diterima oleh neuron tersembunyi dari neuron konteks

z_j : sinyal *output* yang telah diaktivasi pada neuron tersembunyi

γ_{dj} : bobot dari neuron konteks ke-d menuju neuron tersembunyi ke-j, $j = 1,2, \dots, p$

z_in_j : sinyal *input* yang diterima oleh neuron tersembunyi dari neuron *input* dan neuron konteks

y_in_k : sinyal *input* yang diterima oleh neuron *output* dari neuron tersembunyi

y : sinyal *output* yang telah diaktivasi pada neuron *output*

β_0 : bobot bias pada lapisan *output*

β_j : bobot dari neuron tersembunyi menuju neuron *output*

t_k : target yang ingin dicapai

α : laju percepatan

Langkah-langkah pembentukan model terbaik untuk membangun jaringan RNN tipe Elman sebagai berikut.

a. Penentuan jaringan yang optimum

   Proses untuk memperoleh struktur jaringan yang optimum, yaitu dengan menentukan banyaknya lapisan tersembunyi, banyaknya neuron pada lapisan tersembunyi, dan fungsi aktivasi. Neuron tersembunyi

berpengaruh pada banyaknya iterasi dan tingginya *error*. Jika banyaknya neuron tersembunyi semakin besar, maka semakin sedikit iterasi yang dibutuhkan untuk mencapai konvergensi dan semakin kecil *error* yang dihasilkan. Sebaliknya, jika banyaknya neuron tersembunyi diatur semakin kecil, maka semakin banyak iterasi yang dibutuhkan untuk mencapai konvergensi dan semakin besar *error* yang dihasilkan. Fungsi aktivasi digunakan pada lapisan pertama (lapisan tersembunyi) dan lapisan kedua (lapisan *output*). Fungsi aktivasi dapat berupa sembarang fungsi, baik yang kontinu maupun yang diskontinu.

b. Penentuan algoritma pelatihan

Terdapat dua cara algoritma pelatihan sederhana *gradient descent* pada Matlab, yaitu *incremental mode* dan *batch mode*. Proses pelatihan kedua pelatihan pada *incremental mode* dan *batch mode* biasanya akan berjalan cukup lambat. Oleh karena itu, untuk lebih mempercepat proses pelatihan, proses pelatihan akan diperbaiki dengan dua alternatif, yaitu dengan menggunakan teknik heuristik dan dengan menggunakan teknik optimasi numeris (*conjugate gradient*). Pelatihan pada Matlab untuk kedua algoritma tersebut disediakan dalam *batch mode*.

c. Penentuan parameter pelatihan

Parameter-parameter pada proses pelatihan turut menentukan keberhasilan proses pelatihan pada algoritma RNN tipe Elman. Parameter pelatihan dipilih berdasarkan pada algoritma pelatihan yang digunakan.

2) Tahap pengujian

Setelah proses pelatihan pada data latih selesai, maka sistem akan menyimpan bobot dan nilai lapisan konteks pada sistem RNN tipe Elman. Bobot-bobot yang telah dihasilkan pada tahap pelatihan akan dilakukan pengujian terhadap suatu pola *input* yang belum pernah dilatih sebelumnya, yaitu data uji. Pengujian dilakukan melalui dua tahap, yaitu pengujian terhadap data yang dilatih dan pengujian pada data uji. Tahap ini akan menentukan hasil keputusan NN dan dari tahap ini akan didapatkan *output* jaringan.

5. Denormalisasi data

Setelah proses pelatihan selesai dilakukan, maka hasil *output* jaringan yang ternormalisasi dikembalikan lagi seperti semula yang disebut dengan denormalisasi data.

6. Tahap penentuan model

Tahap penentuan model dilakukan untuk melihat apakah model yang dipilih sudah cukup baik untuk diramalkan. Tahap penentuan model terdiri dari penentuan model terbaik dan uji kesesuaian model.

1) Penentuan model terbaik

Menurut Kusumadewi (2004: 196), kinerja dari suatu NN setelah dilakukan pelatihan dapat diukur dengan melihat *error* hasil pelatihan, validasi, dan pengujian terhadap sekumpulan data *input* baru. Salah satu cara yang dapat digunakan untuk evaluasi ini dengan menggunakan analisis regresi terhadap respon jaringan dan target yang diharapkan. Penentuan model terbaik

dilakukan dengan cara melihat plot dan dengan membandingkan nilai MSE dan MAPE jaringan. Model terbaik adalah model dengan MSE dan MAPE terkecil dengan jaringan paling sederhana.

2) Uji kesesuaian model

Uji kesesuaian model dilakukan untuk melihat apakah model terbaik yang telah diperoleh layak atau tidak untuk digunakan sebagai model peramalan. Model dianggap baik apabila nilai *error* dari hasil pelatihan data latih bersifat acak yang artinya proses *white noise* terpenuhi. Pengujian ini dapat dilihat dari plot *regression* pelatihan yang dihasilkan. Jika nilai koefisien korelasi mendekati atau sama dengan 1 maka model jaringan layak digunakan untuk peramalan.

2.4 *Mean Square Error* (MSE)

Mean Square Error (MSE) digunakan untuk mengukur kesalahan nilai dugaan model yang dinyatakan dalam *mean* dari kuadrat kesalahan. Tidak ada yang dapat memastikan bahwa model peramalan yang dipilih akan cocok dengan data yang ada secara tepat. Kriteria yang digunakan untuk mengukur ketepatan model peramalan adalah MSE. Rumus untuk menentukan nilai MSE dinyatakan dengan persamaan (2.58) (Hendikawati, 2015: 95).

$$MSE = \frac{\sum_{t=1}^{n}\left(Y_t - \hat{Y}_t\right)^2}{n} \tag{2.58}$$

dengan:

Y_t : nilai pengamatan pada periode ke-t

$\hat{Y}_t$: nilai peramalan pada periode ke-t

n : banyak pengamatan

2.5 *Mean Absolute Percentage Error* (MAPE)

Mean Absolute Percentage Error (MAPE) digunakan untuk mengukur kesalahan nilai dugaan model dinyatakan dalam bentuk persentase *mean absolute* kesalahan. MAPE umumnya tidak digunakan untuk memilih berbagai alternatif model. Rumus untuk menentukan nilai MAPE dinyatakan dalam persamaan (2.59) (Hendikawati, 2015: 96).

$$MAPE = 100\% \times \frac{\sum_{t=1}^{n} \left| \frac{Y_t - \hat{Y}_t}{Y_t} \right|}{n} \tag{2.59}$$

dengan:

Y_t : nilai pengamatan pada periode ke-t

$\hat{Y}_t$: nilai peramalan pada periode ke-t

n : banyak pengamatan

Perhitungan MAPE akan digunakan untuk mencari akurasi. Rumus untuk menentukan nilai akurasi dinyatakan dalam persamaan (2.60).

$$Akurasi = 100\% - MAPE \tag{2.60}$$

Nilai MAPE yang dihasilkan menunjukkan kemampuan peramalan dengan kriteria sebagai berikut (Rahmadiani dan Anggraeni, 2012).

1. MAPE < 10% : Peramalan sangat baik
2. 10% ≤ MAPE < 20% : Peramalan baik
3. 20% ≤ MAPE < 50% : Peramalan cukup baik
4. MAPE ≥ 50% : Peramalan buruk

2.6 Penjualan

Menurut Zulkarnain (2012: 10-15), penjualan merupakan tujuan dari pemasaran artinya perusahaan melalui departemen atau bagian pemasaran termasuk tenaga penjualan (*sales force*) akan berupaya melakukan kegiatan penjualan untuk menghabiskan produk yang dihasilkan. Penjualan itu sendiri merupakan kegiatan yang terkait proses produksi, finansial, sumber daya manusia, riset, dan pengembangan dan seterusnya sehingga tidak mungkin penjualan yang berhasil tidak disinergikan dengan aspek lainnya dalam perusahaan. Penjualan bukanlah aktivitas yang berdiri sendiri tetapi ditopang oleh aktivitas lainnya dengan tujuan untuk menyampaikan barang atau jasa ke konsumen. Tiga alasan pokok yang mendasari pentingnya mempelajari penjualan yakni setiap orang adalah penjualan, semua organisasi membutuhkan penjualan, dan banyaknya peluang karir dalam bidang ini.

Menurut Swastha (1989: 8-11), menjual adalah ilmu dan seni mempengaruhi pribadi yang dilakukan oleh penjual untuk mengajak orang lain agar bersedia membeli barang atau jasa yang ditawarkannya. Jadi, adanya penjualan dapat tercipta suatu proses pertukaran barang dan atau jasa antara penjual dengan pembeli. Jenis-jenis penjualan dikelompokkan menjadi lima, yaitu:

1. *Trade selling*

   Trade selling dapat terjadi bilamana produsen dan pedagang besar mempersilahkan pengecer untuk berusaha memperbaiki distributor produk-produk mereka. Hal ini melibatkan para penyalur dengan kegiatan promosi,

peragaan, persediaan, dan produk baru. Jadi, titik beratnya adalah pada "penjualan melalui" penyalur daripada "penjualan ke" pembeli akhir.

2. *Missionary selling*

   Missionary selling, penjualan berusaha ditingkatkan dengan mendorong pembeli untuk membeli barang-barang dari penyalur perusahaan. Wiraniaga lebih cenderung pada "penjualan untuk" penyalur. Jadi, wiraniaga sendiri tidak menjual secara langsung produk yang ditawarkan, misalnya penawaran obat kepada dokter.

3. *Technical selling*

   Technical selling berusaha meningkatkan penjualan dengan pemberian saran dan nasehat kepada pembeli akhir barang dan jasanya. Hal ini, tugas utama wiraniaga adalah mengidentifikasi dan menganalisis masalah-masalah yang dihadapi pembeli, serta menunjukkan bagaimana produk atau jasa yang ditawarkan dapat mengatasi masalah tersebut.

4. *New business selling*

   New business selling berusaha membuka transaksi baru dengan merubah calon pembeli menjadi pembeli. Jenis penjualan ini sering dipakai oleh perusahaan asuransi.

5. *Responsive selling*

   Setiap tenaga penjualan diharapkan dapat memberikan reaksi terhadap permintaan pembeli. Dua jenis penjualan utama disini adalah *route driving* dan *retailing*. Para pengemudi yang menghantarkan susu, roti, gas untuk keperluan rumah tangga, para pelayan di took serba ada, toko pakaian, toko spesial,

merupakan contoh dari jenis penjualan ini. Jenis penjualan seperti ini tidak akan menciptakan penjualan yang terlalu besar meskipun layanan yang baik dan hubungan pelanggan yang menyenangkan dapat menjurus kepada pembelian ulang.

Umumnya, para pengusaha mempunyai tujuan mendapatkan laba tertentu (mungkin maksimal) dan mempertahankan atau bahkan berusaha meningkatkannya untuk jangka waktu lama. Tujuan tersebut dapat direalisir apabila penjualan dapat dilaksanakan seperti direncanakan. Demikian tidak berarti bahwa barang atau jasa yang terjual selalu akan menghasilkan laba. Bagi perusahaan, pada umumnya mempunyai tiga tujuan umum dalam penjualannya, yaitu (Swastha, 1989: 80):

1. Mencapai *volume* penjualan tertentu.
2. Mendapatkan laba tertentu.
3. Menunjang pertumbuhan perusahaan.

2.7 Peramalan

Peramalan adalah suatu teknik untuk meramalkan keadaan di masa yang akan datang melalui pengujian keadaan di masa lalu. Dasarnya meramalkan sama halnya dengan memprediksi atau memperkirakan suatu hal, kejadian atau peristiwa masa datang yang berdasar pada masa lalu hingga saat ini.

Peramalan adalah suatu proses memperkirakan secara sistematik tentang apa yang paling mungkin terjadi di masa depan berdasar informasi masa lalu dan sekarang yang dimiliki agar kesalahannya (selisih antara apa yang terjadi dengan hasil perkiraan) dapat diperkecil. Peramalan dapat juga diartikan sebagai usaha memperkirakan perubahan, agar tidak disalahpahami bahwa peramalan tidak

memberi jawaban pasti tentang apa yang terjadi, melainkan berusaha mencari yang sedekat mungkin dengan yang akan terjadi (Mulyono, 2000: 1).

Peramalan adalah data di masa lalu yang digunakan untuk keperluan estimasi data yang akan datang. Peramalan atau *forecasting* merupakan bagian terpenting bagi setiap perusahaan ataupun organisasi bisnis dalam setiap pengambilan keputusan manajemen. Organisasi selalu menentukan sasaran dan tujuan, berusaha menduga faktor-faktor lingkungan, lalu memilih tindakan yang diharapkan akan menghasilkan pencapaian sasaran dan tujuan tersebut. Kebutuhan akan peramalan meningkat sejalan dengan usaha manajemen untuk mengurangi ketergantungannya pada hal-hal yang belum pasti. Peramalan menjadi lebih ilmiah sifatnya dalam menghadapi lingkungan manajemen. Organisasi perlu memiliki pengetahuan dan keterampilan yang meliputi paling sedikit empat bidang untuk melakukan peramalan, yaitu (Makridakis dkk, 1999: 4-6):

1. Identifikasi dan definisi masalah peramalan.
2. Aplikasi serangkaian metode peramalan.
3. Prosedur pemilihan metode yang tepat untuk situasi tertentu.
4. Dukungan organisasi untuk menerapkan dan menggunakan metode peramalan secara formal.

Ada dua hal pokok yang harus diperhatikan dalam proses peramalan yang akurat dan bermanfaat antara lain (Hendikawati, 2015: 2):

1. Pengumpulan data yang relevan berupa informasi yang dapat menghasilkan peramalan yang akurat.

2. Pemilihan teknik peramalan yang tepat yang akan memanfaatkan informasi data yang diperoleh semaksimal mungkin.

Menurut Hendikawati (2015: 2), pada dasarnya terdapat dua pendekatan untuk melakukan peramalan yaitu dengan pendekatan kualitatif dan pendekatan kuantitatif. Metode kualitatif dapat dibagi menjadi metode eksploratoris dan normatif. Metode peramalan kualitatif digunakan ketika data historis tidak tersedia. Metode peramalan kualitatif ini adalah metode subyektif (intuitif). Metode ini mendasar pada informasi kualitatif. Dengan dasar informasi tersebut dapat diprediksi kejadian-kejadian di masa yang akan datang.

Metode peramalan kuantitatif dapat dibagi menjadi dua tipe, yaitu metode regresi (*causal*) dan runtun waktu (*time series*). Metode peramalan kausal meliputi faktor-faktor yang berhubungan dengan variabel yang diprediksi. Sebaliknya, peramalan runtun waktu merupakan metode kuantitatif untuk pendugaan berdasarkan data masa lalu dari suatu variabel yang telah dikumpulkan secara teratur. Data lampau tersebut dengan teknik yang tepat dapat dijadikan acuan untuk peramalan nilai di masa yang akan datang. Tujuan metode peramalan runtun waktu adalah menemukan pola dalam deret data historis mengekstrapolasikan pola tersebut ke masa depan. Model kausal mengasumsikan bahwa faktor yang diramalkan menunjukkan suatu hubungan sebab akibat dengan satu atau lebih variabel bebas. Model runtun waktu dapat digunakan dengan mudah untuk meramal, sedangkan model kausal lebih berhasil untuk pengambilan keputusan dan kebijakan. Peramalan kuantitatif dapat diterapkan bila terdapat tiga kondisi berikut (Makridakis dkk, 1999: 8):

1. Tersedia informasi tentang masa lalu.
2. Informasi tersebut dapat dikuantitatifkan dalam bentuk data numerik.
3. Dapat diasumsikan bahwa beberapa aspek pola masa lalu akan terus berlanjut di masa mendatang.

2.8 Peramalan Penjualan

Menurut Nafarin (2004: 31), ramalan penjualan merupakan proses kegiatan memperkirakan produk yang akan dijual pada waktu yang akan datang dalam keadaan tertentu dan dibuat berdasarkan data yang pernah terjadi dan atau mungkin akan terjadi. Teknik membuat ramalan penjualan dapat dilakukan secara kualitatif dan kuantitatif atau gabungan keduanya. Ramalan penjualan yang dibuat secara kualitatif, seperti dengan menggunakan metode penilaian atau pendapat (*judgment method*). Sumber penilaian yang dipakai sebagai dasar melakukan ramalan penjualan antara lain penilaian dari pramuniaga, penilaian manajer, pemasaran, penilaian para ahli, atau survei konsumen. Peramalan penjualan yang dibuat secara kuantitatif, umumnya menggunakan metode statistik, tetapi dapat juga dengan metode pendapat atau dengan metode khusus. Metode khusus seperti metode industri, analisis lini produk, analisis penggunaan akhir. Baik metode statistik maupun metode pendapat tidak menjamin ketepatan ramalan penjualan dengan realisasi. Ramalan penjualan akan mendekati realisasi penjualan, apabila ramalan penjualan tersebut dapat disusun berdasarkan kontrak jual beli (*sales contract*).

Menurut Nafarin (2007: 96), peramalan (*forecasting*) adalah proses aktivitas meramalkan suatu kejadian yang mungkin terjadi di masa mendatang dengan cara mengkaji data yang ada. Jualan (*sales*) artinya hasil proses menjual atau yang dijual

atau hasil penjualan. Penjualan (*selling*) artinya proses menjual. Peramalan penjualan berarti proses meramalkan produk yang dijual dari perusahaan tertentu dan pada saat tertentu. Peramalan penjualan merupakan faktor penting dalam perencanaan perusahaan karena peramalan penjualan menentukan anggaran penjualan dan anggaran penjualan menentukan anggaran produk, anggaran biaya pabrik, anggaran beban usaha, anggaran kas, anggaran laba rugi, dan anggaran neraca.

2.9 MATLAB (*Matrix Laboratory*)

Menurut Arhami & Desiani (2005: 1), Matlab adalah sebuah program untuk analisis dan komputasi numerik yang merupakan suatu bahasa pemrograman matematika lanjutan yang dibentuk dengan dasar pemikiran menggunakan sifat dan bentuk matriks. Awalnya, program ini merupakan *interface* untuk koleksi rutin-rutin numerik proyek LINPACK dan EISPACK, dikembangkan menggunakan bahasa FORTRAN. Namun sekarang, program ini merupakan produk komersial dari perusahaan Mathworks, Inc. yang dalam perkembangan selanjutnya dikembangkan menggunakan bahasa C++ dan assembler (terutama untuk fungsi-fungsi dasar Matlab).

Matlab telah berkembang menjadi sebuah *environment* pemrograman yang canggih dan berisi fungsi-fungsi *built-in* untuk melakukan tugas pengolahan sinyal, aljabar linier, dan kalkulasi matematis lainnya. Matlab juga berisi *toolbox* yang berisi fungsi-fungsi tambahan untuk aplikasi khusus. Matlab bersifat *extensible*, dalam arti bahwa seorang pengguna dapat menulis fungsi baru untuk ditambahkan di *library* jika fungsi-fungsi *built-in* yang tersedia tidak dapat melakukan tugas

tertentu. Matlab yang merupakan bahasa pemrograman tingkat tinggi berbasis pada matriks sering digunakan untuk teknik komputasi numerik, digunakan untuk menyelesaikan masalah-masalah yang melibatkan operasi matematika elemen, matrik, optimasi, aproksimasi, dan lain-lain. Ada beberapa macam *window* yang tersedia dalam Matlab yang dapat dijelaskan sebagai berikut:

1. Matlab *Command Window/Editor*

   Matlab *command window/editor* merupakan *window* yang pertama kali setiap kali Matlab dijalankan. *Command window* digunakan untuk memanggil *tool* Matlab seperti *editor*, *debugger*, atau fungsi. Ciri *window* ini adalah prompt (>>) yang menyatakan Matlab siap menerima perintah. Perintah dapat berupa fungsi-fungsi pengaturan file (seperti perintah DOS/UNIX) maupun fungsi-fungsi pengaturan bawaan/*toolbox* Matlab sendiri.

2. Matlab *Editor/Debigger* (*Editor* M-File/Pencarian Kesalahan)

   Window ini merupakan *tool* yang disediakan oleh Matlab versi 5 ke atas yang berfungsi sebagai editor *script* Matlab (M-File). Walaupun sebenarnya *script* ini dalam pemrograman Matlab dapat saja menggunakan *editor* lain seperti *Notepad*, *Wordpad*, bahkan *Word*.

3. *Figure Window*

   Window ini adalah hasil visualisasi *script* Matlab. Namun, Matlab memberi kemudahan bagi *programmer* untuk mengedit *window* ini sekaligus memberikan program khusus untuk itu sehingga *window* ini selain berfungsi visualisasi *output* dapat juga sekaligus media input interaktif.

4. Matlab *Help Window*

   Matlab menyediakan system *help* yang dapat diakses dengan perintah *help*.

2.10 Penelitian yang Relevan

Penelitian mengenai implementasi NN untuk menyelesaikan masalah peramalan sudah banyak dikembangkan. Penelitian yang relevan dengan penelitian ini sebagai berikut.

1. Kurniawan (2017) dengan judul "Penerapan Algoritma *Feedforward Neural Network* (FFNN) *Backpropagation* untuk Meramalkan Harga Saham". Penelitian tersebut menerapkan algoritma FFNN dengan algoritma BPNN untuk meramalkan harga saham dengan data harga sama harian PT United Tractors. Ada dua variabel yang digunakan sebagau *input* datanya, yaitu harga pembukaan saham dan harga penutupan saham. Proses analisis algoritma FFNN menggunakan fungsi aktivasi sigmoid bipolar, *linear*, dan algoritma *traingdx*. Kesimpulan akhir dari penelitian tersebut, yaitu diperoleh struktur jaringan terbaik dengan 3 neuron *input* dan 10 neuron lapisan tersembunyi. Peramalan harga saham menghasilkan nilai MAPE sebesar 0,0145753 untuk proses pelatihan (*training*), 0,044259 untuk proses pengujian (*testing*), dan Rp17.405,00 untuk prediksi harga saham periode selanjutnya.
2. Misriati (2016) dengan judul "Peramalan Jumlah Kunjungan Wisata ke Lombok Menggunakan Jaringan Syaraf Tiruan". Penelitian tersebut menggunakan algoritma pelatihan BPNN untuk meramalkan jumlah kunjungan wisatawan ke Lombok dengan data jumlah kunjungan wisatawan mancanegara melalui Bandara Lombok. Penelitian tersebut menggunakan fungsi aktivasi tansig.

Kesimpulan akhir dari penelitian tersebut, yaitu menghasilkan arsitektur jaringan 2 neuron *input*, 6 neuron lapisan tersembunyi, 1 neuron *output* dan diperoleh nilai MSE terbaik sebesar 0,00000000277. Hasil peramalan diperoleh pada bulan Maret 2015 sebesar 5.823 kunjungan dengan kesalahan sebesar 3% dari data aktual yang sebanyak 6.004 kunjungan, bulan April 2015 dengan presentase kesalahan sebesar 1,54% dari data aktual sebanyak 5.725 kunjungan dengan hasil ramalan sebanyak 5.813 kunjugan, dan bulan Mei 2015 dengan presentase kesalahan sebesar 24% dari data aktual sebanyak 5.713 kunjungan dengan hasil ramalan sebanyak 4.347 kunjungan wisata.

3. Salman & Prasetio (2010) dengan judul "Implementasi Jaringan Syaraf Tiruan *Recurrent* dengan Algoritma Pembelajaran *Gradient Descent Adaptive Learning Rate* untuk Pendugaan Curah Hujan Berdasarkan Peubah ENSO (*El-Nin Southern Oscilation*)". Penelitian tersebut menggunakan jaringan *recurrent* tipe Elman dengan 2 lapisan tersembunyi dengan algoritma pembelajaran optimasi teknik heuristik *gradient descent adaptive learning rate* untuk pendugaan curah hujan berdasarkan peubah ENSO. Kesimpulan akhir dari penelitian tersebut, yaitu menghasilkan pendugaan curah hujan terbaik untuk kelompok data pertama adalah pada leap 0 menghasilkan nilai R^2 maksimum 69,2%, leap 1 menghasilkan nilai R^2 maksimum 66,5%, leap 2 menghasilkan nilai R^2 maksimum 61,6%, dan leap 3 menghasilkan nilai R^2 maksimum 55,5%, sedangkan kelompok data kedua adalah 50% data pelatihan dan 50% data pengujian menghasilkan R^2 maksimum 53,6% untuk leap 0. Hasil nilai R^2 pada leap 0 lebih baik dibandingkan pada leap1, leap 2, dan leap 3.

4. Pakaja, dkk (2012) dengan judul "Peramalan Penjualan Mobil Menggunakan Jaringan Syaraf Tiruan dan *Certaintly Factor*". Penelitian tersebut menggunakan algoritma *certainty factor* sebagai nilai pembanding pada bobot koreksi yang telah dilatih dalam BPNN untuk peramalan penjualan mobil Honda. Penelitian tersebut menggunakan fungsi aktivasi sigmoid biner dengan konstanta belajar (α) sebesar 0,1. Kesimpulan akhir dari penelitian tersebut, yaitu diperoleh peramalan pada tahun 2015 akan terjual mobil Honda sebanyak 29.579 unit dengan nilai target *error* sebesar 4,205%.
5. Rizal & Hartati (2017) dengan judul "Prediksi Kunjungan Wisatawan dengan *Recurrent Neural Network Extended Kalman Filter*". Penelitian tersebut menggunakan pendekatan RNN untuk memprediksi data *time series* pada kunjungan wisatawan di pulau Lombok. Penelitian tersebut menggunakan algoritma pelatihan *Extended Kalman Filter*. Penelitian tersebut menggunakan komposisi pembagian data untuk data latih sebesar 70% dan data uji sebesar 30%. Kesimpulan akhir dari penelitian tersebut, yaitu diperoleh prediksi *time series* dengan RNN memberikan akurasi prediksi terbaik pada saat pelatihan sebesar 64,37% dan hasil prediksi terbaik pada saat pengujian sebesar 62,91%.
6. Udin, dkk (2017) dengan judul "Peramalan Kapasitas Baterai *Lead Acid* pada Mobil Listrik Berbasis *levenberg Marquardt Neural Network*". Penelitian tersebut membahas peramalan kapasitas baterai mobil listrik berbasis kecerdasan buatan *levenberg marquardt neural network* dengan menggunakan RNN dan BPNN. Kesimpulan dari penelitian tersebut, yaitu diperoleh hasil peramalan kapasitas baterai sebesar 140 Ah saat kecepatan akselerasi maka

akan habis dalam waktu 1,09 jam, sedangkan pada saat kecepatan maksimum maka akan habis dalam 0,88 jam dan saat beban kecepatan tanjakan maka akan habis dalam waktu 0,64 jam. Metode terbaik dalam meramalkan sisa kapasitas baterai adalah dengan metode RNN. Hal ini ditunjukkan salah satunya saat menggunakan kecepatan maksimum (16,67 m/s) hasil peramalan setelah 30 menit berjalan sisa kapasitas dari BPNN adalah 61 Ah dan RNN adalah 60,5 Ah, sedangkan targetnya adalah 59,5 Ah.

2.11 Kerangka Berpikir

Peramalan adalah suatu teknik untuk meramalkan keadaan di masa yang akan datang melalui pengujian keadaan di masa lalu. Terdapat dua pendekatan untuk melakukan peramalan yaitu dengan pendekatan kualitatif dan pendekatan kauntitatif. Metode peramalan kuantitatif dapat dibagi menjadi dua tipe yaitu metode regresi atau kausal (*causal*) dan runtun waktu (*time series*). Peramalan runtun waktu merupakan metode kuantitatif untuk pendugaan berdasarkan data masa lalu dari suatu variabel yang telah dikumpulkan secara teratur.

Salah satu metode yang dapat digunakan untuk peramalan runtun waktu adalah *neural network.* Metode *neural network* memiliki akurasi yang sangat tinggi dalam melakukan peramalan runtun waktu jika dibandingkan dengan metode regresi linier. Salah satu kelebihan dari *neural network* untuk peramalan adalah dapat meramalkan data runtun waktu non linier. Berdasarkan kemampuan belajar yang dimilikinya, maka *neural network* dapat dilatih untuk mempelajari dan menganalisa pola data masa lalu dan berusaha mencari suatu formula atau fungsi

yang akan menghubungkan pola data masa lalu dengan *output* yang diinginkan pada saat ini (Salman, 2011).

Algoritma pada *neural network* dapat dibagi menjadi dua, yaitu *feedforward network* dan *recurrent/feedback network*. Algoritma pada *feedforward network* yang sering dikaji adalah *backpropagation neural network*, sedangkan pada *recurrent network* yang sering dikaji adalah *recurrent neural network* tipe Elman. *Backpropagation neural network* merupakan algoritma pelatihan terawasi yang bergerak maju dan tidak memiliki loop dimana aliran sinyalnya dari neuron *input* ke neuron *output* tanpa mengetahui korelasi antar data, sedangkan *recurrent neural network* merupakan algoritma *neural network* berulang yang arsitekturnya memiliki minimal satu loop koneksi balik dan neuron yang menerima sinyal koneksi balik dari langkah waktu sebelumnya disebut dengan neuron konteks. *Recurrent neural network* tipe Elman memiliki beberapa lapisan tersembunyi dengan lapisan pertama memiliki bobot-bobot yang diperoleh dari lapisan *input*. Tidak seperti pada *backpropagation neural network*, pada *recurrent neural network* tipe Elman fungsi aktivasi dapat berupa sembarang fungsi, baik yang kontinu maupun yang diskontinu. Pada prinsipnya *recunrrent neural network* sama dengan *backpropagation neural network*, perbedaannya terletak pada tambahan neuron konteks yang hanya menerima *input* internal (*input* balik dari lapisan tersembunyi atau *output*).

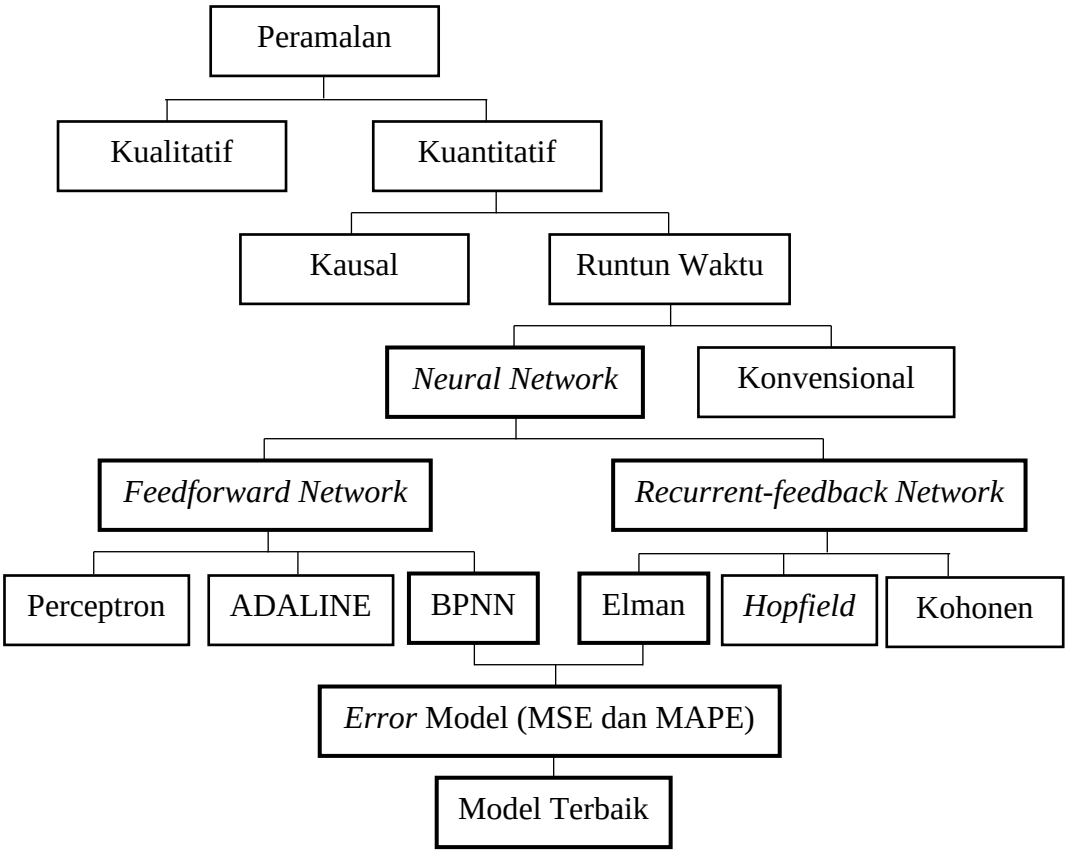
Baik algoritma *backpropagation neural network* maupun *recurrent neural network* tipe Elman dimulai dengan memasukkan pola-pola pelatihan atau yang disebut dengan data latih ke dalam jaringan. Pola masukan dihitung mulai dari

lapisan *input* menuju lapisan tersembunyi diterukan menuju lapisan *output* dengan menggunakan fungsi aktivasi yang ditentukan. Namun, berbeda jika pada *recurrent neural network* yang terdapat lapisan konteks, sehingga pola masukan dihitung mulai dari lapisan *input* menuju lapisan tersembunyi dilanjutkan ke lapisan konteks kembali lagi menuju lapisan tersembunyi yang kemudian diteruskan menuju lapisan *output* dengan menggunakan fungsi aktivasi yang telah ditentukan. Selanjutnya, lapisan *output* memberikan tanggapan yang disebut dengan *output* jaringan. Kemudian *output* jaringan dibandingkan dengan target yang harus dicapai dengan menghitung *error* yang diperoleh dari selisih antara target dengan *output* jaringan. Jika *error* yang dihasilkan lebih kecil dari target *error* maka iterasi dihentikan. Akan tetapi, jika *error* yang dihasilkan lebih besar dari target *error* maka bobot setiap jaringan diperbaiki dengan jalan mempropagasi kembali *error*. Setiap pembaruan bobot yang terjadi diharapkan dapat mengurangi besarnya *error*. Pelatihan akan berlangsung pada beberapa iterasi atau dapat disebut dengan *epoch* dan berhenti setelah jaringan menemukan bobot yang sesuai dengan target *error* atau jumlah *epoch* telah mencapai nilai maksimal yang sudah ditentukan sebelumnya. Tahap pelatihan dilakukan untuk mengatur bobot, sehingga akan diperoleh bobot-bobot yang baik pada akhir pelatihan. Hasil pelatihan jaringan akan digunakan untuk menguji jaringan.

Bobot-bobot yang telah dihasilkan pada tahap pelatihan akan dilakukan pengujian terhadap suatu pola *input* yang belum pernah dilatih sebelumnya, yaitu data uji. Bobot-bobot dari hasil tahap pelatihan diharapkan dapat menghasilkan *error* minimal yang juga akan memberikan *error* yang kecil pada tahap pengujian.

Pengujian akan dilakukan melalui dua tahap, yaitu pengujian terhadap data yang sudah dilatih dan pengujian pada data uji. Pengujian jaringan akan menentukan hasil keputusan *neural network* dan akan didapatkan *output* jaringan. Model terbaik dipilih berdasarkan model dengan MSE dan MAPE terkecil dengan jaringan paling sederhana. Model terbaik dari masing-masing model pada *backpropagation neural network* dan *recurrent neural network* tipe Elman yang telah diperoleh selanjutnya dilakukan peramalan untuk masa yang akan datang. Model peramalan yang dipilih adalah model yang memiliki nilai akurasi terbesar pada hasil peramalannya. Selanjutnya, model dengan nilai akurasi terbaik yang merupakan model yang optimal untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk.

Hasil yang diharapkan dari penelitian ini adalah untuk mendapatkan model terbaik diantara *backpropagation neural network* dan *recurrent neural network* untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk. Hasil penelitian ini dapat digunakan sebagai salah satu alternatif pengambilan keputusan bagi perusahaan dalam meramalkan penjualan sehingga ketika penjualan diprediksi dengan akurat maka pemenuhan permintaan konsumen dapat diusahakan tepat waktu, kerjasama perusahaan dengan relasi tetap terjaga dengan baik, kepuasan konsumen terpenuhi, perusahaan dapat mengatasi hilangnya penjualan atau kehabisan stok, mencegah pelanggan lari ke kompetitor.



Gambar 2.8. Kerangka Berpikir

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan yang telah diuraikan pada bab sebelumnya, maka dapat ditarik kesimpulan sebagai berikut.

1. Model BPNN untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk adalah model BPNN (9-5-1) yang terbangun dari 9 neuron pada lapisan *input*, 5 neuron pada lapisan tersembunyi, dan 1 neuron pada lapisan *output* dengan algoritma pelatihan *Levenberg-Marquardt* dengan inisialisasi Mu yang digunakan adalah 0,02 dan fungsi aktivasi yang digunakan adalah logsig. Persamaan model BPNN (9-5-1) sebagai berikut.

$$y_t = \left(\sum_{j=1}^{5} w_{kj} . \frac{1}{1 + e^{-(v_{j0} + \sum_{i=1}^{9} x_i v_{ji})}} \right) + w_{k0}$$

2. Model RNN tipe Elman untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk adalah model RNN tipe Elman (9-5-1) yang terbangun dari 9 neuron pada lapisan *input*, 5 neuron pada lapisan tersembunyi, dan 1 neuron pada lapisan *output* dengan algoritma pelatihan *gradient descent* dengan momentum dan *adaptive learning rate* dengan momentum yang digunakan adalah 0,2, *learning rate* yang digunakan adalah 0,2, dan fungsi aktivasi yang digunakan adalah logsig. Persamaan model RNN tipe Elman (9-5-1) sebagai berikut.

$$y_t = \beta_0 + \left(\sum_{j=1}^{5} \beta_j . \frac{1}{1 + e^{-(\gamma_{j0} + \gamma_{dj} + \sum_{i=1}^{9} \gamma_{ji} X_i)}} \right)$$

3. Model terbaik untuk peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk adalah model BPNN (9-5-1) dengan tingkat akurasi sebesar 87,9727% dan merupakan peramalan baik. Hasil peramalan penjualan semen di PT Semen Indonesia (Persero) Tbk menggunakan model BPNN (9-5-1) dalam skala bulanan untuk sembilan bulan berikutnya untuk Bulan April 2018 sebesar 2479607 ton, Bulan Mei 2018 sebesar 2344701 ton, Bulan Juni 2018 sebesar 2045132 ton, Bulan Juli 2018 sebesar 2486581 ton, Bulan Agustus 2018 sebesar 2674669 ton, Bulan September 2018 sebesar 2379005 ton, Bulan Oktober 2018 sebesar 2834896 ton, Bulan November 2018 sebesar 2501668 ton, dan Bulan Desember 2018 sebesar 2918820 ton.

5.2 Saran

Berdasarkan hasil dan pembahasan, saran yang dapat diberikan untuk penelitian selanjutnya sebagai berikut.

1. Untuk menghasilkan proses pelatihan agar berjalan lebih cepat dapat dilakukan penelitian terkait optimasi penentuan bobot awal atau dapat menggunakan metode Nguyen-Widrow.
2. Untuk mendapatkan arsitektur model yang lebih baik gunakan lebih banyak variasi pada neuron tersembunyi, lapisan tersembunyi, fungsi aktivasi, algoritma pelatihan, dan parameter-parameter pelatihan yang akan digunakan. Penentuan parameter-parameter akan menentukan besar kecilnya *error* pada tahap pelatihan, tahap pengujian, dan peramalan.

3. Jika pada penelitian ini menggunakan RNN tipe Elman, pada penelitian lain dapat menggunakan RNN tipe *Hopfield*.
4. Jika ingin menggunakan RNN sebagai model peramalan, maka pola data yang digunakan untuk peramalan lebih baik memiliki karakteristik data berulang atau data musiman agar diperoleh hasil peramalan yang memiliki nilai akurasi yang tinggi.
5. Untuk perusahaan jika ingin menggunakan penelitian ini dapat melakukan penelitian lebih lanjut dengan menggunakan metode gabungan, seperti penggabungan antara BPNN dengan *fuzzy* atau penggabungan RNN dengan *fuzzy*, penggabungan BPNN dengan algoritma genetika atau penggabungan RNN dengan algoritma genetika, penggabungan BPNN dengan *hybrid* atau penggabungan RNN dengan *hybrid*, dan sebagainya. Hal ini memungkinkan dapat menghasilkan peramalan yang lebih baik nilai akurasinya.

DAFTAR PUSTAKA

Arhami, M., & Desiani, A. 2005. *Pemrograman MATLAB*. Yogyakarta: ANDI.

Arsyad, L. 1994. *Peramalan Bisnis*. Edisi Pertama. Yogyakarta: BPFE.

Bahadir, E. 2016. "Prediction of Prospective Mathematics Teachers' Academic Success in Entering Graduate Education by Using Back-propagation Neural Network". *Journal of Education and Training Studies* (online). https://files.eric.ed.gov/fulltext/EJ1094642.pdf Vol. IV (5): 113-122. (diunduh tanggal 29 Oktober 2018).

Chakraborty, K., Mehrotra, K., Mohan, C. K., & Ranka, S. 1992. "Forecasting the Behavior of Multivariate Time Series Using Neural Networks". *Neural Networks* (online). https://cyber.sci-hub.mu/MTAuMTAxNi9zMDg5My02MDgwKDA1KTgwMDkyLTk=/chakraborty1992.pdf#view=FitH Vol. V: 961—970. (diunduh tanggal 1 Agustus 2018).

Connor, J. T., Martin, D. G., & Atlas, L. E. 1994. "Recurrent Neural Networks and Robust Time Series Prediction". *IEEE Transactions on Neural Networks* (online). https://dacemirror.sci-hub.mu/journal-article/861dde49215aa7182c8b6193a6dcc20a/connor1994.pdf#view=FitH Vol. V (2): 240—254.

Fausett, L. 1994. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. New Jersey: Prentice-Hall.

Hardianto, H. N. I., Suyanto, & Purnama, B. 2011. "Analisis dan Implementasi Diferential Evolution dan Recurrent Neural Network untuk Prediksi Data Time Series Studi Kasus Kurs Jual Emas". *Tugas Akhir*. Universitas Telkom.

Hendikawati, P. 2015. *Peramalan Data Runtun Waktu (Metode dan Aplikasinya dengan Minitab & Eviews)*. Semarang: FMIPA Unnes.

Hermawan, Arief. 2006. *Jaringan Saraf Tiruan: Teori dan Aplikasi*. Yogyakarta: ANDI.

Hikmah, A. 2017. "Peramalan Deret Waktu Menggunakan Autoregressive (AR), Jaringan Syaraf Tiruan Radial Basis Function (RBF), dan Hibrid AR-RBF pada Inflasi Indonesia". *Unnes Journal of Mathematics* (online). http://skripsi.unnes.ac.id/v2/skripsi/baca/195354/991.aspx ISSN 0215-9945 (diunduh tanggal 1 Agustus 2018).

Kaastra, I., & Boyd, M. 1996. "Designing a Neural Network for Forecasting Financial and Economic Time Series". *Neurocomputing* (online). https://cyber.sci-hub.mu/MTAuMTAxNi8wOTI1LTIzMTIoOTUpMDAwMzktOQ==/kaastra1996.pdf#view=FitH Nomor 10: 215—236. (diunduh tanggal 1 Agustus 2018).

Kohzadi, N., Boyd, M. S., Kermanshahi, B., & Kaastra, I. 1996. "A Comparison of Artificial Neural Network and Time Series Models for Forecasting Commodity Prices". *Neurocomputing* (online). https://cyber.sci-hub.mu/MTAuMTAxNi8wOTI1LTIzMTIoOTUpMDAwMjAtOA==/kohzadi1996.pdf#view=FitH Nomor 10: 169—181. (diunduh tanggal 1 Agustus 2018).

Kontan. 2017. "Permintaan Semen Nasional Tumbuh 7,8%". http://industri.kontan.co.id/news/permintaan-semen-nasional-tumbuh-78. (diakses tanggal 21 April 2018).

Kurniawan, M. A., Kharis, M., & Sugiharti, E. 2017. "Penerapan Metode Feed Forward Neural Network (FFNN) Backpropagation untuk Meramalkan Harga Saham". *Unnes Journal of Mathematics* (online). http://skripsi.unnes.ac.id/v2/skripsi/baca/188335/302.aspx. (diunduh tanggal 17 Juli 2018).

Kusumadewi, F. 2014. "Peramalan Harga Emas Menggunakan Feedforward Neural Networks Dengan Algoritma Backpropagation". *Skripsi*. Universitas Negeri Yogyakarta.

Kusumadewi, S. 2004. *Membangun Jaringan Syaraf Tiruan: menggunakan MATLAB & Excel Link*. Yogyakarta: Graha Ilmu.

Kusumadewi, S., & Hartati, S. 2010. *Neuro-Fuzzy: Integrasi Sistem Fuzzy & Jaringan Syaraf*. Edisi Kedua. Yogyakarta: Graha Ilmu.

Laily, V. O. N., Warsito, B., & Maruddani, D. A. I. 2018. "Comparison of ARCH/GARCH model and Elman Recurrent Neural Network on data return of closing price stock". *Journal of Physics* (online). http://iopscience.iop.org/article/10.1088/1742-6596/1025/1/012103/pdf. (diunduh tanggal 25 Agustus 2018).

Makridakis, S., Wheelwright, S. C., & McGee, V.E. 1999. *Metode dan Aplikasi Peramalan.* Edisi 2 Jilid 1. Terjemahan. Untung Sus Adriyanto dan Abdul Basith. Jakarta: Erlangga.

Misriati, T. 2016. "Peramalan Jumlah Kunjungan Wisatawan Mancanegara ke Lombok Menggunakan Jaringan Syaraf Tiruan". *Jurnal Seminal Nasional Ilmu Pengetahuan dan Teknologi Komputer Nusa Mandiri* (online). https://konferensi.nusamandiri.ac.id/prosiding/index.php/sniptek/article/download/6/3/. (diunduh tanggal 18 Juli 2018).

Muis, S. 2006. *Teknik Jaringan Syaraf Tiruan*. Edisi Pertama. Yogyakarta: Graha Ilmu.

Mulyono, S. 2000. *Peramalan Bisnis dan Ekonometrika*. Yogyakarta: BPFE.

Nafarin, M. 2004. *Penganggaran Perusahaan*. Jakarta: Salemba Empat.

_____.2007. *Penganggaran Perusahaan*. Jakarta: Salemba Empat.

Pakaja, F., Naba, A., & Purwanto. 2012. "Peramalan Penjualan Mobil Menggunakan Jaringan Syaraf Tiruan dan Certainty Factor". *Jurnal EECCIS* (online). http://jurnaleeccis.ub.ac.id/index.php/eeccis/article/viewFile/162/140 Vol. VI (1): 23-28. (diunduh tanggal 24 Oktober 2017).

Portal BUMN. 2013. "Semen Indonesia Waspadai Ketatnya Persaingan Bisnis Semen". http://bumn.go.id/semenindonesia/berita/703/Semen.Indonesia. (diakses tanggal 20 April 2018).

_____. 2018. "Semen Indonesia Optimis Kelebihan Pasokan Semen Tahun Ini Menyusut". http://bumn.go.id/semenindonesia/berita/1-Semen-Indonesia-optimis-kelebihan-pasokan-semen-tahun-ini-menyusut. (diakses tanggal 20 April 2018).

Puspitaningrum, D. 2006. *Pengantar Jaringan Saraf Tiruan*.Yogayakarta: ANDI.

Purnomo, M. H., & Kurniawan, A. 2006. *Supervised Neural Networks dan Aplikasinya*. Yogyakarta: Graha Ilmu.

Rahmadiani, A., & Anggraeni, W. 2012. "Implementasi Fuzzy Neural Network untuk Memperkirakan Jumlah Kunjungan Pasien Poli Bedah di Rumah Sakit Onkologi Surabaya". *Jurnal Teknik ITS* (online). http://ejurnal.its.ac.id/index.php/teknik/article/download/1241/566. Vol. I (diunduh 10 Februari 2019).

Ren, C., An, N., Wang, J., Li, L., Hu, B., & Shang, D. 2013. "Optimal Parameters Selection for BP Neural Network Based on Particle Swarm Optimization: A Case Study of Wind Speed Forecasting". *Knowlede-Based Systems* (online). https://dacemirror.sci-hub.mu/journal-

article/79c2be8e478713dbcb4a7365748a8800/ren2014.pdf#view=FitH Nomor 56: 226-239. (diunduh tanggal 1 Agustus 2018).

Rizal, A. A., & Hartati, S. 2017. "Prediksi Kunjungan Wisatawan dengan Recurrent Neural Network Extended Kalman Filter". *Jurnal Ilmiah Ilmu Komputer* (online). https://ojs.unud.ac.id/4c1bec7e-4ce6-4523-b054-c48cd0dcf752 Vol. X (1): 7—18. (diakses tanggal 24 Agustus 2018).

Salman, A. G., & Prasetio, Y. L. 2010. "Implementasi Jaringan Syaraf Tiruan Recurrent Dengan Metode Pembelajaran Gradient Adaptive Learning Rate untuk Pendugaan Curah Hujan Berdasarkan Peubah ENSO". *Jurnal ComTech* (online). http://research-dashboard.binus.ac.id/uploads/paper/document/publication/Proceeding/ComTech/Vol.%2001%20No.%202%20Desember%202010/19%20-%20Afan%20Galih%20-%20OK.pdf Vol. I (2): 418-429. (diunduh tanggal 18 Juli 2018).

Sani, D. L. 2014. "Penerapan Elman-Recurrent Neural Network pada Peramalan Listrik Jangka Pendek di PT. PLN App Malang". *Jurnal Mahasiswa Statistik* (online). http://download.portalgaruda.org/article.php?article=192263&val=6491&title=PENERAPAN%20ELMAN-RECURRENT%20NEURAL%20NETWORK%20PADA%20PERAMALAN%20KONSUMSI%20LISTRIK%20JANGKA%20PENDEK%20DI%20PT.%20PLN%20APP%20MALANG Vol. II (6): 441-444. (diunduh tanggal 24 Agustus 2018).

Semen Indonesia. 2017. "Permintaan Semen Bakal Terus Naik". http://www.semenindonesia.com/permintaan-semen-bakal-terus-naik/. (diakses tanggal 21 April 2018).

_____. 2018. "Laporan Penjualan". http://www.semenindonesia.com/laporan-penjualan/. (diakses tanggal 18 April 2018).

Siang, J. J. 2004. *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan Matlab*. Yogyakarta: Penerbit Andi.

Siswanto. 2010. *Kecerdasan Tiruan*. Edisi Kedua. Yogyakarta: Graha Ilmu.

Suhada, B. 2009. "Peramalan Produksi Gula Nasional Melalui Pendekatan Artificial Neural Network". *Jurnal Derivatif* (online). http://download.portalgaruda.org/article.php?article=455334&val=9580&title=PERAMALAN%20PRODUKSI%20GULA%20NASIONAL%20ME

LALUI%20%20PENDEKATAN%20ARTIFICIAL%20NEURAL%20NETWORK Vol. III (1): 50—63. (diunduh tanggal 18 Juli 2018).

Susanti, L. A. D., Arna, F., & Sethiawardana. 2013. "Peramalan Harga Saham Menggunakan Recurrent Neural Network dengan Algoritma Backpropagation Through Time (BPTT)". *Makalah Proyek Akhir*. Surabaya: Institut Teknologi Sepuluh Nopember.

Swastha, B. 1989. *Manajemen Penjualan*. Edisi Ketiga. Yogyakarta: BPFE.

Udin, M., Kaloko, B. S., & Hardianto, T. 2017. "Peramalan Kapasitas Baterai Lead Acid pada Mobil Listrik Berbasis Levenberg Marquardt Neural Network". *Berkala Saintek* (online). https://jurnal.unej.ac.id/index.php/BST/article/download/5703/4250/ Vol. V (2): 112—117. (diunduh tanggal 24 Agustus 2018).

Valipour, M., Banihabib, M. E., & Behbahani, S. M. R. 2013. "Comparison of the ARMA, ARIMA, and the Autoregressive Artificial Neural Network Models in Forecasting the Monthly Inflow of Dez Dam Reservoir". *Journal of Hydrology* (online). https://www.sciencedirect.com/science/article/pii/S002216941200981X 476: 433—441. (diunduh tanggal 5 September 2018).

Walid, & Alamsyah. 2017. "Recurrent Neural Network For Forecasting Time Series With Long Memory Pattern". *Journal of Physics: Conference Series* (online). https://www.researchgate.net/publication/316219802_Recurrent_Neural_Network_For_Forecasting_Time_Series_With_Long_Memory_Pattern/fulltext/58f6627e45851506cd30e464/316219802_Recurrent_Neural_Network_For_Forecasting_Time_Series_With_Long_Memory_Pattern.pdf?origin=publication_detail. (diunduh tanggal 2 September 2018).

Walid, Subanar, Rosadi, D., & Suhartono. 2015. "Fractional Integrated Recurrent Neural Network (FIRNN) for Forecasting of Time Series Data in Electricity Load in Jawa-Bali". *Contemporary Engineering Sciences* (online). https://www.researchgate.net/profile/Dedi_Rosadi/publication/290456999_Fractional_integrated_recurrent_neural_network_FIRNN_for_forecasting_of_time_series_data_in_electricity_load_in_java-bali/links/575527a408ae02ac12811b8e/Fractional-integrated-recurrent-neural-network-FIRNN-for-forecasting-of-time-series-data-in-electricity-load-in-java-bali.pdf Vol. VIII (32): 1535—1550. (diunduh tanggal 2 September 2018).

Wang, L., Wang, Z. G., & Liu, S. 2018. "Optimal Forecast Combination Based on Neural Networks for Time Series Forecasting". *Apllied Soft Computing Journal* (online). https://doi.org/10.1016/j.asoc.2018.02.004. (diunduh tanggal 2 September 2018).

Warsito, B. 2009. *Kapita Selekta Statistika Neural Network*. Semarang: BP UNDIP.

Wikipedia. 2018. "Semen Indonesia". http://id.wikipedia.org/wiki/Semen_Indonesia. (diakses tanggal 18 April 2018).

Yang, Y., Hu, J., Lv, Y., & Zhang, M. 2013. "Predictions on the Development Dimensions of Provincial Tourism Discipline Based on the Artificial Neural Network BP Model". "*Higher Education Studies* (online). https://files.eric.ed.gov/fulltext/EJ1080203.pdf Vol. III (3): 13—20. (diunduh tanggal 29 Oktober 2018).

Zhang, G. P. 2004. *Business Forecasting with Artificial Neural Networks: An Overview*. Hershey, PA: Idea Group Publishing.

Zhang, G., & Hu, M. Y. 1998. "Neural Network Forecasting of the British Pound/US Dollar Exchange Rate". *Omega* (online). https://twin.sci-hub.tw/6545/62e451225bbfe1f3da87c81b1bd5a671/zhang1998.pdf#view=FitH Vol. XXVI (4): 495—506. (diunduh tanggal 1 Agustur 2018).

Zulkarnain. 2012. *Ilmu Menjual: Pendekatan Teoritis dan Kecakapan Menjual*. Yogyakarta: Graha Ilmu.