BAB 2

TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Jaringan Saraf Tiruan (*Artificial Neural Network*)

Menurut Haykin (2009), Jaringan Saraf Tiruan (JST) atau *Artificial Neural Network (ANN)* adalah sebuah jaringan yang dirancang untuk menyerupai otak manusia yang bertujuan untuk melaksanakan suatu tugas tertentu. Jaringan ini biasanya diimplementasikan dengan menggunakan komponen elektronik atau disimulasikan pada aplikasi komputer.

2.1.1.1 Definisi

Terdapat berbagai macam definisi Jaringan Saraf Tiruan (JST), diantaranya:

- Menurut Haykin (2009), Jaringan Saraf Tiruan adalah prosesor yang terdistribusi besar – besaran secara parallel yang dibuat dari unit proses sederhana, yang mempunyai kemampuan untuk menyimpan pengetahuan berupa pengalaman dan dapat digunakan untuk proses lain. JST menyerupai otak manusia dalam dua hal:
  1. Pengetahuan didapat oleh jaringan dari lingkungan melalui proses pembelajaran.
  2. Tenaga koneksi *Interneuron*, yang diketahui sebagai bobot sinaptik (*synaptic weights)*, yang digunakan untuk menyimpan pengetahuan yang didapat.
- Menurut Fausett (1994), Jaringan Saraf Tiruan adalah proses sistem informasi yang mempunyai performa karakteristik yang jelas dalam kesamaannya dengan jaringan saraf dalam bidang biologi. JST sudah dikembangkan sebagai generalisasi model matematika dari pengertian manusia atau saraf biologi, berdasarkan asumsi bahwa:

1. Proses informasi yang terjadi di banyak elemen sederhana yang disebut neuron.
2. Sinyal dikirimkan melalui hubungan koneksi di antara neuron – neuron.
3. Setiap hubungan koneksi mempunyai bobot yang saling berhubungan, di mana jaringan saraf yang khas memperbanyak sinyal yang ditransmisi.
4. Setiap neuron mengaplikasikan sebuah fungsi aktifasi (biasanya berupa non – linear*)* ke jaringan *input*nya (jumlah dari bobot sinyal *input*) untuk menentukan sinyal *output*nya.

2.1.1.2 Sejarah

Awal dari Jaringan Saraf Tiruan dimulai pada tahun 1940an. Jaringan Saraf Tiruan pertama dirancang oleh Warren McCulloch dan Walter Pitts pada tahun 1943 yang dikenal dengan *McCulloch-Pitts neurons.* Peneliti ini menyadari bahwa dengan menggabungkan banyak neuron sederhana menjadi sistem saraf adalah sumber peningkatan kemampuan berhitung.

Donald Hebb merancang hukum pembelajaran pertama untuk Jaringan Saraf Tiruan yang dikenal dengan Hebb Learning. Pemikiran dia adalah jika dua neuron aktif secara bersamaan maka kekuatan koneksi antara neuron seharusnya bertambah.

Pada tahun 1950-an dan 1960-an, Jaringan Saraf Tiruan memasuki masa keemasannya. Dimulai oleh Frank Rosenblatt bersama beberapa peneliti lainnya mengenalkan dan mengembangkan sekumpulan besar Jaringan Saraf Tiruan yang disebut *perceptrons.* Aturan pembelajaran *perceptrons* menggunakan pengulangan untuk penyesuaian bobot yang lebih efektif daripada aturan Hebb.

Bernard Widrow dan muridnya, Marcian (Ted) Hoff [Widrow-Hoff, 1960], mengembangkan aturan pembelajaran yang berhubungan dekat dengan aturan pembelajaran

perceptron. Aturan ini menyesuaikan bobot untuk mengurangi perbedaan antara net *input* ke unit *output* yang diharapkan. Aturan pembelajaran Widrow-Hoff untuk *single-layer network* adalah pelopor dari aturan *backpropagation* untuk *multilayer nets.* Jaringan ini disebut juga *ADALINE* yang diinterpretasi dari *Adaptive Linear Neuron* atau *Adaptive Linear System.*

Pada tahun 1970-an pengembangan Jaringan Saraf Tiruan memasuki masa sunyi, karena ada demonstrasi dari Minsky dan Papert dari terbatasnya *perceptrons* (contohnya *single-layer nets*) yang tidak mampu menyelesaikan masalah sederhana seperti *XOR* dan kurangnya metode umum untuk melatih jaringan *multilayer net*. Walaupun begitu penelitian untuk Jaringan Saraf Tiruan masih berlanjut. Banyak dari pemimpin – pemimpin yang mulai mempublikasikan pekerjaan mereka, seperti Kohonen, Anderson, Grossberg, dan Carpenter.

Pada tahun 1980-an antusias dari peneliti kembali terbentuk, karena ditemukannya metode baru untuk menyelesaikan masalah *XOR* tersebut yaitu metode *backpropagation*. Metode ini berguna untuk menyebarkan informasi tentang kesalahan – kesalahan pada unit *output* kembali ke unit *hidden.*

Tokoh kunci lain yang membuat antusias dari peneliti kembali adalah John Hopfield. Bersamaan dengan David Tank, Hopfield mengembangkan beberapa Jaringan Saraf Tiruan berdasarkan bobot tetap dan aktifasi yang dapat beradaptasi. Selain itu juga ada *Neocognition* yang ditemukan oleh Kunihiko Fukushima dan teman – temannya.

2.1.1.3 Keuntungan

Menurut Haykin (2009), Jaringan Saraf Tiruan memberikan banyak sifat dan kemampuan yang berguna diantaranya:

1. *Nonlinearity*

Nonlinearity adalah sifat yang sangat berguna jika mekanisme fisik yang mendasari bertanggung jawab atas pembuatan sinyal *input* adalah *non-linear*.

2. *Input-output Mapping*
Paradigma pembelajaran popular pada saat ini yang disebut *learning with a teacher* atau *supervised learning*, melibatkan modifikasi dari bobot sinaptik Jaringan Saraf Tiruan dengan mengaplikasikan sekumpulan contoh *training* atau contoh tugas. Pelatihan pada jaringan dilakukan berulang – ulang kali untuk semua contoh yang ada, sampai jaringan tersebut mencapai keadaan yang stabil di mana tidak ada perbedaan yang jauh di dalam bobot sinaptik (*synaptic weights*).

3. *Adaptivity*
Jaringan Saraf Tiruan mempunyai kemampuan untuk mengadaptasi bobot sinaptiknya untuk mengubah lingkungannya. Pada intinya, Jaringan Saraf Tiruan dilatih untuk beroperasi didalam lingkungan tertentu dapat dilatih ulang dengan mudah untuk menghadapai perubahan kecil di dalam kondisi lingkungan yang sedang beroperasi.

4. *Evidential Response*
Dalam konteks pengenalan pola, sebuah Jaringan Saraf Tiruan dapat dirancang untuk menyediakan informasi tidak hanya tentang pola mana saja yang harus dipilih, tapi juga keyakinan pada keputusan yang dibuat.

5. *Contextual Information*
Pengetahuan yang direpresentasikan oleh struktur dan keadaan aktifasi Jaringan Saraf Tiruan. Setiap neuron di dalam jaringan berpotensi dipengaruhi oleh aktifitas global dari seluruh neuron lainnya di dalam jaringan.

6. *Fault Tolerance*
Jaringan Saraf Tiruan diimplementasikan dalam bentuk perangkat keras, mempunyai potensi untuk diwariskannya *Fault Tolerance* atau kemampuan untuk berhitung dengan tepat dalam arti performanya

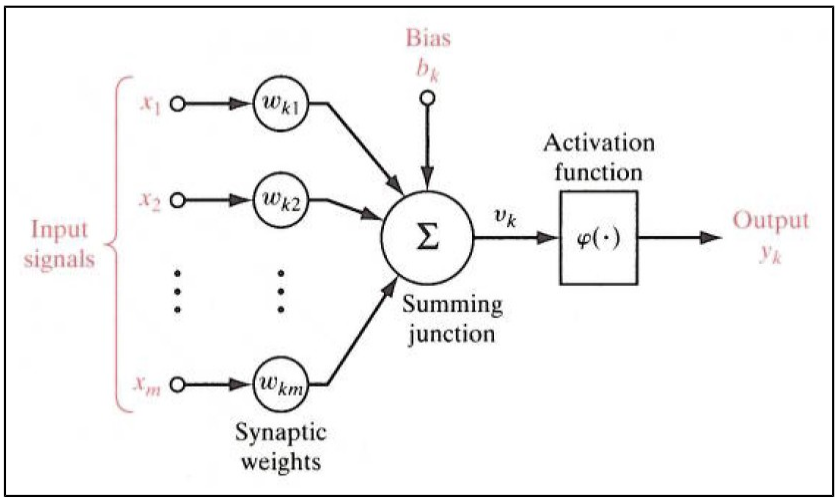
menurun secara perlahan dalam kondisi operasi yang merugikan.

7. *VLSI Implementability*
   Sifat paralel yang besar – besaran dari Jaringan Saraf Tiruan membuatnya berpotensi cepat untuk menghitung tugas tertentu. Fitur yang sama ini membuat Jaringan Saraf Tiruan cocok untuk implementasi menggunakan teknologi *very-large-scale-integrated* (VLSI).
8. *Uniformity of Analysis and Design*
   Pada dasarnya, Jaringan Saraf Tiruan menikmati universalitas sebagai pengolah informasi. Fitur ini menunjukkan dirinya sendiri dalam berbagai cara :
   a. Neuron, dalam satu bentuk atau lainnya merepresentasikan sebuah bahan umum untuk semua Jaringan Saraf Tiruan
   b. Keumuman ini membuatnya memungkinkan untuk menyebarkan teori dan algoritma pembelajaran dalam aplikasi yang berbeda dari Jaringan Saraf Tiruan
   c. Jaringan – jaringan modular dapat dibentuk melalui integrasi modul (*seamless integration of modules*).
9. *Neurobiological Analogy*
   Rancangan dari Jaringan Saraf Tiruan dimotivasi oleh analogi dengan otak, yang menjadi bukti hidup bahwa proses paralel *fault-tolerant* tidak hanya mungkin secara fisik, tapi juga cepat dan kuat. *Neurobiologist* melihat Jaringan Saraf Tiruan sebagai alat penelitian untuk menafsirkan fenomena neurobiologis (*neurobiological phenomena*).

2.1.1.4 Model Neuron

Menurut Haykin (2009), neuron adalah unit pengolah informasi yang merupakan dasar dari proses sebuah Jaringan Saraf Tiruan. Dijelaskan juga ada tiga elemen dasar dari model saraf yaitu:

1. Satu set dari sinapsis, atau penghubung yang masing-masing digolongkan oleh bobot atau kekuatannya.
2. Sebuah penambah untuk menjumlahkan sinyal-sinyal *input*. Ditimbang dari kekuatan sinaptik masing-masing neuron.
3. Sebuah fungsi aktivasi untuk membatasi amplitudo *output* dari neuron. Fungsi ini bertujuan membatasi jarak amplitude yang diperbolehkan oleh sinyal *output* menjadi sebuah angka yang terbatas.

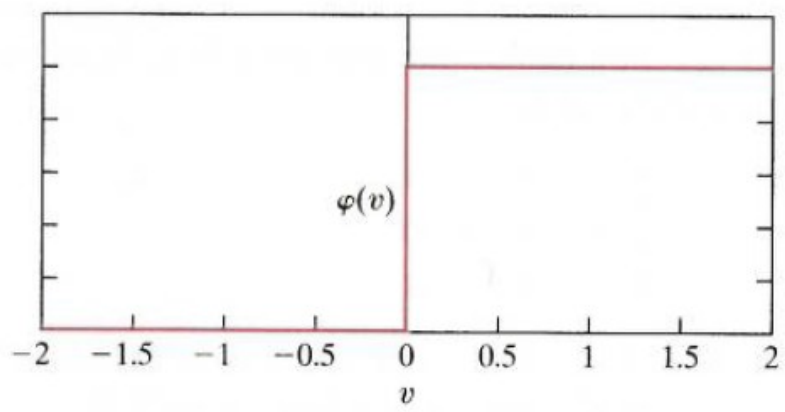


Gambar 2.1 Model neuron *non-linear*

Model saraf juga mencakup bias diterapkan secara eksternal. Bias memiliki efek meningkatkan atau menurunkan *input* bersih fungsi aktivasi tergantung pada apakah positif atau negatif. Fungsi aktivasi mendefinisikan *output* dari *neuron*. Ada dua tipe dasar dari fungsi aktivasi:

1. *Threshold Function* (fungsi aktivasi *threshold*). Di engineering bentuk fungsi ini sering disebut juga *Heaviside function*.

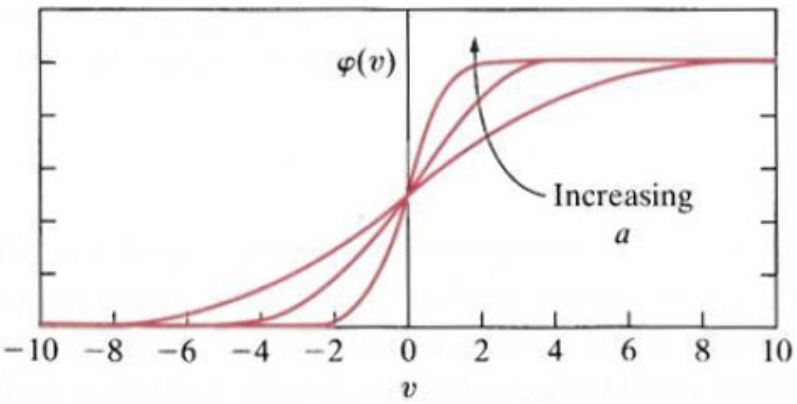
$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$



Gambar 2.2 Grafik Fungsi *Threshold*

2. *Sigmoid Function* (fungsi aktivasi *sigmoid*). Fungsi aktivasi sigmoid merupakan bentuk paling umum dari fungsi aktivasi yang digunakan untuk pembuatan Jaringan Saraf Tiruan. Fungsi yang meningkat secara tepat yang memperlihatkan keseimbangan yang tinggi antara sifat *linear* dan *non-linear*.

$$\varphi(v) = \frac{1}{1 + \exp(-av)}$$



Gambar 2.3 Grafik Fungsi *Sigmoid*

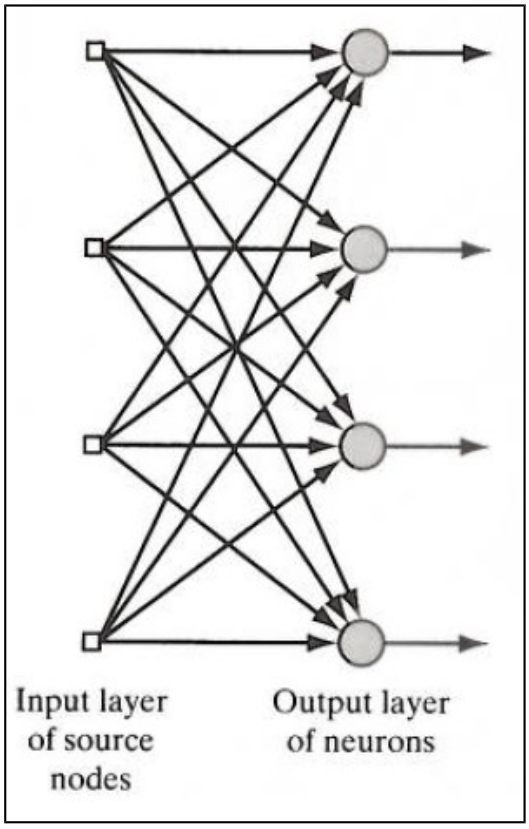
2.1.1.5 Arsitektur

Menurut Haykin (2009), secara umum, ada tiga jenis arsitektur dari Jaringan Saraf Tiruan yaitu:

1. *Single-Layer Feedforward Networks*

   Di dalam Jaringan Saraf Tiruan dengan satu *layer*, neuron - neuron diorganisasi dalam bentuk *layer* - *layer*. Dalam bentuk paling sederhana dari Jaringan Saraf Tiruan dengan satu *layer*, kita mempunyai

sebuah *input layer* dari node sumber di mana informasi diproyeksikan ke *output layer* dari neuron tapi tidak bisa sebaliknya. Dengan kata lain, jaringan ini adalah tipe *feedforward*. *Input layer* dari node sumber tidak dihitung karena tidak ada perhitungan yang dilakukan

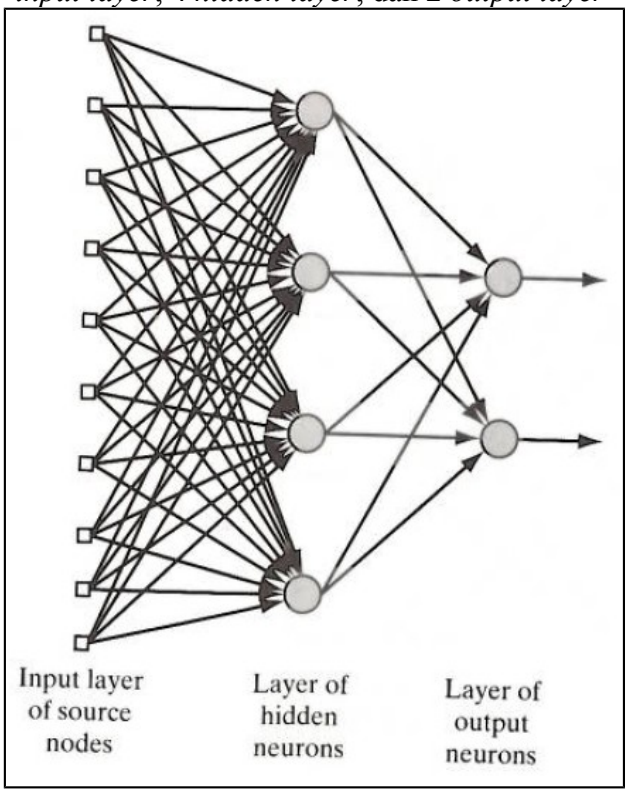


Gambar 2.4 *Single-Layer Feedforward Networks*

2. *Multilayer Feedforward Network*

Jenis kedua dari Jaringan Saraf Tiruan yang bersifat *feedforward* dibedakan dengan adanya keberadaan satu atau lebih *hidden layer*. *Hidden* disini berarti bagian dari Jaringan Saraf Tiruan ini secara langsung tidak terlihat dari *input* atau *output* dari jaringan tersebut. Fungsi dari *hidden layer* adalah untuk mengintervensi antara *input* eksternal dan *output* dari jaringan dalam cara yang berguna. Dengan menambah satu atau lebih *hidden layer*, jaringan dapat mengeluarkan statistik tingkat tinggi dari *input*.

Sumber node di *input layer* dari jaringan menyediakan masing-masing elemen dari pola aktivasi (*vectorinput*), yang merupakan sinyal *input* yang diaplikasikan ke neuron-neuron di *layer* kedua (*hiddenlayer* pertama). Sinyal *output* dari *layer* kedua digunakan sebagai *input-input* ke *layer* ketiga, dan seterusnya sampai ke sisa dari jaringan. Gambar di bawah menunjukkan *multi-layer network* dengan 10 *input layer*, 4 *hidden layer*, dan 2 *output layer*



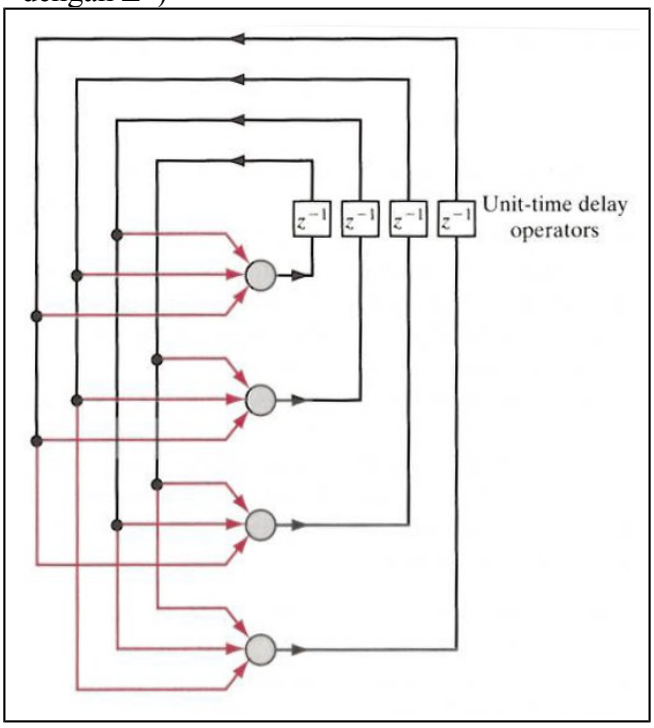
Gambar 2.5 *Multilayer Feedforward Network*

3. *Recurrent Networks*

Jaringan Saraf Tiruan *Recurrent* berbeda dari Jaringan Saraf Tiruan *Feedforward* lainnya dengan mempunyai paling tidak satu *feedback loop*. Contohnya, sebuah jaringan *recurrent* bisa terdiri dari neuron single *layer* dengan setiap neuron memberi sinyal *output* kembali ke *input* dari neuron-neuron lainnya.

Adanya *feedback loop* mempunyai pengaruh yang sangat besar kepada kemampuan pembelajaran

dari jaringan dan juga performanya. *Feedback loop* melibatkan penggunaaan dari cabang-cabang yang terbentuk dari elemen *unit-time delay* (dilambangkan dengan Z^{-1})



Gambar 2.6 *Recurrent Networks*

2.1.1.6 Algoritma *Backpropagation*

Algoritma *Backpropagation* untuk melakukan *training* terhadap suatu jaringan terdiri dari tiga tahap, yaitu *feedforward* dari pola *input training*, *backpropagation* dari error yang terkait, dan penyesuaian bobot.

Langkah-langkah dalam algoritma *backpropagation* oleh Fausett (1994) adalah sebagai berikut

Langkah 0: Inisialisasi bobot (set bobot pada nilai random yang kecil).

Langkah 1: Ketika pada kondisi berhenti salah, lakukan langkah 2 – 9.

Langkah 2: Untuk setiap pasangan *training,* lakukan langkah 3 – 8.

Feedforward

Langkah 3: Setiap unit *input* (X_i, $i = 1, \ldots, n$) menerima sinyal *input* X_i dan memancarkan sinyal ini

kepada semua unit pada lapisan diatasnya (*hidden unit*)

Langkah 4: Setiap *hidden unit* (Z_j,j=*1,. . . ,p*)

menjumlahkan bobot sinyal *input*.

$$z\_in_j = V_{0j} + \sum_{i=1}^{n} X_i V_{ij}$$

mengaplikasikan fungsi aktivasi untuk menghitung sinyal *output*

$$z_j = f(z\_in_j)$$

dan mengirim sinyal ke semua unit di lapisan di atasnya (*output unit*).

Langkah 5: Setiap unit *output* (Y_k, k = *1, . . . , m*) menjumlahkan bobot sinyal *input*.

$$y\_in_k = W_{0k} + \sum_{j=1}^{v} Z_j W_{jk}$$

dan mengaplikasikan fungsi aktivasinya untuk menghitung sinyal *output*.

$$y_k = f(y\_in_k)$$

Backpropagation

Langkah 6: Setiap unit *output* (Y_k, k = *1, . .. , m*) menerima pola target sesuai dengan pola *training input*, menghitung informasi *error*.

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

menghitung koreksi bobotnya (digunakan untuk memperbaharui w_{jk})

$$\Delta w_{jk} = \alpha \delta_k z_j$$

menghitung koreksi bias (digunakan untuk memperbaharui w_{0k})

$$\Delta w_{0k} = \alpha \delta_k$$

mengirim δ_k ke unit lapisan dibawahnya

Langkah 7: Setiap *hidden unit* (Z_j , j = *1, . . . , p*) menjumlahkan delta *input* (dari unit di lapisan

atas).

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk}$$

dikalikan dengan turunan dari fungsi aktivasi untuk menghitung informasi *error*.

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

menghitung koreksi bobot (digunakan untuk memperbaharui v_{ij})

$$\Delta v_{ij} = \alpha \delta_j x_i$$

dan menghitung koreksi bias (digunakan untuk memperbaharui v_{0j})

$$\Delta v_{0j} = \alpha \delta_j$$

Update bobot dan bias

Langkah 8: Setiap unit *output* (Y_k, $k = 1, \ldots, m$) memperbaharui bias dan bobot (j = 0 , ... , p)

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

Setiap *hidden unit* (Z_j, $j = 1, \ldots, p$) memperbaharui bobot dan bias ($i = 0, \ldots, n$)

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

Langkah 9: Tes kondisi berhenti.

Algoritma aplikasi

Setelah *training,* jaringan saraf *backpropagation* diaplikasikan dengan hanya menggunakan fase *feedforward* dari algoritma *training*. Langkah-langkahnya sebagai berikut oleh Fausett (1994):

Langkah 0: Inisialisasi bobot (dari algoritma *training*).
Langkah 1: Untuk setiap vektor *input* lakukan langkah 2-4.
Langkah 2: Untuk $i = 1, \ldots, n$ set aktivasi untuk unit *input* x_i.

Feedforward

Langkah 3: Untuk setiap $j = 1, \ldots, p$

$$z\_in_j = V_{0j} + \sum_{i=1}^{n} X_i V_{ij}$$

$$z_j = f(z\_in_j)$$

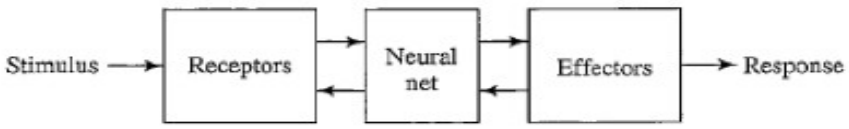
Langkah 4: Untuk setiap $k = 1, \ldots, m$

$$y\_in_k = W_{0k} + \sum_{j=1}^{v} Z_j W_{jk}$$

$$y_k = f(y\_in_k)$$

2.1.2 *Human Brain* (otak manusia)

Oleh Haykin (2009), diberikan penjelasan mengenai detil jaringan saraf dalam otak manusia yang dipakai sebagai model Jaringan Saraf Tiruan. Sistem saraf manusia dibagi dalam tiga tahapan, seperti yang digambarkan pada blok diagram dari gambar 2.7. Pusat sistem adalah otak, diwakili oleh jaringan saraf, yang terus menerima informasi, merasakannya, dan membuat keputusan yang sesuai. Rangsangan yang diterima akan dikirimkan ke dalam sistem melalui *forward transmission* yang berguna untuk menyampaikan informasi berupa sinyal, lalu diubah oleh reseptor (*receptors*) menjadi impuls listrik (*electrical impulses*) yang membawa informasi ke jaringan saraf (otak). Sedangkan efektor (*effectors*) mengkonversi impuls listrik yang sudah dihasilkan oleh jaringan saraf menjadi respon yang dapat dilihat sebagai *output* sistem.



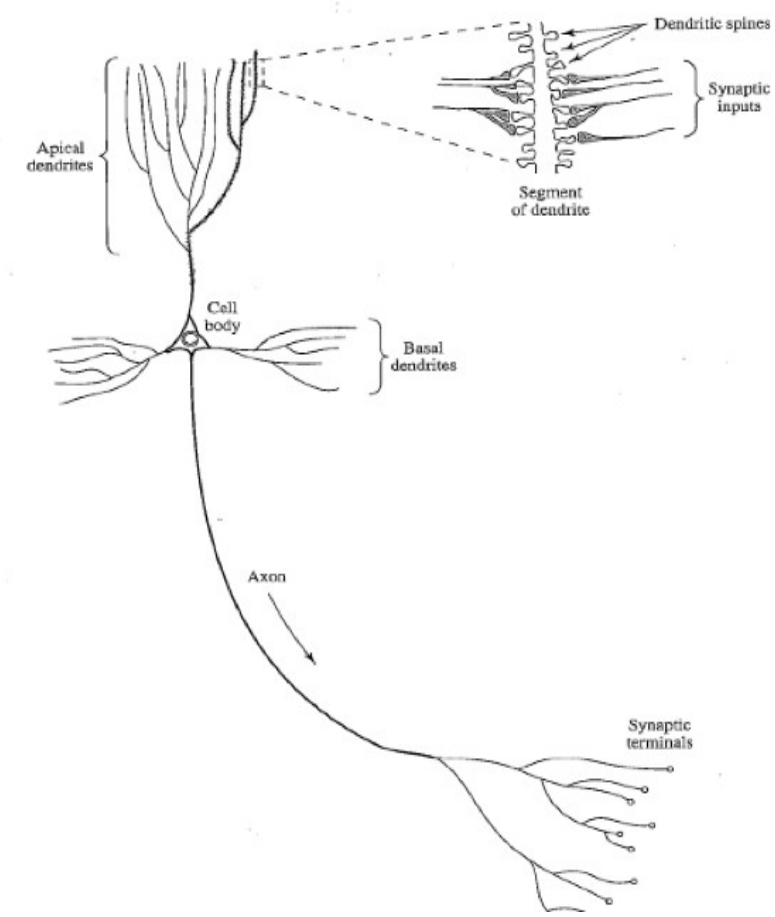
Gambar 2.7 Tiga Tahap Sistem Saraf Manusia

Untuk mempermudah dalam mempelajari otak, Ramon y Cajal

(1911) menyatakan neuron sebagai komponen otak. Karena proses neuron yang lambat, otak menutupi kekurangan itu dengan mempunyai neuron dalam jumlah yang banyak dan interkoneksi yang sangat besar antar neuron. Diperkirakan terdapat sekitar 10 milyar neuron pada korteks manusia (*human cortex*) dan 60 triliun sinapsis atau koneksi.

Sinapsis atau ujung saraf, adalah struktur dasar dan unit fungsional yang menengahi interaksi antar neuron. Jenis sinapsis yang paling umum adalah chemical synapse, yang beroperasi sebagai berikut: *Presynaptic* melepaskan pemancar substansi yang berdifusi melintasi persimpangan sinaptik antara neuron dan kemudian bertindak pada proses *postsynaptic*. Demikian sinapsis mengubah *presynaptic* sinyal elektrik (*electrical signal*) ke sinyal kimia (*chemical signal*) dan kemudian kembali lagi ke *post synaptic* dalam bentuk sinyal elektrik.

Pada otak orang dewasa, akson, jalur transmisi, dan dendrit, zona reseptif merupakan dua jenis filamen sel yang dibedakan atas dasar morfologi. Akson memiliki permukaan yang halus, sedikit cabang, dan lebih panjang, sedangkan dendrit mempunyai permukaan yang tidak biasa, dan memiliki banyak cabang (Freeman, 1975). Ukuran dan bentuk neuron berbeda-beda menurut letaknya di otak. Pada gambar 2.8 diilustrasikan bentuk dari sel piramida (*pyramidal cell*), yang merupakan salah satu jenis yang paling umum dari neuron kortikal. Seperti neuron lainnya, *input* yang diterima berasal dari ujung dendrit. Sel piramida dapat menerima 10,000 atau lebih kontak sinaptik, dan dapat memproyeksikannya ke ribuan sel target.

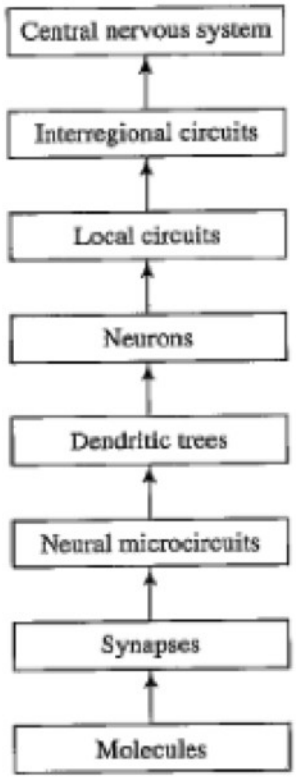


Gambar 2.8 *Pyramidal Cell* (Sel Piramida)

Mayoritas neuron menerjemahkan *output*nya sebagai rangkaian tegangan pulsa singkat yang biasa dikenal sebagai *action potentials* atau *spike* yang dekat atau terletak pada badan sel neuron dan lalu merambat melintasi neuron individu pada kecepatan konstan dan amplitudo. Karena fisik akson yang tipis, sangat panjang, dan kapasitasnya yang tinggi, maka *action potentials* sangat dibutuhkan untuk komunikasi antar neuron.

Pada otak, terdapat dua skala organisasi anatomi, yaitu skala kecil dan skala besar, dan perbedaan fungsi pada tingkat yang rendah dan yang lebih tinggi. Pada gambar 3 terdapat hirarki tingkat jalinan organisasi yang dihasilkan dari pekerjaan yang luas pada analisis daerah lokal di otak (Sheperd dan Koch, 1990; Churchland and Sejnow-ski, 1992). Sinapsis merupakan tingkat yang paling mendasar, tergantung pada molekul dan ion. Pada tingkat selanjutnya, terdapat *neural microcircuits*, *dendritic trees* (pohon dendrit), dan lalu neuron.

Neural microcircuit mengacu pada perakitan sinapsis yang diatur dalam pola konektivitas untuk menghasilkan operasi fungsional yang penting. *Neural microcircuit* berkumpul untuk membentuk *dendritic subunits* dalam *dendritic trees* pada neuron individu. Pada tingkat kompleksitas berikutnya, terdapat *local circuits* yang terbuat dari neuron-neuron yang sama atau berbeda. Setelah itu terdapat *interregional circuits* yang terdiri dari jalur-jalur (*pathways*), kolum-kolum (*columns*), dan peta topografi (*topographic maps*), yang melibatkan beberapa daerah yang terletak di berbagai bagian otak.



Gambar 2.9 Hierarki Tingkat Jaringan Pada Otak

Topographic maps disusun untuk menanggapi informasi sensorik yang masuk, biasanya dalam bentuk lembaran, seperti di *superior colliculus*. Pada level kompleksitas terakhir, *topographic maps* dan *interregional circuit* (sirkuit antar daerah) lainnya menengahi jenis perilaku tertentu dalam *central neurvous system* (sistem saraf pusat).

2.1.3 *Forecasting* (Prediksi /Peramalan)

Menurut Marshall dan Oliver (1995), peramalan adalah sesuatu yang harus dilakukan jika ingin merencanakan untuk masa depan. Kebanyakan Peramalan menggunakan data masa lalu untuk mengidentifikasi trend jangka pendek, menengah ataupun panjang dan menggunakan pola-pola ini untuk memproyeksikan posisi sekarang untuk masa yang akan datang

Peran dari Peramalan ini adalah untuk menggunakan semua informasi yang tersedia dan saling berhubungan untuk mempertajam pengetahuan dan mengurangi ketidakpastian dari hasil pengukuran yang mempengaruhi keputusan dan hasil akhir. Informasi bisa dalam bentuk pengamatan data yang lalu dari beberapa variabel tambahan yang mempengaruhi atau dipengaruhi dari nilai pengukuran yang digunakan pada perumusan masalah

Menurut Edward dan Finlay (1997), untuk meramal suatu nilai berdasarkan data yang lalu, maka digunakan teknik *time series analysis. Time series* adalah kumpulan hasil observasi dari suatu nilai sebuah variabel yang membentuk suatu rangkaian waktu. Pendekatan dari teknik ini ada dua, salah satunya melakukan peramalan dengan mengidentifikasi pola dari data yang lalu. Cara yang lain adalah dengan melakukan peramalan secara langsung dari nilai-nilai dari suatu data.

Filosofi dari pendekatan berdasarkan pola dari masa lalu adalah:

1. Masa yang akan datang akan sama seperti masa lalu.
2. Laju perubahan di masa yang akan datang akan sama seperti masa lalu.
3. Perubahan pada laju perubahan di masa yang akan datang akan sama seperti masa lalu.

Tujuannya adalah untuk mengidentifikasi pola yang cukup kompleks untuk memproduksi peramalan yang diterima, tetapi tidak terlalu kompleks daripada yang dibutuhkan.

Pendekatan berdasarkan nilai-nilai data yang memusatkan lebih pada pekerjaan dengan nilai yang lalu. Gagasan ini adalah

melakukan peramalan dengan memperbaharui data dengan data yang baru segera setelah data itu tersedia.

2.1.4 Nilai Tukar

Menurut Copeland (2008), cara mudah mengerti nilai tukar adalah harga. Ketika berhadapan dengan nilai tukar, muncul fakta bahwa dalam pasar mata uang banyak transaksi yang melibatkan perantara yang bertindak sebagai pembeli sementara untuk agen yang ingin menjual barangnya dan sebaliknya bagi yang ingin membeli. Dalam beberapa kasus, mereka memberikan biaya tambahan atau komisi untuk jasa tersebut agar pembeli dan penjual sepakat. Namun dalam kebanyakan transaksi, sumber dari penghasilan mereka berada pada selisih antara harga di mana mereka membeli mata uang tersebut dan harga mereka menjual. Ada kalimat khusus untuk menjelaskan hal ini oleh Copeland (2008:8)

> *"The **bid rate** for currency A in terms of currency B is the rate at which dealers buy currency A (sell currency B). The **offer** (or **ask**) **rate** is the rate at which dealers sell currency A (buy currency B). The (**bid**/**ask**) **spread** is the gap between the offer and bid rates."*

Seperti harga-harga barang, harga nilai tukar ditentukan oleh jumlah penawaran dan permintaan. Jika diibaratkan dengan dolar dan rupiah, maka semua penawaran dari dolar adalah sama dengan permintaan kepada rupiah dan sebaliknya.

Ada beberapa sistem tukar yang dikenal salah satunya adalah sistem nilai tukar mengambang (*floating exchange rate*). Sistem nilai tukar mengambang merupakan sistem nilai tukar yang tingkatnya ditentukan secara eksklusif oleh keseimbangan yang mendasari penawaran dan permintaan untuk mata uang yang terlibat tanpa campur tangan dari luar. Jika permintaan dolar bertambah dan penawaran rupiah bertambah maka akan membuat rupiah turun hingga posisi di bawah normal. Membuat bertambahnya penukaran dolar menjadi rupiah daripada membeli dolar dengan rupiah. Dan kalau permintaan rupiah bertambah dan penawaran dolar juga

bertambah maka akan membuah harga rupiah naik di atas harga normal membuat bertambahnya pembelian dolar dengan rupiah.

2.1.5 *Unified Modelling Language* (*UML*)

Whitten dan Bentley (2007) menjelaskan dalam bukunya beberapa bentuk diagram UML, antara lain :

2.1.5.1 Use Case

Use case diagram adalah diagram yang menggambarkan interaksi antara sistem dan pengguna. Dengan kata lain, secara grafis menggambarkan siapa yang menggunakan sistem dan bagaimana interaksinya dengan sistem.

Bagian utama dalam *use case* ada 2 yaitu:

1. *Actor*
   Use case dimulai oleh pengguna eksternal dinamakan aktor. Seorang aktor memulai aktivitas pada sistem dengan tujuan menyelesaikan suatu tugas yang menghasilkan sesuatu. Aktor digambarkan dengan *stick figure* dengan penamaan sesuai peran dari aktor.
2. *Relationship*
   Relationship digambarkan sebagai garis antara dua simbol di *use case diagram*. *Relationship* digunakan untuk menggambarkan hubungan interaksi yang terjadi.

2.1.5.2 *Sequence Diagram*

Sequence Diagram atau yang disebut juga *system sequence diagram* adalah diagram yang menggambarkan interaksi antara aktor dan sistem pada skenario *use case*. *Sequence diagram* membantu kita mengidentifikasi *high-level messages* yang masuk dan keluar dari sistem. Pesan ini dikirimkan antar objek untuk saling berkomunikasi.

Notasi dari *Sequence Diagram*

1. *Actor* – aktor digambarkan dengan simbol aktor dari *use case*.

2. *System* – kotak menggambarkan sistem sebagai "*black box*" atau secara keseluruhan. Titik dua (:) adalah notasi standar *sequence diagram* untuk menunjukkan "*instance*" yang berjalan dari sistem.
3. *Lifelines* – garis *vertical* putus-putus yang memanjang ke bawah dari simbol aktor dan sistem menunjukkan riwayat dari *sequence*.
4. *Activation bars* – batang yang diletakkan di atas *lifeline* menunjukkan periode waktu ketika peserta aktif dalam interaksi
5. *Input messages* – panah *horizontal* dari aktor ke sistem menunjukkan pesan input. Konvensi pesan di UML adalah dengan memulai kata pertama dengan huruf kecil dan kata tambahan dengan huruf besar tanpa spasi. Parameter dipisahkan dengan tanda koma.
6. *Output messages* – panah horizontal dari sistem ke aktor dengan garis putus-putus.

Panduan menggambar *sequence diagram*

1. Pilih *use case* mana yang akan digambarkan *sequence diagram*-nya.
2. Gambar persegi panjang untuk menunjukkan sistem secara keseluruhan dan tambahkan *lifeline* dibawahnya.
3. Identifikasi setiap aktor yang secara langsung menyediakan *input* ke sistem atau secara langsung menerima *output* dari sistem. Tambahkan *lifeline* di bawah aktor.
4. Periksa *use case narrative* untuk mengenali input dan output sistem. Abaikan message di dalam sistem. Gambarkan pesan eksternal dengan panah horizontal dari *lifeline* aktor ke sistem atau sebaliknya. Namakan input sesusai penamaan UML.
5. Tambahkan *frame* untuk menunjukkan pesan opsional dengan kondisi.
6. Pastikan pesan yang ditampilkan dalam urutan yang benar.

2.1.5.3 *Activity Diagram*

Activity Diagram adalah diagram yang digunakan secara grafik menggambarkan aliran dari proses, langkah – langkah *use case* atau logika dari perilaku objek (metode). Mereka mirip dengan *flowchart* namun berbeda pada mekanisme yang disediakan untuk menggambarkan aktivitas yang terjadi secara parallel. Oleh karena itu mereka sangat berguna untuk memodelkan *action* yang akan dilakukan ketika suatu operasi dilakukan maupun hasil dari tindakan tersebut. *Activity diagram* fleksibel karena mereka dapat digunakan pada saat desain dan analisis.

Notasi dari *activity diagram*

1. *Initial node* – lingkaran *solid* yang menggambarkan permulaan dari proses.
2. *Actions – rounded rectangle* yang menggambarkan langkah individual.
3. *Flow* – panah pada diagram menunjukkan. Kebanyakan *flow* tidak membutuhkan kata-kata untuk menjelaskan mereka kecuali datang dari *decision.*
4. *Decision* – bentuk *diamond* dengan satu *flow* masuk dan dua atau lebih *flow* keluar. *Flow* yang keluar menandakan kondisi.
5. *Merge* – bentuk *diamond* dengan dua atau lebih *flow* keluar dan satu *flow* masuk. Kombinasi *flow* yang dipisahkan oleh *decision*. Proses berlanjut dengan *flow* yang mana saja yang masuk ke *merge*.
6. *Fork* – batang hitam dengan satu *flow* masuk dan dua atau lebih *flow* keluar. *Action* dalam *flow* parallel di bawah fork dapat terjadi pada urutan apapun atau berbarengan.
7. *Join* – batang hitam dengan dua atau lebih *flow* masuk dan satu *flow* keluar, di akhir proses yang berbarengan. Semua *action* yang datang ke *join* harus lengkap sebelum proses berlanjut.

8. *Activity final* – lingkaran *solid* di dalam lingkaran *hollow* (kosong) menggambarkan akhir dari proses.

Panduan untuk menggambar *activity diagram*

1. Mulai dengan *initial node* sebagai titik awal
2. Tambahkan partisi jika sesuai dengan analisis
3. Tambahkan *action* untuk setiap langkah di *use case* (atau setiap langkah yang dilakukan aktor)
4. Tambahkan *flow* dari setiap *action* ke *action* lainnya, *decision point*, atau *end point*. Untuk presisi maksimum dari arti, setiap *action* hanya boleh ada satu aliran masuk dan satu aliran keluar dengan semua *forks, joins, decisions* dan *merges* ditampilkan secara eksplisit.
5. Tambahkan *decisions* dimana *flow* menyebar ke rute yang berbeda. Jangan lupa untuk menyambungkan kembali dengan *merge.*
6. Tambahkan *fork* dan *joins* dimana aktivitas dijalankan parallel.
7. Akhiri dengan satu notasi dengan *activity final*

2.2 *Related Works*

2.2.1 *Stock Price Prediction using Neural Network with Hybridized Market Indicators*

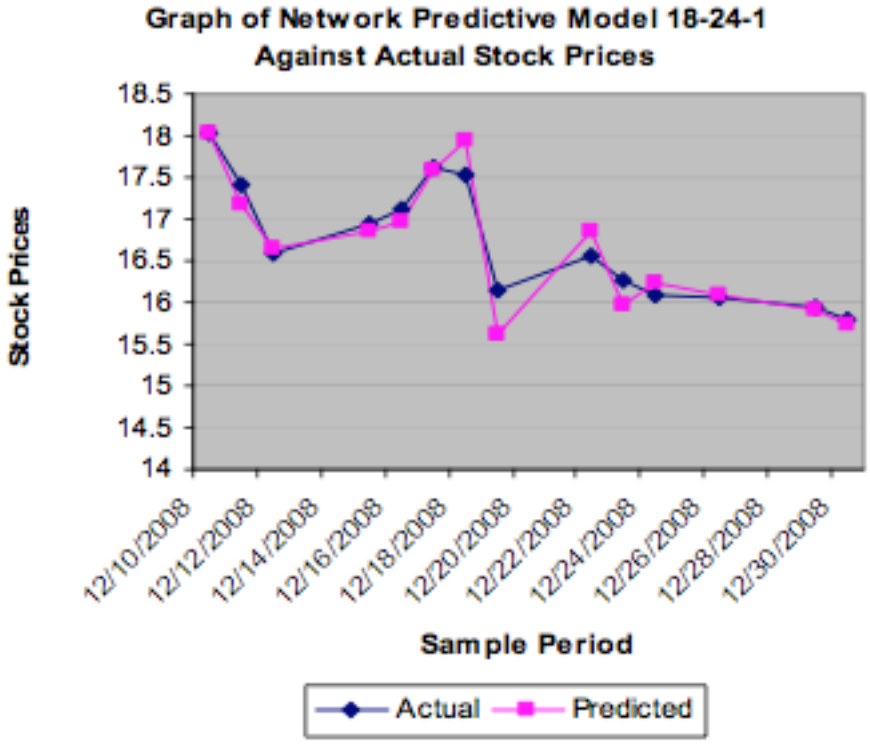
Adebiyi A.A., Ayo C.K. , Adebiyi M.O. , dan Otokiti S.O. menulis jurnal tentang prediksi harga saham menggunakan Jaringan Saraf Tiruan (*Neural Network*) dengan indicator pasar campuran (*hybrid*) yang menggabungkan penggunaan variabel-variabel analisis teknis dan analisis dasar (*fundamental*) dari indikator pasar saham. Jaringan Saraf Tiruan (JST) adalah teknik data *mining* yang dapat belajar dan mendeteksi hubungan antar variabel *non-linear*. JST juga memungkinkan analisis yang lebih dalam pada set data yang besar terutama yang memiliki kecenderungan untuk berfluktuasi dalam periode waktu yang singkat.

Jurnal ini akan membahas tentang perbandingan model JST untuk memprediksi pasar saham dengan menggunakan JST *multilayer*

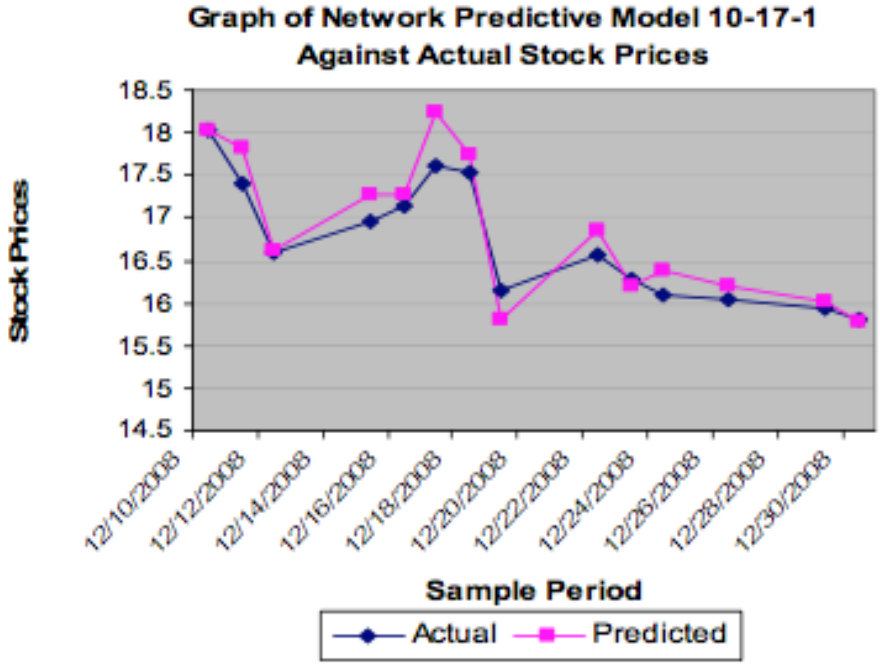
perceptron dengan jumlah 3 lapisan yang terdiri dari *input layer* (x *node*), *hidden layer* (z *node*), dan *output layer* (y *node*). Algoritma pembelajaran (*training*) yang dipakai adalah algoritma *backpropagation* dengan fungsi aktivasi *sigmoid* karena telah ditemukan literatur yang membuktikan bahwa fungsi aktivasi *sigmoid* lebih baik daripada fungsi-fungsi lainnya, seperti *Unit Step function, Gaussian function, Piecewise Linear function,* dan *Binary Transfer function*.

Perbandingan model JST akan dilakukan dengan membandingkan model JST dengan variabel *input* yang berbeda, yaitu model JST *hybrid* (18 – z – 1, z=18, … ,26) dengan variabel *hybrid* (variabel analisis teknis dan analisis fundamental) dan model JST analisis teknis (10 – z – 1, z = 10, … , 18) dengan variabel analisis teknis sebagai variabel *input* masing-masing JST. Variabel analisis teknis adalah indeks inti pasar saham (harga saham saat ini, harga pembukaan, harga penutupan, volume, harga tertinggi, harga terendah, dll), sedangkan variabel analisis fundamental adalah indeks kinerja perusahaan (harga per pendapatan tahunan, rumor/berita, nilai buku, status keuangan, dll).

Hasil perbandingan yang didapat dengan kedua JST diatas dengan jumlah *input node, hidden node* yang berbeda dan 1 *output node* membuktikan bahwa JST *hybrid* lebih akurat daripada JST analisis teknis. Hal ini didapat dengan membandingkan hasil yang paling akurat dari masing-masing JST, yaitu JST *hybrid* dengan jumlah node *input-hidden-output* 18-24-1 (gambar 2.10) dengan JST analisis teknis dengan jumlah node *input-hidden-output* 10-17-1 (gambar 2.11).



Gambar 2.10 Prediksi *neural network* I



Gambar 2.11 Prediksi *neural network* II

Berdasarkan jurnal diatas, dapat disimpulkan bahwa dengan menggabungkan variabel utama dengan variabel-variabel lainnya yang *non-linear* dapat menghasilkan hasil prediksi yang lebih akurat

daripada menggunakan 1 variabel utama saja. Begitu pula dengan tugas akhir ini yang menggabungkan mata uang sebagai variabel utama dan faktor-faktor lainnya seperti inflasi, harga minyak mentah, jumlah uang beredar, dan tingkat suku bunga sebagai variabel lainnya agar dapat memperoleh hasil prediksi yang lebih akurat.

2.2.2 *Artificial Neural Network Model for Forecasting Foreign Exchange Rate*

Dalam jurnal oleh A.A.Philip, A.A.Taofiki, A.A.Bidemi dijelaskan bahwa model statistik saat ini yang digunakan untuk prediksi, tidak bisa menangani secara efektif ketidakpastian dan ketidakstabilan dari data pertukaran mata uang. AFERFM (*Artificial Neural Network Foreign Exchange Rate Forecasting Model*) dirancang untuk membetulkan beberapa masalah ini. Dalam beberapa tahun belakangan, Jaringan Saraf Tiruan (*neural network*) berhasil digunakan untuk pemodelan financial time series. Jaringan Saraf Tiruan (JST) adalah fungsi universal yang dapat memetakan setiap fungsi *non-linear* tanpa asumsi dari sifat-sifat data. JST juga lebih noise tolerant, dengan kemampuan untuk mempelajari sistem yang kompleks dengan data yang tidak lengkap atau rusak. Selain itu mereka juga lebih fleksibel, memiliki kemampuan untuk mempelajari sistem dinamis melalui proses *training* kembali dengan data baru.

Desainnya dibagi menjadi dua fase: *training* dan *forecasting*. Dalam fase *training*, *backpropagation* digunakan untuk melakukan *training* nilai tukar mata uang dan mempelajari bagaimana memperkirakan *input*. Fungsi aktivasi *sigmoid* digunakan untuk merubah *input* ke jarak standar [0,1]. Bobot yang digunakan diacak antara [-0.1,0.1] untuk mendapat *input* yang konsisten dengan *training*. Salah satu model lain yang digunakan untuk memprediksi nilai tukar mata uang adalah *Hidden Markov Foreign Exchange Rate Forecast Model* (HFERFM).

Dalam *training,* umumnya meliputi pemberian sampel *training* sebagai vektor *input* melalui JST*,* menghitung kesalahan dari *output layer*, kemudian menyesuaikan bobot dari jaringan untuk

meminimalisasi kesalahan. Proses menentukan besarnya perubahan dari faktor bobot yang menghasilkan *output* akurat adalah yang disebut sebagai *training*. Dalam backpropagation, penyesuaian bobot terus dilakukan sampai kesalahan (error) lebih kecil dari batas yang diinginkan di mana jaringan dianggap sudah terlatih.

Algoritma yang digunakan untuk menyebarkan pembetulan kesalahan (error) adalah:

$$W_{ij}(\text{baru}) - W_{ij}(\text{lama}) = \frac{\partial E}{-\Delta \partial W_{ij}}$$

Di mana W adalah bobot, E adalah parameter *error*, Δ adalah faktor proportional yang disebut *learning rate.*

Dari sejumlah tes yang telah dilakukan, kesimpulan yang didapat, akurasi dari AFERFM adalah 81.2% lebih baik dibanding dengan menggunakan HFERFM dengan akurasi 69.9%. Tingkat perubahan dari AFERFM lebih pasti dan tidak berubah-rubah dibanding dengan HFERFM.

Perbedaan di antara metode yang dipakai dalam tugas akhir ini dengan metode di dalam jurnal ini adalah penggunaan faktor-faktor lain yang mempengaruhi ditambah nilai tukar mata uang dalam kurun waktu tertentu sebagai *input* sedangkan metode dalam jurnal ini hanya memakai nilai tukar mata uang sebagai *input*. Dengan ini diharapkan, metode kami dapat menghasilkan akurasi yang lebih baik.

2.2.3 *A Network Model for Time-Series Forecasting*

Kemudian satu jurnal menarik dari Nicolae Morariu, Eugenia Iancu, Sorin Vlad berjudul *"A Network Model for Time-series Forecasting"* mereka berpendapat bahwa pengkajian atas tingkat pengembangan untuk suatu aktivitas tertentu dapat dilakukan dengan menggunakan analisis evolusi indikator yang menggambarkan baik tingkat kuantitatif serta mutasi kualitatif secara bersamaan.

Jika diperhatikan karakteristik sederet data x_1, x_2, x_3 …, x_n, x adalah bentuk yang didefinisikan oleh karakteristik tersebut. Kemudian bentuk x dapat dianggap sebagai vektor x (x_1, x_2, x_3 …, x_n).

Parameter sebanyak n dari vektor x diberikan metode pengolahan yang berbeda: normalisasi data, transformasi linier dan nonlinier, seleksi pola. Jika nilai parameter besarnya berbeda, maka parameter dengan nilai absolut yang tinggi akan memiliki pengaruh yang lebih besar atas hasil klasifikasi dan nilai parameter harus memiliki urutan yang sama besarnya. Metode yang sering digunakan adalah metode penyesuaian domain. Transformasi dari nilai parameter berikut yang diterapkan adalah:

$$X_{i,new} = \frac{(X_{i,old} - X_{i,min})}{X_{i,max} - X_{i,min}}$$

Prediksi adalah masalah sulit dan dapat dilihat sebagai pengenalan pola sementara yang sangat cocok diselesaikan dengan JST. Subjek variable perdiksi dapat berbeda dari data sebelumnya (*independent variables*). Metode *backpropagation* dan fungsi aktivasi *sigmoid* adalah hal yang paling penting dan merupakan metode yang digunakan untuk pelatihan *multilayer neural networks feedfoward*.

Dalam jurnal ini terdapat satu metode lagi untuk menyelesaikan masalah prediksi, yaitu analisa regresi. Dalam hal sistem linear, pendekatan regresi memiliki relevansi praktis. Karena kemampuan mereka untuk mendeteksi ketergantungan non-linear dalam *input* set data, jaringan saraf yang merupakan alternatif yang efisien untuk metode yang ada.

Jurnal ini juga mengkonfirmasikan bahwa prediksi untuk periode yang lebih panjang tidak akurat (4 tahun). Untuk mendapatkan hasil 4 tahun kedepan disarankan memprediksi secara bertahap 1 tahun, 2 tahun, 3 tahun ke depan. Berdasarkan jurnal ini prediksi memang cocok untuk diselesaikan dengan JST. Oleh karena itu, lebih baik prediksi dilakukan secara bertahap hingga mencapai waktu yang diinginkan dengan mengisi kekosongan data pada tahap-tahap sebelumnya yang dibutuhkan untuk prediksi selanjutnya agar menghasilkan prediksi yang akurat pada hasil akhir.