UNIVERSITAS INDONESIA

PENGEMBANGAN ALGORITMA *RADIAL BASIS FUNCTION* DENGAN FUNGSI *ERROR CROSS-ENTROPY* PADA JARINGAN SARAF TIRUAN TUNGGAL DAN *ENSEMBLE* SERTA PERBANDINGANNYA DENGAN *BACKPROPAGATION*

SKRIPSI

DIMAS ADITYAMURTHI
0706267641

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2011

UNIVERSITAS INDONESIA

PENGEMBANGAN ALGORITMA *RADIAL BASIS FUNCTION* DENGAN FUNGSI *ERROR CROSS-ENTROPY* PADA JARINGAN SARAF TIRUAN TUNGGAL DAN *ENSEMBLE* SERTA PERBANDINGANNYA DENGAN *BACKPROPAGATION*

SKRIPSI

Diajukan sebagai salah satu syarat dalam memperoleh gelar Sarjana Teknik

DIMAS ADITYAMURTHI
0706267641

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2011

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Dimas Adityamurthi**
NPM	**: 0706267641**
Tanda Tangan	**:** 
Tanggal	**: 15 Juni 2011**

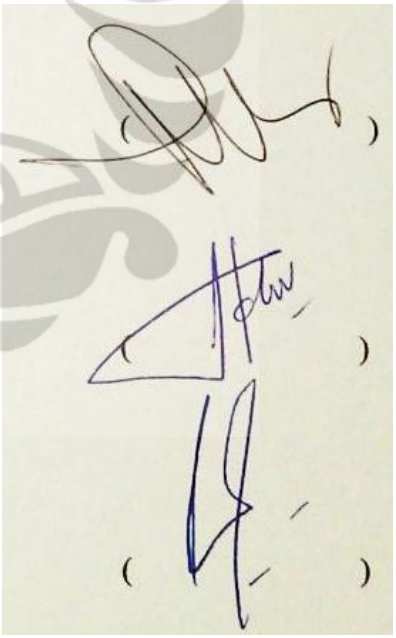
HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama : Dimas Adityamurthi
NPM : 0706267641
Program Studi : Teknik Elektro
Judul Skripsi : Pengembangan Algoritma *Radial Basis Function* dengan Fungsi *Error Cross-Entropy* pada Jaringan Saraf Tiruan Tunggal dan *Ensemble* serta Perbandingnnya dengan *Backpropagation*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI



Pembimbing :
Prof. Dr.Eng. Drs. Benyamin Kusumoputro M.Eng. ( )

Penguji 1 :
Dr. Abdul Halim, M.Eng. ( )

Penguji 2 :
Ir. Aries Subiantoro, M.See. ( )

Ditetapkan di : Depok

Tanggal : 4 Juli 2011

KATA PENGANTAR

Puji dan syukur ke hadirat Tuhan Yang Maha Esa, karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan skripsi ini. Skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelas Sarjana Teknik jurusan Teknik Elektro pada Fakultas Teknik Universitas Indonesia.

Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk dapat menyelesaikan skripsi ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1) Prof. Dr. Eng. Drs. Benyamin Kusumoputro M.Eng., selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan penulis dalam penyusunan skripsi ini;
2) orang tua dan keluarga penulis yang telah memberikan bantuan dukungan material dan moral; serta
3) teman-teman penulis yang tidak dapat disebutkan satu per satu yang telah membantu penulis, baik secara aktif ataupun pasif, dalam menyelesaikan skripsi ini.

Akhir kata, penulis berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu di masa yang akan datang.

Depok, 15 Juni 2011

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Dimas Adityamurthi
NPM	: 0706267641
Program Studi	: Teknik Elektro
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis Karya	: Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul:

PENGEMBANGAN ALGORITMA *RADIAL BASIS FUNCTION* DENGAN FUNGSI *ERROR CROSS-ENTROPY* PADA JARINGAN SARAF TIRUAN TUNGGAL DAN *ENSEMBLE* SERTA PERBANDINGANNYA DENGAN *BACKPROPAGATION*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/ formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 15 Juni 2011
Yang menyatakan

(Dimas Adityamurthi)

ABSTRAK

Nama : Dimas Adityamurthi
Program Studi : Teknik Elektro
Judul : Pengembangan Algoritma *Radial Basis Function* dengan Fungsi *Error Cross-Entropy* pada Jaringan Saraf Tiruan Tunggal dan *Ensemble* serta Perbandingnnya dengan *Backpropagation*

Jaringan Saraf Tiruan (JST) merupakan suatu model matematis yang dewasa ini banyak digunakan dan algoritma pembelajarannya merupakan hal yang menarik untuk dipelajari. Pada skripsi ini, akan dibahas mengenai JST berbasis *Radial Basis Function* (RBF) dan perbandingannya dengan JST berbasis *backpropagation* yang dewasa ini banyak digunakan. Optimasi JST *ensemble* dengan algoritma *Negative Correlation Learning* (NCL) berbasis RBF juga akan dilakukan pada skripsi ini. Set data yang akan digunakan selama percobaan adalah data UC Irvine Machine Learning Repository (UCI) dan citra manusia cahaya tampak dan infra merah.

Kata kunci:
Jaringan Saraf Tiruan (JST), *Radial Basis Function* (RBF), *backpropagation*, *Negative Correlation Learning* (NCL), JST *ensemble*

ABSTRACT

Name : Dimas Adityamurthi
Study Program : Electrical Engineering
Title : Development of Radial Basis Function Algorithm Using Cross-Entropy Error Function in Single and Ensemble Artificial Neural Network and Its Comparison with Backpropagation Algorithm

Artificial neural network is a mathematical model that nowadays has been applied widely and its learning algorithm has become an interesting object to learn. This thesis is going to discuss artificial neural network based on Radial Basis Function (RBF) and its comparison with backpropagation that has been widely purposed. Afterward, optimation is conducted in term of ensemble artificial neural network with Negative Correlation Learning (NCL) algorithm based on RBF. The databases to use are UC Irvine Machine Learning Repository (UCI) and pictures of human face that are achieved from infra-red and visible-light cameras.

Key words:
Artificial neural network, Radial Basis Function (RBF), backpropagation, Negative Correlation Learning (NCL), ensemble neural network

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

Bab ini menjelaskan latar belakang penelitian yang dilakukan pada skripsi, permasalahan yang diteliti, tujuan penelitian, manfaat penelitian, batasan masalah penelitian, tahapan penelitian, serta sistematika penulisan skripsi.

1.1 Latar Belakang

Dewasa ini, perkembangan teknologi sudah sangat maju. Penggunaannya sangat luas dan tidak dapat dipisahkan dari kehidupan manusia. Pada zaman sekarang, teknologi yang dikembangkan cenderung mengarah kepada teknologi cerdas yang memiliki kemampuan untuk berpikir dan mengambil keputusan layaknya otak manusia. Dalam proses pengembangannya, banyak cara telah diupayakan agar didapatkan teknologi yang secerdas mungkin dengan kemampuan dan performa yang baik. Namun demikian, hingga sekarang teknologi ini terbentur masalah biaya komputasi yang cukup tinggi apabila dibandingkan dengan manfaat yang diperoleh.

Salah satu metode yang digunakan untuk mengembangkan teknologi cerdas adalah dengan menggunakan Jaringan Saraf Tiruan (JST) yang prinsip kerjanya diadaptasi dari cara kerja jaringan saraf biologis pada manusia, dimulai dari bagian penerima rangsang, proses berpikir atau belajar, hingga proses pengambilan keputusan. Untuk memperoleh JST yang baik, maka jaringan ini perlu dilatih dengan sejumlah data untuk dilakukan pembelajaran terhadapnya. Terdapat banyak algoritma yang telah dikembangkan dalam melatih JST, seperti *backpropagation*, Levenberg-Marquardt, *counterpropagation*, dan lain-lain. Algoritma *backpropagation* merupakan algoritma yang paling diminati untuk saat ini karena dianggap paling baik dibandingkan algoritma lainnya dikarenakan memiliki keakuratan yang cukup tinggi, adaptif, dan tahan terhadap gangguan (*noise*). Namun demikian, algoritma *backpropagation* memiliki kekurangan dari segi biaya komputasi yang tinggi dan kompleksitas dari algoritmanya yang membuat algoritma ini sulit diterapkan pada alat-alat sederhana yang tidak memiliki kemampuan komputasi yang tinggi layaknya komputer-komputer.

Pada skripsi ini, akan diperkenalkan algoritma *Radial Basis Function* (RBF) yang merupakan suatu algoritma pelatihan JST dengan memanfaatkan persamaan probabilitas Gaussian yakni fungsi dari jarak antara tiap data dengan vektor nilai tengah untuk setiap kelompok data. Seperti yang akan dibahas, algoritma yang ditawarkan oleh RBF lebih sederhana dibandingkan dengan algoritma *backpropagation* (BP) yang paling umum digunakan sehingga diharapkan dapat memperbaiki kekurangan yang dimiliki oleh algoritma BP.

Pada kehidupan sehari-hari, suatu keputusan yang baik akan diperoleh apabila keputusan tersebut dipilih dari hasil pertimbangan-pertimbangan lebih dari satu orang di mana setiap orang memiliki pengetahuan yang mungkin tidak sama dengan orang lainnya (*diversity of opinion*). Mengacu kepada pemikiran tersebut, pada skripsi ini juga akan dibahas mengenai JST *ensamble* sebagai bentuk optimum performa dari JST dengan memanfaatkan *Negative Correlation Learning* (NCL) sebagai alat bantu pembangkit *diversity of opinion*.

1.2 Pernyataan Masalah

Masalah yang akan dibahas pada laporan skripsi ini adalah penggunaan algoritma RBF pada pelatihan JST untuk memperbaiki kekurangan dari algoritma BP, yakni di bagian waktu komputasi. Selanjutnya akan dilakukan pengembangan JST *ensemble* dengan menggunakan algoritma RBF untuk meningkatkan kemampuan JST dalam hal performa tingkat pengenalan.

1.3 Tujuan Penelitian

Berikut ini adalah tujuan dari penelitian yang ingin dilakukan, yaitu:

1. memahami cara kerja dan performa dari JST dengan algoritma RBF dalam mengenali set data *UC Irvine Machine Learning Repository* (UCI) dan citra wajah manusia;
2. memahami cara kerja JST *ensemble* dengan algoritma RBF dalam mengenali set data UC *Irvine Machine Learning Repository* (UCI) dan citra wajah manusia;

3. memahami pengaruh banyaknya jumlah *ensemble* pada JST yang digunakan terhadap performa dari jaringan tersebut dengan menggunakan algoritma RBF;
4. memahami pengaruh dari fungsi *error* kuadratis terhadap performa dari JST tunggal dan *ensemble* dengan algoritma RBF; serta
5. memahami pengaruh dari fungsi *error cross-entropy* terhadap performa dari JST tunggal dan *ensemble* dengan algoritma RBF.

1.4 Batasan Masalah

Berikut ini adalah batasan masalah yang digunakan pada penelitian ini:

1. Data yang digunakan pada penelitian ini diambil dari pusat data UC *Irvine Machine Learning Repository* (UCI), set data citra wajah cahaya tampak, dan set data citra wajah infra merah dengan mengunakan kamera CCD yang memiliki sensor *near infra red.*
2. Penelitian dilakukan untuk mengetahui performa dari JST dengan algoritma RBF dalam hal tingkat pengenalan dan waktu komputasi yang dibandingkan dengan alagoritma pembelajaran BP untuk kedua jenis fungsi *error* kuadratis dan *cross-entropy*.
3. Penelitian juga dilakukan untuk mengetahui pengaruh dari jumlah *ensemble* yang digunakan terhadap performa JST dengan algoritma RBF dalam hal tingkat pengenalan dan waktu komputasi untuk kedua jenis fungsi *error* kuadratis dan *cross-entropy*.

1.5 Manfaat Penelitian

Berikut manfaat dari penelitian yang dilakukan:

1. Pembaca dapat memahami cara kerja JST dengan algoritma RBF, baik pada jaringan tunggal maupun *ensemble* dengan fungsi *error* kuadratis dan *cross-entropy*.
2. Hasil penelitian dapat dijadikan acuan dalam percobaan selanjutnya dalam hal pelatihan JST dengan algoritma RBF.
3. Hasil penelitian dapat digunakan sebagai algoritma pengendalian pada suatu proses kendali.

1.6 Tahapan Penelitian

Berikut tahapan penelitian yang dilakukan:

1. mengumpulkan set data dari pusat data UC *Irvine Machine Learning Repository* (UCI), yaitu *Balance scale weight and distance*, *Breast cancer*, *BUPA liver disorders*, *Credit card approval*, *Glass detection*, *Hearth deseases*, *Ionosphere*, *Iris plants*, dan *Sonar*. Selain itu juga digunakan set data citra wajah yang diperoleh dari Prof. Dr.Eng. Drs. Benyamin Kusumoputro M.Eng. selaku pembimbing penulis dan set data kedua diperoleh dari laporan skripsi yang disusun oleh Stephen Roy Imantaka, S.T. dengan judul Sistem Pengenal Wajah Berbasis *Neural Network Ensemble* untuk Citra Infra Merah (2010);
2. menormalisasi nilai semua data agar memiliki nilai antara 0 hingga 1. Khusus untuk data wajah, dimensi data juga mengalami pereduksian yang tergabung dalam proses *Principal Component Analysis* (PCA);
3. membagi data menjadi 2 bagian yakni data yang digunakan untuk pelatihan dan data yang digunakan sebagai pengujian sebanyak 50:50;
4. menetapkan target untuk tiap data baik yang digunakan pada bagian pelatihan maupun pengujian;
5. melakukan proses pelatihan terhadap data pelatihan; serta
6. melakukan pengujian pada JST hasil pelatihan dengan menggunakan data pengujian.

1.7 Sistematika Penulisan

Adapun sistematika penulisan yang digunakan pada skripsi ini adalah sebagai berikut:

BAB 1 merupakan bab pendahuluan yang berisiskan latar belakang, pernyataan masalah, tujuan penulisan, batasan masalah, manfaat penulisan, tahapan penelitian, dan sistematika penulisan.

BAB 2 merupakan bab tinjauan pustaka dan metodologi penelitian yang berisikan penjelasan JST pada umumnya, pelatihan JST dengan algoritma BP, pelatihan JST dengan algoritma RBF, NCL, pelatihan JST *ensemble* dengan algoritma RBF, *Principal Component Analysis* (PCA), Nguyen-Widrow, data

percobaan, pengolahan data percobaan, perangkat dan parameter percobaan, dan skema percobaan.

BAB 3 menjelaskan analisis performa JST tunggal dengan algoritma RBF dan perbandingannya dengan algoritma BP yang membahas tujuan percobaan, prosedur percobaan, algoritma program, hasil percobaan, analisis hasil percobaan, dan kesimpulan hasil percobaan.

BAB 4 merupakan bab yang berisi analisis performa JST *ensemble* dengan algoritma RBF yang membahas tujuan percobaan, prosedur percobaan, algoritma program, hasil percobaan, analisis hasil percobaan, dan kesimpulan hasil percobaan.

BAB 5 merupakan bab yang berisi kesimpulan dan saran dari seluruh rangkaian penelitian.

BAB 2
JST TUNGGAL DAN *ENSEMBLE* BERBASIS *RADIAL BASIS FUNCTION* (RBF) SERTA METODOLOGI PERCOBAAN

Bab ini akan menjelaskan mengenai teori dasar jaringan saraf tiruan (JST), pelatihan JST dengan algoritma *backpropagation* (BP), pelatihan JST dengan algoritma *Radial Basis Function* (RBF), *Negative Correlation Learning* (NCL), pelatihan*ensemble* JST dengan algoritma RBF, Principal Component Analysis (PCA), metodeinisialisasi Nguyen-Widrow, data percobaan, pengolahan awal data percobaan, perangkan dan parameter percobaan, dan skema percobaan.

2.1 Jaringan Saraf Tiruan (JST)

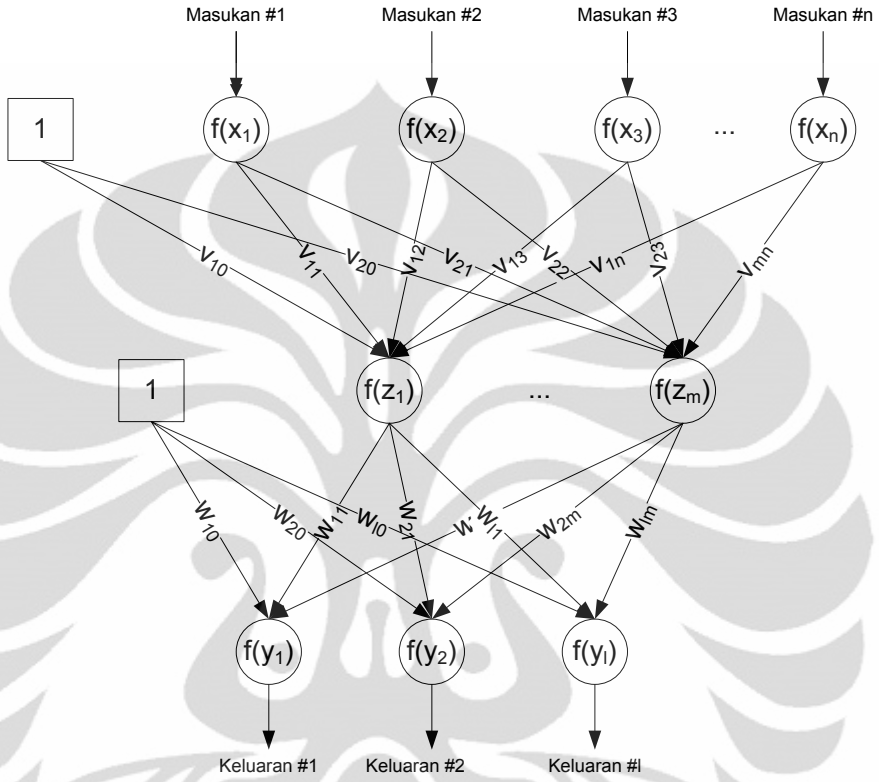
Artificial Neural Network atau Jaringan Saraf Tiruan (JST) adalah sistem pemroses infomasi yang memiliki karakteristik menyerupai jaringan syaraf pada manusia. Layaknya sistem saraf manusia, jaringan ini berfungsi untuk mengenali suatu pola atau memetakan suatu masukan menjadi keluaran yang dilatih melalui suatu proses pelatihan. Dengan kata lain, JST merupakan suatu model matematis non-linear.

JST ini merupakan generalisasi dari pemodelan matematis dalam proses kognitif berdasarkan asumsi:

1. pemrosesan informasi terjadi pada elemen sederhana yang dinamakan neuron;
2. sinyal antar neuron berhubungan melalui saluran penghubung;
3. setiap saluran penghubung mempunyai nilai bobot, dan melakukan operasi perkalian dengan sinyal yang ditranmisikan; serta
4. setiap neuron memberlakukan fungsi aktivasi (biasanya tidak linier) pada masukan total untuk mendapatkan sinyal keluarannya.

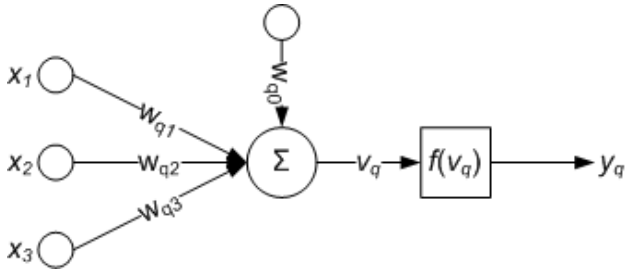
Dengan mengacu pada proses alam, maka JST dirancang seperti pada gambar 2.1.Secara umum, JST memiliki 3 lapisan yakni lapisan masukan, tersembunyi, dan keluaran di mana tiap lapisan terdiri dari neuron-neuron tempat terjadinya proses matematis. Informasi yang masuk ke lapisan masukan identik dengan informasi yang masuk ke *dendrite* dari indera pada manusia, hasil dari lapisan keluaran identik dengan informasi yang telah diolah sebelum masuk ke

bagian motorik pada manusia, dan bagian di antara informasi masukan dan informasi keluaran identik dengan jaringan saraf pada manusia tempat informasi diolah.



Gambar 2.1. Arsitektur JST

Pada gambar 2.1, antarlapisan terdapat bobot-bobot dan bias. $\boldsymbol{v}_{mn} \in \Re^{M\times N}$ dan $\boldsymbol{v}_{m0} \in \Re^{M\times 1}$ adalah bobot dan bias antara lapisan masukan dengan lapisan tersembunyi serta $\boldsymbol{w}_{lm} \in \Re^{L\times M}$ dan $\boldsymbol{w}_{l0} \in \Re^{L\times 1}$ adalah bobot dan bias antara lapisan tersembunyi dengan lapisan keluaran. Bobot-bobot ini berfungsi sebagai penguat atau pelemah sama seperti jaringan saraf pada makhluk hidup. Bias berguna untuk mempermudah proses komputasi. Di dalam setiap neuron, semua informasi yang masuk akan mengalami proses penjumlahan kemudian dikenakan fungsi aktifasi layaknya *threshold* pada neuron saraf sebenarnya seperti ditunjukkan pada gambar 2.2.



Gambar 2.2. Komputasi di Dalam Neuron

Terdapat beberapa jenis fungsi aktifasi yang umum digunakan. Pemilihan fungsi aktifasi mana yang cocok untuk digunakan disesuaikan dengan kebutuhan dan kondisi data. Namun, fungsi aktifasi yang paling sering digunakan adalah fungsi sigmoid dan tangent sigmoid karena fungsi ini memiliki sifat linear di sekitar titik origin, memiliki nilai saturasi, dan dapat diturunkan pada titik origin.

Tabel 2.1. Fungsi Aktivasi

Nama Fungsi Aktifasi		Fungsi Aktifasi
Linear Function		$y_q = f_{lin}(v_q) = v_q$
Hard Limiter	Threshold Function	$y_q = f_{hl}(v_q) = \begin{cases} 0 & if\ v_q < 0 \\ 1 & if\ v_q \geq 0 \end{cases}$
	Signum	$y_q = f_{shl}(v_q) = \begin{cases} -1 & if\ v_q < 0 \\ 1 & if\ v_q \geq 0 \end{cases}$
Saturating Linear Function	Binary Saturating Linear Function	$y_q = f_{sl}(v_q) = \begin{cases} 0 & if\ v_q < 0 \\ v_q + 0.5 & if\ -1 \leq v_q \leq 1 \\ 1 & if\ v_q > 0 \end{cases}$
	Bipolar Saturating Linear Function	$y_q = f_{ssl}(v_q) = \begin{cases} -1 & if\ v_q < 0 \\ v_q + 0.5 & if\ -1 \leq v_q \leq 1 \\ 1 & if\ v_q > 0 \end{cases}$
Sigmoid	Binary Sigmoid Function	$y_q = f_{bs}(v_q) = \frac{1}{1 + e^{-\alpha v_q}}$
	Hyperbolic Tangent Sigmoid	$y_q = f_{hts}(v_q) = \frac{e^{\alpha v_q} - e^{-\alpha v_q}}{e^{\alpha v_q} + e^{-\alpha v_q}} = \frac{1 - e^{-2\alpha v_q}}{1 + e^{-2\alpha v_q}}$

Pelatihan pada JST pada umumnya terbagi dua, yakni pelatihan yang diarahkan dan pelatihan yang tidak diarahkan. Pelatihan yang diarahkan adalah pelatihan di mana keluaran dari setiap neruron pada JST diarahkan sesuai dengan kebutuhan dengan cara membandingkan keluaran yang dihasilkan JST dengan keluaran yang diinginkan dan menggunakan selisihnya untuk memperbaik

parameter-parameter yang mempenaruhi keputusan. Kebalikannya, pelatihan yang tidak diarahkan adalah pelatihan di mana keluaran dari setiap neuron pada JST tidak dibandingkan dengan keluaran yang diharapkan tetapi membandingkannya dengan keluaran yang mungkin dihasilkan dan menggunakan selisihnya guna memperbaiki parameter-parameter untuk mempertegas keputusan.

Seperti yang telah dijelaskan, selisih antara nilai keluaran JST dengan nilai yang diharapkan maupun nilai yang paling memungkinkan akan digunakan untuk memperbaiki parameter-parameter yang berhubungan dengan pengambilan keputusan. Oleh karena itu, perlu terdapat fungsi yang berguna untuk mencari *error* yang terjadi. Fungsi *error* merupakan fungsi yang digunakan untuk memprediksi *error* yang terjadi antara keluaran JST dan keluaran yang diharapkan. Umumnya, fungsi *error* yang digunakan adalah fungsi Lyapunov atau fungsi kuadratis karena sederhana. Namun demikian, permasalahan yang masih belum bisa diatasi adalah kurang mampunya persamaan ini untuk tidak terjebak di dalam *error* lokal. Persamaan 2.1 menunjukkan hubungan matematis fungsi ini. Fungsi lainnya adalah fungsi *cross-entropy* yang ditunjukkan pada persamaan 2.2. Fungsi *cross-entropy* diklaim memiliki waktu stagnasi yang lebih kecil dibandingkan fungsi *error* kuadratis (Nasr, Badr, & Joun, 2002). Namun demikian, pemakaiannya masih jarang ditemukan.

$$E = \frac{1}{L}\sum_{l=1}^{L}\left(t_l - y_l\right)^2 \tag{2.1}$$

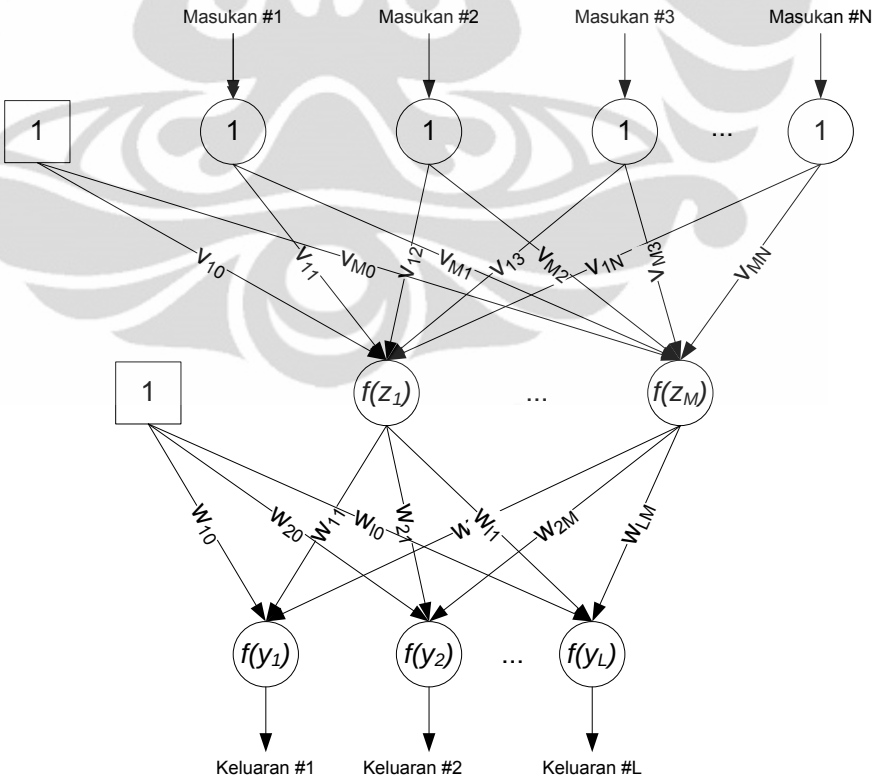
$$E = \frac{1}{L}\sum_{l=1}^{L}\left(-t_l \ln\left(y_l\right) + \left(1-t_l\right)\ln\left(1-y_l\right)\right) \tag{2.2}$$

2.2 Pelatihan JST dengan Algoritma *Backpropagation* (BP)

Metode pelatihan *backpropagation* merupakan salah satu metode *supervised learning* (pembelajaran yang diarahkan) dan menggunakan fungsi *error* untuk memperbaiki bobot-bobot yang ada di antara lapisan keluaran dan tersembunyi serta lapisan tesembunyi dan masukan (Werbos, 1974). Dengan kata lain, apabila terdapat dua set data, yaitu pasangan data masukan dan keluaran, maka algoritma ini akan bekerja dengan cara menyesuaikan bobot-bobot yang ada

dengan cara mempropagasi balik besarnya nilai *error* hingga mendapatkan bobot-bobot yang unik yang dibutuhkan untuk melakukan pemetaan nonlinier terhadap data-data masukan menjadi data-data keluaran yang bersesuaian.

Sebagai ilustrasi terhadap algoritma BP, apabila terdapat satu pasangan data masukan dan target pelatihan yang ingin dilakukan pelatihan agar terbentuk suatu kumpulan bobot-bobot yang dapat memetakan data tersebut. Dengan menggunakan algoritma BP, data akan mengalami *feedforward* (propagasi maju) dari lapisan masukan ke lapisan tersembunyi dan selanjutnya data akan diteruskan dari lapisan tersembunyi menuju lapisan keluaran. *Error* yang dihasilkan dari lapisan keluaran terhadap target pelatihan digunakan untuk memperbaiki bobot-bobot yang ada pada jaringan tersebut dengan cara *backpropagation* (propagasi balik) dari lapisan keluaran menuju lapisan tersembunyi dan dari lapisan tersembunyi menuju lapisan masukan. Begitu seterusnya hingga akhirnya *error* yang dihasilkan dapat ditolerir dan proses pelatihan berhenti sehingga diperoleh suatu set bobot-bobot unik yang dapat memetakan data tersebut.



Gambar 2.3. Arsitektur JST BP

Secara umum, arsitektur JST yang digunakan pada algoritma ini adalah JST *multilayer* (banyak lapis) seperti yang ditunjukkan pada gambar 2.3. Seperti yang telah dibahas sebelumnya, memungkinkan bagi suatu JST untuk hanya memiliki satu buah lapisan. Untuk kasus demikian, maka algoritma BP tetap dapat diterapkan dengan menganggap lapisan tersebut merupakan lapisan keluaran.

Untuk memahami prinsip kerja proses pelatihan JST dengan algoritma BP, berikut ini akan dibahas persamaan matematis yang digunakan pada algoritma ini:

Propagasi maju

Apabila terdapat JST dengan 3 buah lapisan masing-masing adalah lapisan masukan, lapisan tersembunyi, dan lapisan keluaran dengan jumlah neuron pada tiap lapisan, secara berturut-turut sebanyak *N*, *M*, dan *L* seperti yang ditunjukkan pada gambar 2.3, besarnya nilai yang masuk ke lapisan tersembunyi dapat dihitung dengan persamaan 2.3 berikut:

$$z_m = v_{m0} + \sum_{n=1}^{N} v_{mn} \times x_{pn} \tag{2.3}$$

di mana, x_{pn} menunjukkan nilai data dengan indeks ke-*p* dimensi ke-*n*, v_{mn} menunjukkan bobot antara neuron pada lapisan masukan ke-*n* dengan neuron pada lapisan tersembunyi ke-*m*, v_{m0} merupakanbias neuron pada lapisan tersembunyi ke-*m*, dan z_m merupakan nilai yang masuk ke neuron pada lapisan tersembunyi ke-*m*.

Nilai keluaran dari setiap neuron pada lapisan tersembunyi dapat dihitung dengan menggunakan persamaan 2.4 berikut:

$$Z_m = f(z_m) \tag{2.4}$$

di mana $f(z_m)$ menunjukkan fungsi aktifasi untuk setiap nilai masukan pada neuron di lapisan tersembunyi. Nilai yang dihasilkan pada lapisan tersembunyi selanjutnya akan dikirim untuk menjadi masukan pada lapisan keluaran. Nilai yang masuk ke masing-masing neuron pada lapisan keluaran dapat dihitung dengan menggunakan persamaan 2.5 berikut:

$$y_l = w_{l0} + \sum_{m=1}^{M} w_{lm} \times Z_m \tag{2.5}$$

di mana w_{lm} menunjukkan bobot antara neuron pada lapisan tersembunyi ke-*m* dengan neuron pada lapisan keluaran ke-*l*, w_{l0} menunjukkan bias neuron pada lapisan keluaran ke-*l*, dan y_l merupakan nilai yang masuk ke neuron pada lapisan tersembunyi ke-*l*.

Nilai keluaran neuron pada lapisan keluaran dapat dihitung dengan mengunakan persamaan 2.6 berikut:

$$Y_l = f(y_l) \tag{2.6}$$

di mana $f(y_l)$ menunjukkan fungsi aktifasi untuk setiap nilai masukan pada neuron di lapisan keluaran.

Propagasi balik dengan fungsi *error* kuadratis

Untuk setiap data yang masuk ke JST dan selesai melakukan propagasi maju, maka *error* dihitung relatif terhadap target yang bersesuaian dengan persamaan 2.7 berikut:

$$E_p = \frac{1}{2}(\boldsymbol{T}_p - \boldsymbol{Y})^T(\boldsymbol{T}_p - \boldsymbol{Y}) = \frac{1}{2}\sum_{l=1}^{L}(T_{pl} - Y_l)^2 = \frac{1}{2}\sum_{l=1}^{L} e_l^{\,2} \tag{2.7}$$

di mana, $\boldsymbol{T}_p \in \Re^{L\times 1}$ merupakan vektor targetdengan indeks data ke-*p* dan $\boldsymbol{Y} \in \Re^{L\times 1}$ merupakan vektor keluaran dari lapisan keluaran yang diperoleh dari data ke-*p*.

Tujuan dari propagasi balik adalah mengecilkan *error* yang dihasilkan dengan cara mencari nilai bobot antar lapisan yang menghasilkan *error* minimum. Secara matematis, *error* minimum tercapai ketika nilai bobot menghasilkan kemiringan (gradien) 0 pada grafik *error*. Besarnya nilai bobot ini dapat ditentukan dengan memanfaatkan teorema Newton, yakni:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{2.8}$$

Sebelum mencari besarnya nilai bobot v_{mn} dan w_{lm} hasil propagasi balik, akan lebih baik untuk membahas mengenai besarnya nilai turunan suatu fungsi terhadap multi variable. Misalkan terdapat fungsi *error* kuadratis terhadap suatu kumpulan bobot di suatu lapisan dilambangkan dengan ***E(w)*** di mana ***w*** adalah vektor kumpulan bobot-bobot yang hendak diperbaikidan hendak dicari nilai kumpulan bobot yang menghasilkan *error* minimum. Sesuai dengan persamaan

Newton, maka besarnya nilai besarnya kumpulan bobot yang hendak dicari dapat ditentukan dengan menggunakan persamaan 2.9:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{E'(\boldsymbol{w}_k)}{E''(\boldsymbol{w}_k)} \tag{2.9}$$

Secara matematis, nilai turunan pertama dan kedua dari persamaan *error* kuadratis, dapat ditentukan dengan persamaan 2.10 dan 2.11:

$$E'(\boldsymbol{w}) = \boldsymbol{J}_k{}^T \boldsymbol{e}_k \tag{2.10}$$

$$E''(\boldsymbol{w}) = \boldsymbol{J}_k{}^T \boldsymbol{J}_k + \boldsymbol{S}_k \approx \boldsymbol{J}_k{}^T \boldsymbol{J}_k \tag{2.11}$$

dengan

$$\boldsymbol{J}_{\mathrm{k}} = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \cdots & \frac{\partial e_1}{\partial w_{\mathrm{N}}} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \cdots & \frac{\partial e_2}{\partial w_{\mathrm{N}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_L}{\partial w_1} & \frac{\partial e_L}{\partial w_2} & \cdots & \frac{\partial e_L}{\partial w_{\mathrm{N}}} \end{bmatrix} \tag{2.12}$$

$$\boldsymbol{S}_{\mathrm{k}} = \begin{bmatrix} \frac{\partial^2 e_1}{\partial w_1{}^2} & \frac{\partial^2 e_1}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 e_1}{\partial w_1 \partial w_N} \\ \frac{\partial^2 e_2}{\partial w_2 \partial w_1} & \frac{\partial^2 e_2}{\partial w_2{}^2} & \cdots & \frac{\partial^2 e_2}{\partial w_2 \partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 e_L}{\partial w_N \partial w_1} & \frac{\partial^2 e_L}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 e_L}{\partial w_N{}^2} \end{bmatrix} \tag{2.13}$$

di mana $\boldsymbol{J} \in \Re^{L \times N}$ merupakan matriks Jacobian yang isinya berupa kumpulan turunan pertama fungsi *error* terhadap masing-masing bobot dan $\boldsymbol{S}$ merupakan matriks kumpulan turunan kedua fungsi *error* terhadap bobot. *N* menunjukkan banyaknya bobot yang ada di antara lapisan yang hendak diperbaiki. Ketika mendekati nilai minimum, elemen nilai $\boldsymbol{S}$ akan menjadi sangat kecil sehingga dapat diabaikan dan persamaan 2.11 dapat menjadi sederhana.

Dengan mensubsitusikan persamaan 2.10 dan 2.11 ke dalam persamaan 2.9, maka diperoleh persamaan 2.14 yang dikenal sebagai persamaan *Steepest Descent*:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \left(\boldsymbol{J}_k{}^T \boldsymbol{J}_k\right)^{-1} \boldsymbol{J}_k{}^T \boldsymbol{e}_k \tag{2.14}$$

Permasalahan yang dihadapi pada saat merubah nilai bobot secara iterasi dengan persamaan 2.14 adalah perlunya operasi *inverse* dari turunan kedua yang cenderung menjadi matriks singular ketika besarnya nilai *error* mendekati nilai 0. Permasalahan ini dapat diatasi dengan menambahkan suatu konstanta μ, untuk mencegah matriks menjadi singular.

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \left(\boldsymbol{J}_k{}^T \boldsymbol{J}_k + \mu \boldsymbol{I}\right)^{-1} \boldsymbol{J}_k{}^T \boldsymbol{e}_k \tag{2.15}$$

Untuk nilai μ yang mendekati 1, maka besarnya nilai perkalian Jacobian dapat diabaikan dan persamaan 2.15, dapat disederhanakan menjadi persamaan 2.16 dimana α merupakan konstanta pembelajaran.

$$\begin{aligned} \boldsymbol{w}_{k+1} &= \boldsymbol{w}_k - \left(\mu \boldsymbol{I}\right)^{-1} \boldsymbol{J}_k{}^T \boldsymbol{e}_k \\ \boldsymbol{w}_{k+1} &= \boldsymbol{w}_k - \alpha \boldsymbol{J}_k{}^T \boldsymbol{e}_k \end{aligned} \tag{2.16}$$

Persamaan 2.16 merupakan persamaan *Steepest Descent* yang sering digunakan dalam proses pengoptimuman JST. Dari persamaan 2.16, secara harafiah dapat dilihat bahwa besarnya nilai suatu bobot di iterasi sekarang merupakan penjumlahan dari nilai bobot tersebut pada iterasi sebelumnya dikurangi dengan hasil perkalian antara konstanta pembelajaran dengan turunan fungsi *error* terhadap bobot yang hendak diubah.

Dengan mengetahui persamaan *Steepest Descent*, maka persamaan matematis untuk perubahan nilai bobot untuk setiap iterasinya dapat disederhanakan dan mudah untuk dihitung. Sesuai yang telah dijelaskan sebelumnya, proses perubahan bobot dilakukan dari lapisan keluaran menuju lapisan masukan. Untuk bobot antara lapisan keluaran dengan lapisan tersembunyi, besarnya nilai bobot untuk setiap iterasinya dapat dihitung dengan persamaan 2.17 berikut:

$$w_{lm}(k+1) = w_{lm}(k) - \alpha \frac{\partial E_p}{\partial w_{lm}} = w_{lm}(k) - \Delta w_{lm} \tag{2.17}$$

di mana

$$\frac{\partial E_p}{\partial w_{lm}} = \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial y_l} \times \frac{\partial y_l}{\partial w_{lm}} = -\left(T_{pl} - Y_l\right) \times f'\left(y_l\right) \times Z_m \tag{2.18}$$

Untuk bias yang berada diantara lapisan keluaran dan lapisan tersembunyi, besarnya nilai bias untuk setiap iterasinya dapat dihitung dengan menggunakan persamaan 2.19 berikut:

$$w_{l0}(k+1) = w_{l0}(k) - \alpha \frac{\partial E_p}{\partial w_{l0}} = w_{l0}(k) - \Delta w_{l0} \tag{2.19}$$

di mana

$$\frac{\partial E_p}{\partial w_{l0}} = \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial y_l} \times \frac{\partial y_l}{\partial w_{l0}} = -(T_{pl} - Y_l) \times f'(y_l) \times 1 \tag{2.20}$$

Selanjutnya perubahan bobot dilanjutkan ke bobot-bobot yang terletak di antara lapisan tersembunyi dengan lapisan masukan. Perubahan bobot-bobot ini dapat dihitung dengan cara yang sama dengan cara yang digunakan pada saat melakukan perubahan bobot-bobot di antara lapisan keluaran dengan lapisan tersembunyi yang telah dilakukan sebelumnya.

$$v_{mn}(k+1) = v_{mn}(k) - \alpha \frac{\partial E_p}{\partial v_{mn}} = v_{mn}(k) - \Delta v_{mn} \tag{2.21}$$

di mana

$$\begin{aligned} \frac{\partial E_p}{\partial v_{mn}} &= \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial y_l} \times \frac{\partial y_l}{\partial Z_m} \times \frac{\partial Z_m}{\partial z_m} \times \frac{\partial z_m}{\partial v_{mn}} \\ &= -(T_{pl} - Y_l) \times f'(y_l) \times w_{lm} \times f'(z_m) \times x_{pn} \end{aligned} \tag{2.22}$$

Untuk bias yang berada diantara lapisan tersembunyi dan lapisan masukan, besarnya nilai bias untuk setiap iterasinya dapat dihitung dengan menggunakan persamaan 2.23 berikut:

$$v_{m0}(k+1) = v_{m0}(k) - \alpha \frac{\partial E_p}{\partial v_{m0}} = v_{m0}(k) - \Delta v_{m0} \tag{2.23}$$

di mana

$$\begin{aligned} \frac{\partial E_p}{\partial v_{m0}} &= \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial y_l} \times \frac{\partial y_l}{\partial Z_m} \times \frac{\partial Z_m}{\partial z_m} \times \frac{\partial z_m}{\partial v_{m0}} \\ &= -(T_{pl} - Y_l) \times f'(y_l) \times w_{lm} \times f'(z_m) \times 1 \end{aligned} \tag{2.24}$$

Dari kedua persamaan 2.22 dan 2.24, terlihat bahwa semakin besar *error* yang terjadi, semakin besar nilai dari turunan pertama fungsi *error* terhadap bobot yang bersangkutan.

Pemilihan besarnya nilai konstanta pembelajaran merupakan suatu langkah yang penting. Pemilihan nilai konstanta pembelajaran yang besar akan mempercepat proses pelatihan menuju *error* minimum namun akan sangat sulit untuk mendapatkan nilai bobot yang menghasilkan *error* minimum. Namun apabila konstanta pembelajaran bernilai kecil, proses pelatihan akan menjadi lambat dan memperbesar kesempatan untuk dapat menemukan besarnya nilai bobot yang menghasilkan *error* minimum namun akan memperbesar kemungkinan untuk terjebak dalam lokal minimum (gradien 0 di titik yang bukan minimum).

Propagasi balik dengan fungsi *error cross-entropy*

Propagasi balik dengan fungsi *error cross-entropy* pada dasarnya sama seperti BP dengan persamaan *error* kuadratis yang telah dibahas sebelumnya. Fungsi *error cross-entropy* dapat dilihat pada persamaan 2.25 berikut:

$$E_p = \sum_{l=1}^{L}\left(-T_{pl}\ln(Y_l)-(1-T_{pl})\ln(1-Y_l)\right) \tag{2.25}$$

Sama seperti fungsi *error* kuadratis, metode penurunan *error* dengan menggunakan fungsi *error* ini menggunakan metode *Steepest Descend* yang telah dibahas. Dari hasil pembahasan sebelumnya, penerapan metode *Steepest Descend* untuk memperbaiki nilai bobot yang ada di antara lapisan keluaran dan tersembunyi dapat menggunakan persamaan 2.17. Dengan menggunakan fungsi *error Cross-Entropy*, maka diperoleh besarnya perbaikan, Δw_{lm}, berikut:

$$\frac{\partial E_p}{\partial w_{lm}} = \frac{\partial E_p}{\partial Y_l}\times\frac{\partial Y_l}{\partial y_l}\times\frac{\partial y_l}{\partial w_{lm}} = -\left(\frac{T_{pl}-Y_l}{Y_l(1-Y_l)}\right)\times f'(y_l)\times Z_m \tag{2.26}$$

Dengan cara yang sama, besarnya perbaikan bias pada lapisan keluaran (Δw_{l0}), bobot yang ada di antara lapisan tersembunyi dan lapisan masukan (Δv_{mn}), dan bias untuk lapisan tersembunyi (Δv_{m0}), secara berurut-urut dapat ditentukan dengan persamaan 2.19, 2.21, dan 2.23 yang disederhanakan menjadi berikut:

$$\frac{\partial E_p}{\partial w_{l0}} = \frac{\partial E_p}{\partial Y_l}\times\frac{\partial Y_l}{\partial y_l}\times\frac{\partial y_l}{\partial w_{l0}} = \frac{T_{pl}-Y_l}{Y_l(1-Y_l)}\times f'(y_l)\times 1 \tag{2.27}$$

$$\frac{\partial E_p}{\partial v_{mn}} = \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial y_l} \times \frac{\partial y_l}{\partial Z_m} \times \frac{\partial Z_m}{\partial z_m} \times \frac{\partial z_m}{\partial v_{mn}}$$
$$= -\left(\frac{T_{pl} - Y_l}{Y_l(1 - Y_l)} \right) \times f'(y_l) \times w_{lm} \times f'(z_m) \times x_{pn} \quad (2.28)$$

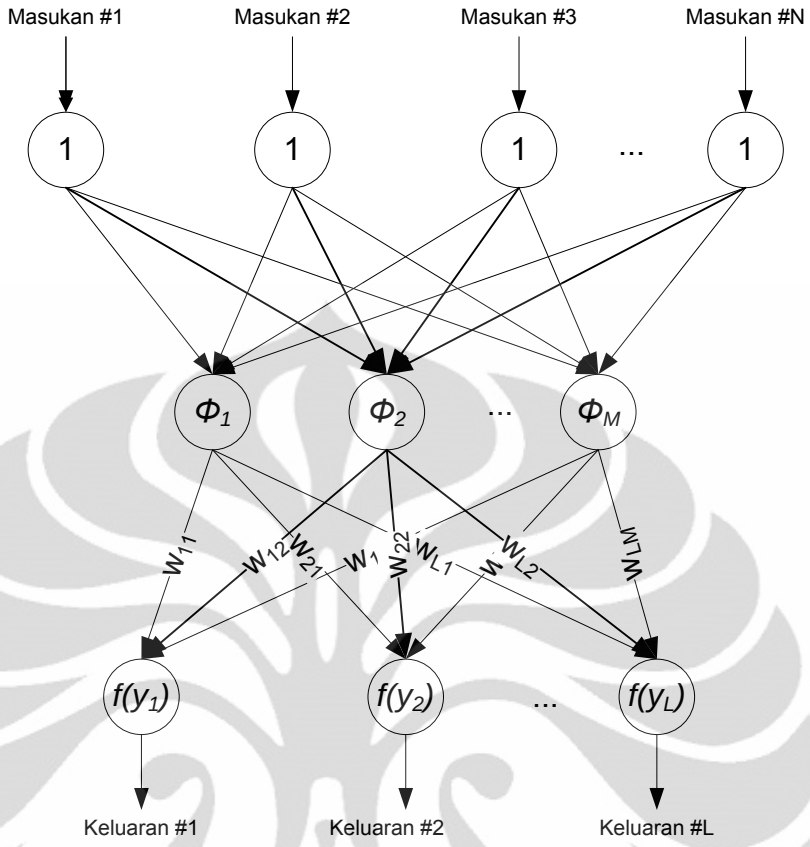
$$\frac{\partial E_p}{\partial v_{m0}} = \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial y_l} \times \frac{\partial y_l}{\partial Z_m} \times \frac{\partial Z_m}{\partial z_m} \times \frac{\partial z_m}{\partial v_{m0}}$$
$$= -\left(\frac{T_{pl} - Y_l}{Y_l(1 - Y_l)} \right) \times f'(y_l) \times w_{lm} \times f'(z_m) \times 1 \quad (2.29)$$

2.3 Pelatihan JST dengan Algoritma *Radial Basis Function* (RBF)

Pada pemakaiannya, pelatihan JST yang diarahkan dapat dianggap sebagai proses *curve-fitting* (Ham & Kostanic, 2000). Dengan kata lain, JST digunakan untuk memetakan suatu *input* menjadi *output* yang berkesesuaian. Salah satu metode untuk memetakan masukan menjadi keluaran adalah RBF dengan arsitekturnya dapat dilihat pada gambar 2.4. RBF merupakan salah satu algoritma pelatihan pada JST yang memanfaatkan jarak antara data dengan nilai tengah data hasil pengelompokan. Secara umum, RBF merupakan gabungan antara fungsi basis radial dengan BP. Radial basis digunakan dari lapisan masukan hingga tersembunyi dan BP digunakan di sisa lapisannya. Dari gambar 2.4, terlihat fungsi aktivasi yang digunakan pada lapisan tersembunyi dan lapisan keluaran berbeda. Fungsi aktivasi yang digunakan pada hasil fungsi radial, adalah sebagai berikut:

1. $\phi(x) = x$ fungsi linear
2. $\phi(x) = x^3$ pendekatan kubik
3. $\phi(x) = x^2 \ln(x)$ fungsi *thin-plate-spline*
4. $\phi(x) = \exp\left(-x^2/(2\sigma^2)\right)$ fungsi Gaussian
5. $\phi(x) = \sqrt{x^2 + \sigma^2}$ fungsi multikuadratik
6. $\phi(x) = \frac{1}{\sqrt{x^2 + \sigma^2}}$ fungsi inversi multikuadratik

Berdasarkan perubahan nilai tengah, pelatihan dengan algoritma RBF dapat dibagi kedalam 2 jenis, yakni pelatihan RBF dengan nilai vektor tengah yang tetap dan pelatihan RBF dengan menggunakan pendekatan gradien stokastik.



Gambar 2.4. Arsitektur JST RBF

2.3.1 Pelatihan RBF Metode Nilai Tengah Tetap

Algoritma pelatihan RBF yang paling sederhana adalah dengan mengasumsikan bahwa nilai tengah RBF selalu tetap. Pendekatan ini pertama kali dikenalkan oleh Broomhead dan Lowe. Pada umumnya, metode ini menggunakan nilai tengah yang diambil secara acak dari data pelatihan. Jumlah nilai tengah yang cukup yang diperoleh secara acak ini akan terdistribusi menurut *Probability Density Function* (PDF) dari data masukan sehingga dapat mewakili seluruh data masukan. Namun demikian, sangat sulit untuk mendapatkan ukuran kuantitas nilai tengah yang mampu mewakili seluruh data. Solusi yang paling mudah adalah dengan mengambil banyak vektor data masukan sebagai vektor nilai tengah untuk memastikan bahwa jumlah vektor nilai tengah sudah cukup mewakili seluruh data.

Ketika vektor nilai tengah telah terpilih, keluaran dari jaringan ini dapat dihitung dengan menggunakan persamaan 2.30 berikut:

$$y_{pl} = \sum_{m=1}^{M} w_{lm} \phi(\boldsymbol{x}_p, \boldsymbol{c}_m) \tag{2.30}$$

di mana *p* menunjukkan indeks data, *l* menunjukkan indeks neuron pada lapisan keluaran, *m* menunjukkan indeks neuron pada lapisan tersembunyi, *M* menunjukkan jumlah neuron pada lapisan tersembunyi. Dengan menyusun ulang persamaan 2.30menjadi bentuk matriks, maka diperoleh:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} = \begin{bmatrix} \phi(\boldsymbol{x}_1, \boldsymbol{c}_1) & \phi(\boldsymbol{x}_1, \boldsymbol{c}_2) & \cdots & \phi(\boldsymbol{x}_1, \boldsymbol{c}_M) \\ \phi(\boldsymbol{x}_2, \boldsymbol{c}_1) & \phi(\boldsymbol{x}_2, \boldsymbol{c}_2) & \cdots & \phi(\boldsymbol{x}_2, \boldsymbol{c}_M) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\boldsymbol{x}_P, \boldsymbol{c}_1) & \phi(\boldsymbol{x}_P, \boldsymbol{c}_2) & \cdots & \phi(\boldsymbol{x}_P, \boldsymbol{c}_M) \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ \vdots \\ w_{lm} \end{bmatrix} \tag{2.31}$$

$$\boldsymbol{y} = \boldsymbol{\varphi} \boldsymbol{w} \tag{2.32}$$

di mana $\boldsymbol{y} \in \Re^{P \times 1}$ merupakan hasil darilapisan keluaran JST yang sebenarnya, $\boldsymbol{\varphi} \in \Re^{P \times (m \times l)}$ merupakan matriks pemetaan yang dilakukan oleh lapisan tersembunyi, dan $\boldsymbol{w} \in \Re^{(m \times l) \times 1}$ merupakan vektor bobot yang ada di antara lapisan tersembunyi dan lapisan keluaran yang banyaknya sebesar (*m* x *l*) x 1. Contohnya, apabila jumlah neuron keluaran adalah 3 dan jumlah neuron tersembunyi adalah 2, maka ukuran vektor **w** adalah 6 x 1.

Karena nilai tengah selalu tetap, hasil operasi pemetaan yang dilakukan oleh lapisan tersembunyi akan selalu tetap. Sehingga proses pelatihan hanya ditujukan untuk mencari besarnya nilai vektor bobot, ***w***, agar hasil pemetaan data masukan menjadi data keluaran dapat terpenuhi. Fungsi *error* yang umumnya digunakan adalah fungsi *error* kuadratis seperti yang telah di jelaskan pada bagian BP dengan menggunakan *error* kuadratis pada bagian Pelatihan JST dengan algoritma *BP*.

$$E_p(\boldsymbol{w}) = \frac{1}{2} (\boldsymbol{T}_p - \boldsymbol{y})^T (\boldsymbol{T}_p - \boldsymbol{y}) \tag{2.33}$$

di mana $\boldsymbol{T}_p \in \Re^{L \times 1}$merupakan vektor keluaran yang diharapkan. Dengan mensubstitusikan persamaan 2.32 ke dalam 2.33, maka diperoleh hasil berikut:

$$E_p(\boldsymbol{w}) = \frac{1}{2} (\boldsymbol{T}_p - \boldsymbol{\varphi w})^T (\boldsymbol{T}_p - \boldsymbol{\varphi w}) = \frac{1}{2} (\boldsymbol{T}_p{}^T \boldsymbol{T}_p - 2\boldsymbol{T}_p{}^T \boldsymbol{\varphi w} + \boldsymbol{w}^T \boldsymbol{\varphi}^T \boldsymbol{\varphi w}) \tag{2.34}$$

Untuk memperkecil nilai *error* E_p dapat digunakan persamaan:

$$\frac{\partial E_p(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\boldsymbol{\varphi}^T \boldsymbol{T}_p + \boldsymbol{\varphi}^T \boldsymbol{\varphi} \boldsymbol{w} = 0 \tag{2.35}$$

Dengan kata lain, nilai vektor bobot, ***w***, dapat ditentukan dengan persamaan:

$$\boldsymbol{w} = \left(\boldsymbol{\varphi}^T \boldsymbol{\varphi}\right)^{-1} \boldsymbol{\varphi}^T \boldsymbol{T}_p = \boldsymbol{\varphi}^+ \boldsymbol{T}_p \tag{2.36}$$

di mana $\boldsymbol{\varphi}^+$ menyatakan *pseudoinvers* dari pemetaan tidak linear matriks $\boldsymbol{\varphi}$.

Dari persamaan 2.36, dapat dilihat bahwa untuk banyaknya nilai tengah yang memadai, besarnya nilai bobot dapat dihitung secara langsung dalam sekali perhitungan. Dengan demikian, algoritma RBF ini sangat cepat dan sederhana dibandingkan dengan algoritma BP. Apabila banyaknya vektor nilai tengah dapat ditentukan dengan baik, *error* 0 mungkin akan diperoleh walaupun sangat sulit untuk mendapatkan vektor nilai tengah ini.

Untuk kasus RBF yang menggunakan fungsi aktifasi Gaussian pada lapisan tersembunyi, besarnya inisialisasi nilai parameter lebar RBF umumnya ditentukan dengan persamaan 2.37 berikut:

$$\sigma = \frac{d_{max}}{\sqrt{k}} \tag{2.37}$$

di mana d_{max} merupakan jarak *euclidean* terbesar antar vektor nilai tengah yang terpilih dan k merupakan banyaknya vektor nilai tengah yang terpilih.

2.3.2 Pelatihan RBF Metode *Stochastic Gradient Approach*

Pada bagian sebelumnya, telah dijelaskan mengenai pelatihan JST dengan nilai tengah tetap. Seperti yang telah dijelaskan, pendekatan ini menghasilkan algoritma pelatihan yang sangat sederhana dan sangat cepat. Namun demikian, untuk mendapatkan banyaknya vektor tengah yang baik, banyak data harus diambil secara acak. Hal ini akan menyebabkan ukuran jaringan yang relatif besar, bahkan untuk permasalahan yang sederhana.

Pendekatan gradien stokastik untuk melatih JST dengan metode RBF memungkinkan untuk melakukan pelatihan terhadap tiga parameter yakni bobot, nilai tengah RBF, dan lebar RBF. Dengan demikian, JST memiliki kemampuan untuk mempebaiki nilai tengah dan lebarnya, layaknya perbaikan bobot guna mendapatkan nilai tengah dan lebar RBF yang baik.

Proses pelatihan untuk ketiga parameter nilai tengah RBF, lebar RBF, dan bobot-bobot merupakan proses pembelajaran yang diarahkan. Untuk memahami proses pelatihan ketiga parameter ini, berikut akan dibahas persamaan matematis yang dibagi ke dalam 2 bagian, yakni propagasi maju dan propagasi mundur.

Propagasi maju

Apabila terdapat sebuah JST yang hendak dilatih dengan algoritma RBF seperti yang ditunjukkan pada gambar 2.4 dengan fungsi aktifasi yang digunakan pada lapisan tersembunyi adalah fungsi Gaussian. Inisialisasi besarnya vektor nilai tengah diperoleh dengan cara mencari rata-rata vektor untuk setiap kelasnya. Akibatnya, banyaknya vektor nilai tengah sama dengan banyaknya jumlah kelas. Dengan demikian, besarnya nilai yang keluar dari tiap neuron pada lapisan tersembunyi dapat ditentukan oleh persamaan berikut:

$$Z_m = \sum_{n=1}^{N} \exp\left(-\frac{\left(x_{pn} - c_{mn}\right)^2}{2\sigma_m^{\,2}} \right) \tag{2.38}$$

di mana x_{pn} menunjukkan data dengan indeks ke-p dimensi ke-n, c_{mn} menunjukkan vektor nilai tengah ke-m dimensi ke-n, σ_m menunjukkan lebar dari RBF ke-m dan Z_m merupakan keluaran dari neuron ke-m pada lapisan tersembunyi. Vektor hasil dari lapisan tersembunyi akan dipropagasi maju menuju lapisan keluaran. Besarnya nilai yang dihasilkan pada tiap neuron keluaran dapat ditentukan dengan menggunakan persamaan 2.39 berikut:

$$Y_l = \sum_{m=1}^{M} w_{lm} Z_m \tag{2.39}$$

di mana w_{lm} adalah bobot dari neuron pada lapisan tersembunyi ke-m menuju neuron pada lapisan keluaran ke-l dan Y_l merupakan hasil dari neuron pada lapisan keluaran ke-l.

Propagasi balik dengan fungsi *error* kuadratis

Untuk setiap data yang masuk ke JST dan selesai melakukan propagasi maju, maka *error* dihitung relatif terhadap target yang bersesuaian dengan menggunakanpersamaan 2.40berikut:

$$E_p = \frac{1}{2}\left(\boldsymbol{T}_p - \boldsymbol{Y}\right)^T\left(\boldsymbol{T}_p - \boldsymbol{Y}\right) = \frac{1}{2}\sum_{l=1}^{L}\left(T_{pl} - Y_l\right)^2 = \frac{1}{2}\sum_{l=1}^{L} e_l{}^2 \tag{2.40}$$

di mana $\boldsymbol{T}_p \in \Re^{L\times 1}$ merupakan vektor target urutan data ke-*p* dan $\boldsymbol{Y} \in \Re^{L\times 1}$ merupakan vektor keluaran dari lapisan keluaran yang diperoleh dari data ke-*p*.

Seperti yang telah dijelaskan, tujuan dari proses pelatihan RBF dengan pendekatan gradien stokastik adalah untuk mencari nilai bobot yang berada di antara lapisan keluaran dan lapisan tersembunyi, vektor nilai tengah RBF, dan lebar RBF untuk memperkecil *error*. Untuk memperbaiki ketiga besaran tersebut, digunakan proses iterasi dengan metode *Steepest Descend* sama seperti yang dilakukan di BP.Untuk memperbaiki bobot yang berada di antara lapisan keluaran dan lapisan tersembunyi, dapat menggunakan persamaan 2.41 berikut:

$$w_{lm}(k+1) = w_{lm}(k) - \alpha\frac{\partial E_p}{\partial w_{lm}} = w_{lm}(k) - \Delta w_{lm} \tag{2.41}$$

di mana

$$\frac{\partial E_p}{\partial w_{lm}} = \frac{\partial E_p}{\partial Y_l}\times\frac{\partial Y_l}{\partial w_{lm}} = -\left(T_{pl} - Y_l\right)\times Z_m \tag{2.42}$$

Selanjutnnya, *error* dipropagasi balik lagi menuju lapisan tersembunyi guna memperbaiki vektor nilai tengah RBF dan lebar RBF. Untuk memperbaiki vektor nilai tengah RBF, digunakan persamaan 2.43 berikut:

$$c_{mn}(k+1) = c_{mn}(k) - \alpha_c\frac{\partial E_p}{\partial c_{mn}} = c_{mn}(k) - \Delta c_{mn} \tag{2.43}$$

di mana

$$\frac{\partial E_p}{\partial c_{mn}} = \frac{\partial E_p}{\partial Y_l}\times\frac{\partial Y_l}{\partial Z_m}\times\frac{\partial Z_m}{\partial c_{mn}} = -\left(T_{pl} - Y_l\right)\times w_{lm}\times\frac{\left(x_{pn} - c_{mn}\right)}{\sigma_m{}^2}Z_m \tag{2.44}$$

Untuk memperbaiki lebar RBF, dapat menggunakan persamaan 2.45 berikut:

$$\sigma_m(k+1) = \sigma_m(k) - \alpha_\sigma\frac{\partial E_p}{\partial \sigma_m} = \sigma_m(k) - \Delta\sigma_m \tag{2.45}$$

di mana

$$\frac{\partial E_p}{\partial \sigma_m} = \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial Z_m} \times \frac{\partial Z_m}{\partial \sigma_m} = -\left(T_{pl} - Y_l\right) \times w_{lm} \times \frac{\left\| x_{pn} - c_{mn} \right\|^2}{{\sigma_m}^3} Z_m \qquad (2.46)$$

Propagasi balik dengan fungsi *error Cross-Entropy*

Seperti pada fungsi *error* kuadratis, untuk setiap data yang masuk ke JST dan selesai melakukan propagasi maju, maka *error* dihitung relatif terhadap target yang bersesuaian dengan menggunakanpersamaan 2.47 berikut:

$$E_p = \sum_{l=1}^{L} \left(-T_{pl} \ln\left(Y_l\right) - \left(1 - T_{pl}\right) \ln\left(1 - Y_l\right) \right) \qquad (2.47)$$

di mana $\boldsymbol{T}_p \in \Re^{L \times 1}$ merupakan vektor target urutan data ke-*p* dan $\boldsymbol{Y} \in \Re^{L \times 1}$ merupakan vektor keluaran dari lapisan keluaran yang diperoleh dari data ke-*p*.

Sama halnya fungsi *error* kuadratis, propagasi balik dilakukan guna memperbaiki bobot-bobot yang ada di antara lapisan keluaran dan tersembunyi, nilai tengah RBF dan lebar RBF dengan menggunakan metode *Steepest Descend*. Berdasarkan pembahasan sebelumnya, besarnya penyesuaian bobot-bobot antara lapisan keluaran dan tersembunyi dapat ditentukan dengan persamaan 2.41. Dengan menggunakan fungsi *error Cross-Entropy*, maka besarnya Δw_{lm} dapat ditentukan dengan persamaan:

$$\frac{\partial E_p}{\partial w_{lm}} = \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial w_{lm}} = -\left(\frac{T_{pl} - Y_l}{Y_l\left(1 - Y_l\right)} \right) \times Z_m \qquad (2.48)$$

Dengan cara yang sama, maka besarnya perbaikan nilai tengah RBF (Δc_{mn}) danlebar RBF ($\Delta \sigma_m$)secara berurut-urut dapat ditentukan dengan persamaan 2.43 dan 2.45 yang disederhanakan menjadi berikut:

$$\frac{\partial E_p}{\partial c_{mn}} = \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial Z_m} \times \frac{\partial Z_m}{\partial c_{mn}} = -\left(\frac{T_{pl} - Y_l}{Y_l\left(1 - Y_l\right)} \right) \times w_{lm} \times \frac{\left(x_{pn} - c_{mn}\right)}{{\sigma_m}^2} Z_m \qquad (2.49)$$

$$\frac{\partial E_p}{\partial \sigma_m} = \frac{\partial E_p}{\partial Y_l} \times \frac{\partial Y_l}{\partial Z_m} \times \frac{\partial Z_m}{\partial \sigma_m} = -\left(\frac{T_{pl} - Y_l}{Y_l\left(1 - Y_l\right)} \right) \times w_{lm} \times \frac{\left\| x_{pn} - c_{mn} \right\|^2}{{\sigma_m}^3} Z_m \qquad (2.50)$$

Dari hasil penjelasan matematis, dapat terlihat bahwa pelatihan dengan menggunakan metode pendekatan gradien stokastik membutuhkan algoritma yang

tidak sederhana. Namun demikian, algoritma ini memungkinkan untuk melewati proses pencari vektor nilai tengah yang relatif sulit untuk ditemukan.

2.4 *Negative Correlation Learning* (NCL)

Untuk menghasilkan suatu keputusan bersama yang tepat, maka diperlukan perbedaan opini di antara anggotanya. Sama seperti halnya JST *ensemble*, untuk membuat keputusan akhir yang akurat, diperlukan classifier yang bervariasi, yang dihasilkan oleh suatu *diversity generator*. Berangkat dari pemikiran tersebut, (Liu & Yao, 1999) mengajukan suatu metode untuk mendiversifikasi proses pelatihan pada JST *ensemble*, yaitu *Negative Correlated Learning*.

Tujuan dari NCL adalah untuk melatih setiap JST yang berada pada satu sistem dengan bagian atau aspek yang berbeda dari suatu kumpulan data pelatihan sehingga jaringan *ensemble* dapat melakukan proses pelatihan yang lebih baik. Untuk mencapai hal tersebut, setiap JST diterapkan suatu penalti yang terdapat pada fungsi *error*-nya.

Berikut algoritma yang ditawarkan oleh NCL. Apabila terdapat suatu kumpulan data beserta target pelatihan yang berkesesuaian yang dilambangkan dengan $\boldsymbol{x}(n) \in \Re$ dan $\boldsymbol{T}(n) \in \Re$, di mana *n* menunjukkan urutan data. Setiap JST ke-*i* memiliki keluaran $F_i(n)$ dan keluaran untuk keseluruhan jaringan adalah $F(n)$ yang dinyatakan sebagai berikut:

$$F(n) = \frac{1}{M}\sum_{i=1}^{N} F_i(n) \tag{2.51}$$

di mana *M* menunjukkan banyaknya JST pada sistem tersebut.

Fungsi *error* kuadratis

Seperti yang telah dijelaskan sebelumnya, algoritma NCL bekerja dengan cara menambahkan efek penalti pada fungsi *error* masing-masing jaringan seperti berikut:

$$E_i = \frac{1}{N}\sum_{n=1}^{N} E_i(n) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\left(F_i(n) - d(n)\right)^2 + \frac{1}{N}\sum_{n=1}^{N}\lambda p_i(n) \tag{2.52}$$

di mana $E_i(n)$ adalah nilai fungsi *error* jaringan ke-*i* pada pola latihan ke-*n*. Suku pertama pada persamaan 2.52 adalah *empirical error* dan suku kedua adalah fungsi penalti. Dengan meminimalkan $p_i(n)$, maka hasil pelatihan masing-masing JST dapat dikorelasikan secara negatif. Parameter λ $(0 \le \lambda \le 1)$ berfungsi sebagai kekuatan penalti. Sebagai besar parameter ini, semakin kuat pengaruh penalti. Fungsi penalti sendiri didefinisikan dengan persamaan berikut:

$$p_i(n) = (F_i(n) - F(n)) \sum_{j \neq i}^{M} (F_J(n) - F(n)) \tag{2.53}$$

Untuk dapat diterapkan ke algoritma pelatihan, maka perlu diketahui turunan persamaan 2.52 terhadap keluaran JST ke-*i* yang dapat dinyatakan ke dalam persamaan berikut:

$$\begin{aligned} \frac{\partial E_i(n)}{\partial F_i(n)} &= (F_i(n) - d(n)) + \lambda \frac{\partial p_i(n)}{\partial F_i(n)} \\ &= (F_i(n) - d(n)) + \lambda \sum_{j \neq i}^{M} (F_j(n) - F(n)) \\ &= (F_i(n) - d(n)) + \lambda (MF(n) - F_i(n) - (M-1)F(n)) \\ &= F_i(n) - d(n) + \lambda (F(n) - F_i(n)) \\ &= F_i(n) - d(n) - \lambda (F_i(n) - F(n)) \\ &= (1-\lambda)(F_i(n) - d(n)) + \lambda (F(n) - d(n)) \end{aligned} \tag{2.54}$$

Fungsi *error Cross-Entropy*

Sama halnya dengan fungsi *error* kuadratis, algortima NCL bekerja dengan cara menambahkan efek penalti pada fungsi *error* masing-masing jaringan seperti berikut:

$$\begin{aligned} E_i &= \frac{1}{N} \sum_{n=1}^{N} E_i(n) \\ &= \frac{1}{N} \sum_{n=1}^{N} (-T(n) \ln(F_i(n)) - (1-T(n)) \ln(1-F_i(n))) + \frac{1}{N} \sum_{n=1}^{N} \lambda p_i(n) \end{aligned} \tag{2.55}$$

di mana $E_i(n)$ adalah nilai fungsi *error* jaringan ke-*i* pada pola latihan ke-*n*. Suku pertama pada persamaan 2.55 adalah *empirical error* dan suku kedua adalah fungsi penalti. Dengan meminimalkan $p_i(n)$, maka hasil pelatihan masing-masing JST dapat dikorelasikan secara negatif. Parameter λ $(0 \le \lambda \le 1)$ berfungsi sebagai kekuatan penalti. Sebagai besar parameter ini, semakin kuat pengaruh penalti. Fungsi penalti dapat dilihat pada persamaan 2.53.

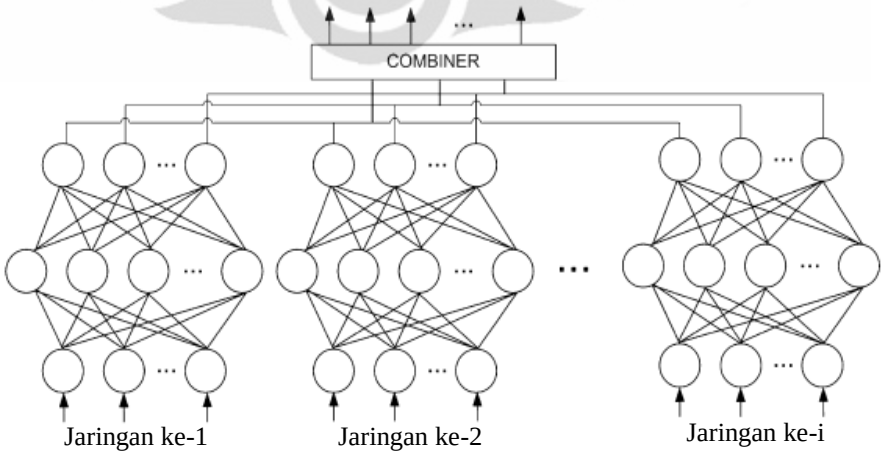
Untuk dapat diterapkan ke algoritma pelatihan, maka perlu diketahui turunan persamaan 2.55 terhadap keluaran JST ke-*i* yang dapat dinyatakan ke dalam persamaan berikut:

$$\begin{aligned}\frac{\partial E_i(n)}{\partial F_i(n)} &= -\left(\frac{d(n)-F_i(n)}{F_i(n)(1-F_i(n))}\right)+\lambda\frac{\partial p_i(n)}{\partial F_i(n)} \\ &= -\left(\frac{d(n)-F_i(n)}{F_i(n)(1-F_i(n))}\right)+\lambda\sum_{j\neq i}^{M}\left(F_j(n)-F(n)\right) \\ &= -\left(\frac{d(n)-F_i(n)}{F_i(n)(1-F_i(n))}\right)+\lambda\left(MF(n)-F_i(n)-(M-1)F(n)\right) \\ &= -\left(\frac{d(n)-F_i(n)}{F_i(n)(1-F_i(n))}\right)+\lambda\left(F(n)-F_i(n)\right)\end{aligned} \tag{2.56}$$

Setelah mengetahui persamaan 2.50 dan 2.56, algoritma pelatihan dapat digunakan untuk memodifikasi parameter-parameter yang hendak diubah guna mendapatnya *error* minimum seperti yang dilakukan pada algoritma BP maupun RBF.

2.5 Pelatihan JST *Ensemble* dengan Algoritma RBF

JST *ensemble* pertama kali diajukan pada tahun 1990 oleh Lars Kai Hansen dan Peter Salomon. JST *ensemble* merupakan suatu sistem dengan jumlah JST lebih dari satu yang dilatih bersama secara *supervised* (diarahkan) dan mengambil keputusan bersama.



Gambar 2.5. Arsitektur JST *Ensamble*

Arsitektur JST *ensemble* secara umum dapat dilihat pada gambar 2.5 di mana terdapat *i* jaringan dengan indeks untuk masing-masing jaringan mengikuti aturan yang sama seperti pada jaringan tunggal yang telah dijelaskan pada butir 2.3. Dari gambar 2.5, setiap JST dapat berdiri sendiri di mana masing-masing keluaran JST dihubungan ke sebuah *combiner* yang berfungsi sebagai pengambil keputusan. Ada beberapa metode untuk mengambil keputusan, yakni:

1. Metode rata-rata

$$H_{i,l}(x) = \frac{1}{I}\sum_{j=1}^{I} Y_{j,l}(x)$$

2. Metode nilai maksimum

$$H_{i,l}(x) = \max_{j=1-L}\left[Y_{i,j}(x)\right]$$

3. Metode perkalian produk

$$H_{i,l}(x) = \prod_{j=1}^{I} Y_{j,l}(x)$$

4. Metode voting

$$H(x) = \max_{k=1,2,..C}\left[\sum_{j=1}^{N} B(x)\right]$$

di mana B adalah jumlah kelas yang diurutkan di bawah kelas k pada jaringan ke-i. Perhitungan ini disebut dengan *Borda Count* (Lee, Hong, & Kim, 2009).

Dari pembahasan sebelumnya, telah diketahui persamaan matematis untuk pelatihan JST tunggal dengan algoritma RBF serta algoritma yang ditawarkan NCL dalam proses pelatihan. Pada JST *ensemble*, proses pelatihan merupakan gabungan antara pelatihan JST tunggal dan NCL. Proses propagasi maju berlangsung dari data yang dikirim ke tiap lapisan pada tiap JST hingga masing-masing JST memiliki keputusan yang kemudian dikombinasikan guna mendapatkan keputusan bersama. Setelah sebuah data melalui proses propagasi maju, maka *error* yang dihasilkan oleh keputusan bersama di propagasi balik dengan algoritma NCL terhadap *error* masing-masing JST yang selanjutnya dipropagasi balik guna memperbaiki nilai bobot-bobot, besar nilai tengah RBF, dan lebar RBF untuk masing-masing JST.

2.6 *Principal Component Analysis* (PCA)

PCA pertama kali dikembangkan oleh Karl Pearson (1901). PCA merupakan suatu metode pemeriksaan otomatis dan sistematis terhadap korelasi di antara banyak variabel, yang bertujuan untuk mencari *Principal Component* yang terselubung. *Principal Component* adalah variabel-variabel yang tidak memiliki hubungan di antaranya. Dengan kata lain, PCA adalah suatu model statistik yang digunakan untuk mengidentifikasi pola suatu data berdimensi banyak dan menyatakannya kembali dalam data berdimensi lebih rendah dengan cara mencari *Principal Component*.

Konsep utama yang ditawarkan oleh PCA adalah mereduksi dimensi dari suatu set data terdiri dari banyak variabel, di mana kemungkinan-kemungkinan variasi sebisa mungkin dipertahankan. Hal ini dapat dicapai dengan cara mentransformasi data tersebut menjadi *Principal Component*. Berikut akan dibahas proses PCA:

1. Menyusun data

   Data yang hendak diproses, digabungkan dan disusun ulang hingga memperoleh di dalam tiap baris berisikan data dan tiap kolom berisikan atribut atau dimensi dari data seperti berikut:

$$\boldsymbol{x} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{P1} & x_{P2} & \cdots & x_{PN} \end{bmatrix} \tag{2.57}$$

2. Normalisasi Z

   Setiap data pada matriks $\boldsymbol{x}$ akan ditentukan nilai-z (*zscore*) yang digabung kedalam matriks$\boldsymbol{Z}$, yakni setiap elemen dari matriks $\boldsymbol{x}$ akan dikurang dengan rata-rata dari atribut atau dimensi yang sama dan dibagi dengan standar deviasi dari atribut atau dimensi tersebut.

$$\boldsymbol{Z} = \begin{bmatrix} \frac{x_{11}-\bar{x}_1}{\mathrm{std}(x_1)} & \frac{x_{12}-\bar{x}_2}{\mathrm{std}(x_2)} & \cdots & \frac{x_{1N}-\bar{x}_N}{\mathrm{std}(x_N)} \\ \frac{x_{21}-\bar{x}_1}{\mathrm{std}(x_1)} & \frac{x_{22}-\bar{x}_2}{\mathrm{std}(x_2)} & \cdots & \frac{x_{2N}-\bar{x}_N}{\mathrm{std}(x_N)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_{P1}-\bar{x}_1}{\mathrm{std}(x_1)} & \frac{x_{P2}-\bar{x}_2}{\mathrm{std}(x_2)} & \cdots & \frac{x_{PN}-\bar{x}_N}{\mathrm{std}(x_N)} \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1N} \\ z_{21} & z_{22} & \cdots & z_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ z_{P1} & z_{P2} & \cdots & z_{PN} \end{bmatrix} \tag{2.58}$$

3. Menghitung matriks kovarians

   Mencari matriks kovarians dari matriks data baru, ***x***, dengan menggunakan formula:

$$C_Z = \frac{\boldsymbol{Z}^T \boldsymbol{Z}}{(n-1)} \tag{2.59}$$

4. Menghitung vektor dan nilai eigen dari matriks kovarians

   Karena matriks kovarians adalah persegi, maka pencarian nilai eigen dan vektor eigen yang berpasangan dapat dilakukan. Semua vektor eigen yang diperoleh merupakan vektor yang saling tegak lurus dan memiliki besar sama dengan satu (*unit eigen vector*) atau disebut dengan *orthogonal*. Hasil dari vektor eigen sangat penting karena hal ini menunjukkan representasi data di dalam ruang eigen.

5. Memilih komponen dan membentuk sebuah matriks transformasi

   Nilai eigen yang diperoleh pada langkah sebelumnya menunjukkan tingkat kepentingan data tersebut. Vektor eigen yang memiliki nilai eigen terbesar merupakan komponen utamadari suatu set data. Dengan mengurutkan nilai eigen dari terbesar hingga terkecil, maka akan diperoleh komponen utama dari matriks data. Selanjutnya, vektor eigen dengan nilai eigen kecil atau yang dapat diabaikan, dapat dibuang. Hal ini akan berakibat terhadap hilangnya informasi pada matriks data, namun tidak signifikan. Vektor eigen yang disisakan digunakan sebagai matriks tansformasi untuk mengubah dimensi dari matriks data.

$$Feature\ vector = (eig_1 \quad eig_2 \quad ... \quad eig_n) \tag{2.60}$$

6. Menghitung set data baru

   Matriks transformasi yang berisi vekor eigen digunakan untuk mengubah dimensi matriks data menjadi lebih kecil dengan cara mengalikan matriks datadengan matriks transformasi. Hasil dari perkalian ini berupa matriks data baru dengan dimensi yang lebih kecil namun memiliki informasi yang sama dengan matriks awalnya.

$$FinalData = RowFeatureVector \times RowDataAdjust \tag{2.61}$$

2.7 Nguyen-Widrow

Pada awalnya, bobot-bobot yang digunakan pada JST menggunakan nilai acak yang kecil. Bobot-bobot ini harus cukup kecil sehingga JST tidak mulai dari titik di ruang *error* yang berada di daerah saturasi. Ketika JST beroperasi di daerah saturasi, akan dibutuhkan banyak iterasi untuk proses pembelajara agar mencapai nilai konvergen. Salah satu algoritma untuk menentukan besarnya nilai bobot-bobot awal adalah algoritma yang diajukan oleh Nguyen dan Widrow yang dikenal dengan algoritma Nguyen-Widrow. Berikut algoritma yang ditawarkan oleh metode ini:

1. Hitung faktor skala dengan formula berikut:

$$\beta = 0.7\sqrt[n_0]{n_1} \tag{2.62}$$

   di mana n_0 adalah banyaknya neuron pada lapisan tempat bobot berasal dan n_1 adalahbanyaknya neuron pada lapisan yang dituju oleh bobot
2. Tentukan nilai bobot, w_{ij}, awal secara acak antara -0.5 hingga 0.5
3. Tentukan kembali nilai bobot awal dengan formula berikut:

$$w_{ij} = \beta \frac{w_{ij}}{\sqrt{\sum_{i=1}^{n_1} w_{ij}^{\ 2}}} \tag{2.63}$$

4. Tentukan nilai bias secara acak di antara $-\beta$ hingga β

2.8 Data Percobaan

Pada percobaan yang dilakukan, 11 set data digunakan untuk mengetahui perilaku dan performa dari JST dengan algoritma pelatihan RBF. Secara garis besar, data yang digunakan terdiri dari 2 bagian,yakni 9 set data diperoleh dari badan bersitifikasi internasional UC *Irvine Machine Learning Repository* (UCI) dan 2 set data lainnya diambil secara mandiri. Ringkasan dari ketiga belas data yang digunakan dapat dilihat pada tabel 2.2 di mana 9 set data pertama merupakan data UCI dan 2 set data berikutnya merupakan data yang diambil secara mandiri berupa citra wajah manusia. Untuk detail informasi terhadap data yang digunakan, dapat dilihat pada lampiran A.

Tabel 2.2. Data Percobaan

No.	Nama Data	Jumlah Data	Jumlah Kelas	Jumlah Atribut	Jumlah Data per Kelas
1.	*Balance Scale Weight & Distance*	625	3	5	kelas 1 = 49, kelas 2 = 288, dan kelas 3 = 288
2.	*Breast Cancer*	699	2	11	kelas 1 = 458 dan kelas 2 = 241
3.	*BUPA Liver Disorders*	345	2	7	kelas 1 = 145 dan kelas 2 = 200
4.	*Credit Approval*	690	2	16	kelas 1 = 307 dan kelas 2 = 383
5.	*Glass Identification*	214	7	11	kelas 1 = 70, kelas 2 = 76, kelas 3 = 17, kelas 4 = 0, elas 5 = 13, kelas 6 = 9, dan kelas 7 = 29
6.	*Hearth Diseases*	270	2	14	kelas 1 = 150 dan kelas 2 = 170
7.	*Ionosphere*	351	2	35	kelas 1 = 225 dan kelas 2 = 126
8.	*Iris Plants*	150	3	5	masing-masing kelas = 50
9.	*Sonar*	208	2	61	kelas 1 = 111 dan kelas 2 = 107
10.	Foto dengan kamera infra merah	200	10	1200	masing-masing kelas = 20
11.	Foto dengan kamera cahaya tampak	100	10	900	masing-masing kelas = 10

Selain menggunakan set data UCI, set data citra wajah manusia juga digunakan dengan tujuan untuk mengetahui kemampuan JST untuk mengenali wajah manusia. Data citra wajah yang digunakan dibagi menjadi dua yakni citra wajah yang diambil dengan mneggunakan *webcam* dimana terdapat cahaya tampak dan citra wajah lainnya diperoleh tanpa adanya cahaya dan dengan menggunakan kamera infra merah. Data citra wajah manusia pertama diperoleh dari Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng. selaku pembimbing penulis dan data kedua diperoleh dari skripsi yang disusun oleh Stephen Roy Imantaka, ST dengan judul Sistem Pengenal Wajah Berbasis *Neural Network Ensemble* untuk Citra Infra Merah (2010). Dengan demikian, proses pengambilan data tidak dijelaskan pada skripsi ini melainkan pada laporan yang dirujuk dalam penulisan skripsi ini.

2.9 Pengolahan Awal Data Percobaan

Setelah mendapatkan data-data percobaan, langkah selanjutnya adalah menormalisasi data-data tersebut. Normalisasi perlu dilakukan untuk mengurangi akibat dari ketidak presisian antara satu data dengan data lainnya di dalam satu atribut yang sama untuk setiap percobaannya. Metode normalisasi yang digunakan

pada percobaan ini adalah dengan mencari nilai z seperti yang dilakukan pada tahapan awal PCA.

Seperti yang telah disebutkan sebelumnya, jumlah data yang digunakan pada percobaan berjumlah 11 data. Untuk 9 set data UCI, metode normalisasi dapat langsung diterapkan karena data yang tersedia oleh badan ini telah dalam bentuk angka (data huruf diubah menjadi data angka dengan penyesuaian tertentu). Untuk lebih jelasnya dapat dilihat pada gambar2.6.

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{33} & x_{23} \end{bmatrix} \xrightarrow{\text{normalisasi}} X = \begin{bmatrix} \frac{x_{11} - \bar{x}_1}{\text{std}(x_1)} & \frac{x_{12} - \bar{x}_2}{\text{std}(x_2)} & \frac{x_{13} - \bar{x}_3}{\text{std}(x_3)} \\ \frac{x_{21} - \bar{x}_1}{\text{std}(x_1)} & \frac{x_{22} - \bar{x}_2}{\text{std}(x_2)} & \frac{x_{23} - \bar{x}_3}{\text{std}(x_3)} \end{bmatrix}$$

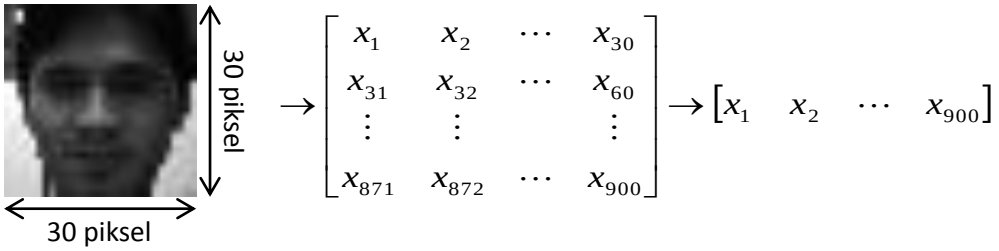
Gambar 2.6. Contoh Normalisasi

di mana $\bar{x}$ dan std(x) merupakan rata-rata dan standar deviasi dari atribut.Sedangkan untuk 2 set data citra wajah manusia, perlu dilakukan pengolahan terlebih dahulu sebelum dilakukan normalisasi. Berikut langkah-langkah pengolahan data citra wajah:

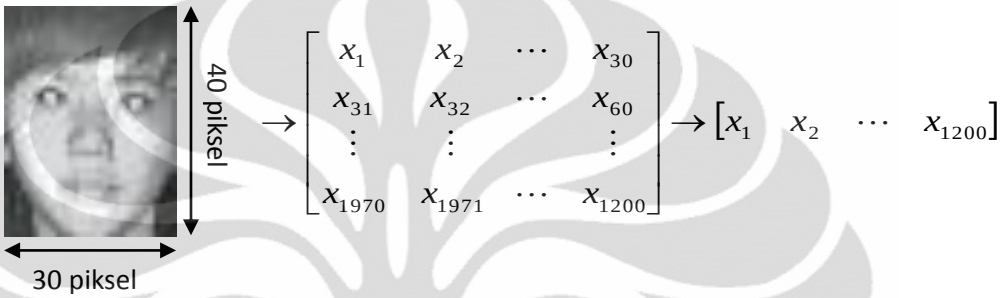
1. *Pre-processing*

   Pada tahap ini, data yang berupa tampilan visual akan dinyatakan ke dalam numerik dengan memperhatikan nilai RGB (*Red-Green-Blue*) dari setiap piksel citra. Karena pencitraan yang diperoleh telah diubah ke dalam *greyscale*, maka besarnya nilai R,G, dan B akan menjadi sama untuk setiap pikselnya.

   Untuk data percobaan, terdapat perbedaan jumlah piksel citra yang digunakan. Pada data citra wajah cahaya tampak, ukuran citra adalah 30 x 30 (900 piksel). Sedangkan citrawajah kamera infra merah, jumlah pikselcitra yang digunakan adalah 40 x 30 (1200 piksel). Dengan demikian, dimensi (jumlah atribut) hasil pembacaan skala RGB dari kedua jenis data menjadi berbeda. Namun untuk mempermudah proses PCA, dimensi hasil pembacaan RGB disusun ulang dimana setiap atribut menempati kolom yang berbeda seperti yang ditunjukkan pada gambar 2.7 dan 2.8.



Gambar 2.7. Contoh *Pre-processing* CitraWajahCahaya Tampak



Gambar 2.8. Contoh Pre-processing CitraWajah Infra Merah

2. Ekstraksi fitur

Pada tahap ini, data numerik hasil pembacaan skala RGB citra mengalami PCA untuk direduksi dimensinya atau jumlah atribut yang terlebih dulu dinormalisasi seperti yang telah dijelaskan pada bagian dasar teori. Hal ini bertujuan untuk menyederhanakan komputasi dan mempercepat waktu pelatihan.

Setelah melakukan normalisasi dan/ atau PCA, data selanjutnya dibagi ke dalam 2 bagian, yaitu set data yang digunakan untuk pelatihan dan set data yang digunakan untuk pengujian. Pada percobaan ini, perbandingan antara set data pelatihan dan set data pengujian adalah 50:50. Jumlah perbandingan ini dipilih karena perbandingan ini dinilai memiliki hasil yang paling optimal.

2.10 Perangkat dan Parameter Percobaan

Dalam percobaan ini, perangkat yang digunakan mencakup perangkat keras dan perangkat lunak. Berikut perangkat percobaan yang digunakan:

1. Perangkat keras

Perangkat keras yang digunakan dalam percobaan ini berupa komputer 3 buah yang dipakai untuk menghitung. Berikut spesifikasi komputer yang digunakan:

a. Komputer 1

Processor : Interl Core2 Duo E4500 @2.2GHz

Memori : 1024 MB RAM

Sistem operasi : Windows WindowsXP Professional

Percobaan : JST tunggal RBF dan BP

b. Komputer 2

Processor : Intel Xeon T7100 @2.66GHz

Memori : 3328 MB RAM

Sistem operasi : Windows WindowsXP Professional

Percobaan : JST *ensemble* RBF

c. Komputer 3

Processor : Intel Xeon T7100 @2.66GHz

Memori : 3328 MB RAM

Sistem operasi : Windows WindowsXP Professional

Percobaan : JST *ensemble* RBF

2. Perangkat lunak

Untuk perangkat lunak yang digunakan dalam pemograman dan eksperimen ini adalah MATLAB versi 7.8.0 (R2009a) perangkat lunak keluaran MathWork, Inc. Fasilitas yang digunakan dari Matlab ini adalah menggunakan M-file untuk membuat fungsi yang nantinya dieksekusi pada Command Window.

3. Parameter Percobaan

Pada percobaan, besarnya nilai parameter yang digunakan pada pemograman ditentukan dengan cara manual, yaitu mencoba beberapa nilai berdasarkan pengalaman dan mencari nilai yang terbaik. Nilai parameter yang diperoleh akan diterapkan untuk seluruh data guna menyederhanakan algoritma dan pemograman. Berikut besarnya nilai parameter yang digunakan pada percobaan:

a. Konstanta pembelajaran *backpropagation* (α) : 0.1

b. Konstanta pembelajaran *center* (α_c) : 0.1

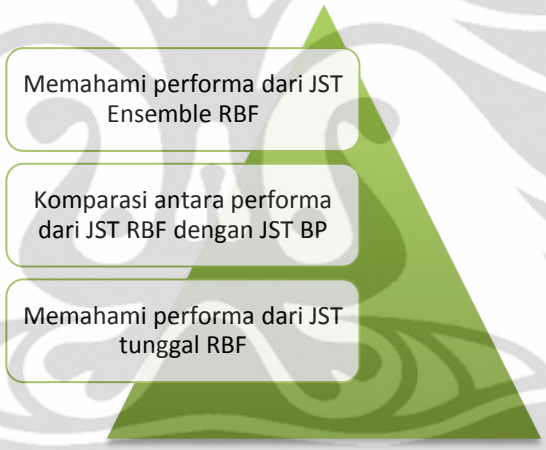
c. Konstanta pembelajaran spread (α_s) : 0.1

d. Konstanta momentum (μ) : 0.2

e. Konstanta penalti (λ) : 0.5
f. Kondisi stop *error* : 0.01
g. Kondisi stop epoh maksimum : 10000
h. Jumlah atribut (dimensi) hasil PCA : 30
i. Jumlah neuron tersembunyi pada JST BP adalah setengah dari dimensi masukan

2.11 Skema Percobaan

Pada percobaan yang dilakukan, terdapat sebuah skema percobaan yang digunakan sebagai acuan dalam kerangka percobaan. Skema percobaan yang digunakan pada percobaan ini dapat dilihat pada gambar 2.9.



Gambar 2.9. Skema Percobaan

Seperti yang terlihat pada gambar 2.9, skema penelitian yang dilakukan terdiri dari 3 tahapan. Berikut penjelasan untuk tiap tahapan:

1. Memahami performa dari JST tunggal RBF

   Percobaan pertama ini berisikan percobaan JST dengan algoritma RBF dengan menggunakan data-data yang telah disebutkan pada tabel 2.2 serta menggunakan fungsi *error* kuadratis dan *Cross-Entropy*. Berbeda dengan teori yang telah dibahas, pada percobaan ini algoritma BP yang digunakan di antara lapisan tersembunyi dan keluaran menggunakan bias. Hal ini dilakukan untuk mempermudah proses komputasi algoritma ini. Tujuan dari percobaan ini adalah untuk memahami perilaku dan performa dari JST tunggal dengan algoritma RBF.

2. Perbandingan antara JST BP dan JST RBF

   Percobaan kedua ini berisikan percobaan JSTBP dengan menggunakan data-data seperti yang telah disebutkan pada tabel 2.2 dan menggunakan fungsi *error* kuadratis dan *Cross-Entropy* sama layaknya seperti pada skema percobaan pertama. Tujuan dari percobaan ini adalah untuk membandingkan antara performa yang ditunjukkan oleh algoritma pelatihan RBF dengan performa yang ditunjukkan oleh algoritma pelatihan BP.

3. JST *Ensemble* RBF

   Percobaan ketiga ini berisikan percobaan JST *ensemble* dengan algoritma RBF dengan jumlah *ensemble* sebanyak 3 dan 5 dengan data yang telah disebutkan pada tabel 2.2 dan menggunakan fungsi *error* kuadratis dan *Cross-Entropy* untuk masing-masing jumlah *ensemble*. Sama halnya pada skema percobaan pertama, pada percobaan ini algoritma BP yang digunakan di antara lapisan tersembunyi dan keluaran untuk tiap jaringan menggunakan bias. Hal ini dilakukan untuk mempermudah proses komputasi algoritma ini.Percobaan ini dilakukan dengan tujuan mengoptimumkan kemampuan pengenalan dari JST tunggal RBF serta mengetahui performa dari JST *ensemble* RBF.

BAB 3
ANALISIS PERFORMA JST TUNGGAL

Pada bab ini akan dibahas mengenai hasil percobaan pertama yang dilakukan mengenai JST tunggal dengan algoritma permbelajaran *Radial Basis Function* (RBF). Secara garis besar, penulisan yang digunakan pada bab ini mengikuti aturan laporan percobaan yang meliputi tujuan yang hendak dicapai, prosedur percobaan yang dilakukan, algoritma program yang digunakan untuk memperoleh data hasil percobaan, hasil dari percobaan yang dilakukan, analisis hasil percobaan, dan kesimpulan yang diperoleh dari percobaan pada bab ini.

3.1 Tujuan Percobaan JST RBF Tunggal

Berikut ini merupakan tujuan yang hendak dicapai pada percobaan yang dilakukan pada bab ini:

1. memahami performa dari JST RBF dengan fungsi *error* kuadratis dan *Cross-Entropy* dengan data UCI dan wajah manusia; serta
2. membandingkan performa dari JST RBF dengan JST BP.

3.2 Prosedur Percobaan JST RBF Tunggal

Pada percobaan yang dilakukan, terdapat langkah-langkah atau prosedur yang dikembangkan guna mempermudah memahami jalannya percobaan. Berikut langkah-langkah percobaan yang digunakan:

Gambar 3.1. Prosedur Percobaan JST Tunggal

Seperti terlihat pada gambar 4.1, terdapat 5 tahap percobaan yang coba dikembangkan di mana setiap blok melambangkan aktivitas atau kegiatan yang dilakukan. Berikut penjelasan untuk masing-masing tahapan percobaan:

1. Pengumpulan data

   Tahap pengumpulan data adalah tahap di mana penulis mengumpulkan dan menyusun ulang data yang hendak digunakan. Seperti yang dijelaskan

pada butir 2.8, jumlah data yang digunakan pada percobaan ini adalah 11 data yang diperoleh dari 3 sumber berbeda, yaitu 9 data (nama data yang digunakan dapat diperoleh pada tabel 2.2) yang diperoleh dari badan bersitifikasi internasional UC *Irvine Machine Learning Repository* (UCI), satu set data citrawajah cahaya tampak yang diperoleh dari Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng. selaku pembimbing penulis, dan satu set datacitra wajah infra merah yang diperoleh dari laporan skripsi yang disusun oleh Stephen Roy Imantaka, ST. dengan judul Sistem Pengenal Wajah Berbasis Neural Network *Ensemble* untuk Citra Infra Merah (2010). Setelah mendapatkan seluruh data, data kemudian disusun ulang untuk kepentingan dan kemudahan akses data. Berikut contoh penataan data percobaan *Iris plants*:

```
5.1,3.5,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
```

$$\rightarrow \begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 & 1 \\ 7.0 & 3.2 & 4.7 & 1.4 & 2 \\ 6.3 & 3.3 & 6.0 & 2.5 & 3 \end{bmatrix}$$

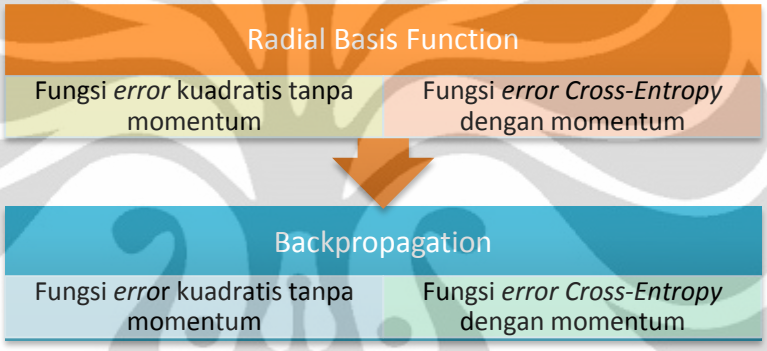
2. Pengolahan data

Langkah pengolahan data adalah tahap di mana data yang telah diperoleh dan telah mengalami penataan, diolah terlebih dahulu seperti yang telah dijelaskan pada butir 2.9 sebelum dilakukan pelatihan terhadapnya. Secara sederhana, tahap ini merupakan tahap menormalisasi data UCI dan melakukan PCA terhadap 2 data wajah lainnya di mana metode dan langkah normalisasi dan PCA telah dijelaskan pada butir 2.9. Setelah melalui proses normalisasi atau PCA, data kemudian dibagi menjadi 2 bagian, yaitu set data yang digunakan untuk pelatihan dan set data yang digunakan untuk pengujian. Banyaknya proporsi set data yang digunakan untuk set data pelatihan dan pengujian adalah 50:50. Berikut contoh tahap normalisasi terhadap set data *Iris plants* yang diperoleh dari tahap sebelumnya:

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 & 1 \\ 7.0 & 3.2 & 4.7 & 1.4 & 2 \\ 6.3 & 3.3 & 6.0 & 2.5 & 3 \end{bmatrix} \xrightarrow{z-score} \begin{bmatrix} -1.08 & 1.09 & -1.11 & -1.01 & 1 \\ 0.90 & 0.87 & 0.28 & 0.03 & 2 \\ 0.17 & -0.22 & 0.83 & 0.99 & 3 \end{bmatrix}$$

3. Pelatihan

Pelatihan adalah tahap di mana JST mempelajari data-data yang telah mengalami normalisasi atau PCA. Pada percobaan ini, terdapat 4 metode pelatihan yang dilakukan seperti yang dapat dilihat pada gambar 3.2 di mana untuk setiap masing-masing metode dilakukan 5 kali percobaan. Secara garis besar, terdapat 2 percobaan yang dilakukan, yaitu pelatihan JST dengan algoritma RBF dan pelatihan JST dengan algoritma BP di mana untuk tiap algoritma percobaan menggunakan masing-masing 2 buah fungsi *error*, yaitu fungsi *error* kuadratis dan *Cross-Entropy* yang diterapkan pada data pelatihan.



Gambar 3.2. MetodologiPercobaan JSTTunggal

Pada metode yang digunakan, terlihat bahwa untuk fungsi *error Cross-Entropy* terdapat tambahan berupa momentum. Momentum adalah suatu efek yang diberikan pada proses pelatihan dengan cara menambahkan besarnya nilai parameter pada saat $k-1$. Besarnya efek momentum ini ditentukan dengan menggunakan konstanta momentum (μ) yang besarnya dapat dilihat pada parameter percobaan (butir 2.10).

Hasil yang diperoleh dari tahap pelatihan adalah besarnya vektor nilai tengah RBF, lebar RBF serta bobot-bobot dan bias-bias JST yang akan digunakan pada tahap pengujian. Selain itu, *error* terkecil, banyaknya epoh pelatihan, serta waktu komputasi pelatihan juga diperoleh dari tahap ini.

4. Pengujian

Pengujian adalah tahap di mana nilai-nilai keluaran dari hasil pelatihan JST terhadap data pelatihan, diterapkan kembali terhadap set data pengujian untuk setiap percobaan. Hasil yang diperoleh dari tahap pengujian adalah

performa dari JST yang meliputi tingkat pengenalan terhadap data pengujian serta waktu komputasi dari proses pengujian.

5. Perbandingan

Perbandingan merupakan tahap di mana hasil percobaan pelatihan JST untuk kedua algoritma RBF dan BP, dibandingkan satu sama lain untuk fungsi *error* dan data percobaan yang berkesesuaian. Tujuan dari tahap ini adalah untuk mendapatkan perbandingan antara performa yang ditunjukkan algoritma pelatihan JST RBF dengan performa yang ditunjukkan algoritma pelatihan JST BP.

3.3 Algoritma Program JST RBF Tunggal

Berikut keempat algoritma yang digunakan untuk masing-masing metode pelatihan JST tunggal:

3.3.1 Algoritma JST RBF Fungsi *Error* Kuadratis

Berikut algoritma yang dikembangkan dalam percobaan JST RBF untuk fungsi *error* kuadratis:

A. Inisialisasi

a) Inisialisasi bobot dengan metode Nguyen-Widrow

1) Bias antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{l0} = \begin{bmatrix} w_{10} & w_{20} & \cdots & w_{L0} \end{bmatrix}$$

2) Bobot antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{lm} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L1} & w_{L2} & \cdots & w_{LM} \end{bmatrix}$$

b) Vektor masukan

$$\boldsymbol{x}_{pn} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{P1} & x_{P2} & \cdots & x_{PN} \end{bmatrix}$$

c) Vektor target

$$\boldsymbol{T}_{pl} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1L} \\ t_{21} & t_{22} & \cdots & t_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ t_{P1} & t_{P2} & \cdots & t_{PL} \end{bmatrix}$$

d) Vektor nilai tengah RBF yang diambil dari rata-rata tiap kelas

$$\boldsymbol{c}_{mn} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{M1} & c_{M2} & \cdots & c_{MN} \end{bmatrix}$$

e) Lebar RBF dengan menggunakan persamaan berikut:

1) Untuk $m = 1 - M$, lakukan langkah berikut:

1. $\sigma_m = \dfrac{d_{\max}}{\sqrt{k}}$

B. Untuk epoh < *epoh_max* dan err_total > *err_max*, lakukan langkah berikut:

a) Untuk $p = 1 - P$, lakukan langkah berikut:

1) Perhitungan lapisan tersembunyi

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) $zin_m = \sum_{n=1}^{N} \dfrac{\left(x_{pn} - c_{mn}\right)^2}{2{\sigma_m}^2}$

ii) $z_m = \exp\left(-zin_m\right)$

2) Perhitungan pada lapisan keluaran

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $yin_l = w_{l0} + \sum_{m=1}^{M} z_m w_{lm}$

ii) $y_l = \dfrac{1}{1 + \exp\left(-yin_l\right)}$

3) Perhitungan *error*

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $e_l = y_l - T_{pl}$

b) $E_p = \sum_{l=1}^{L} {e_l}^2$

4) Pembelajaran bobot dan bias antara lapisan keluaran dengan tersembunyi

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $Sy_l = y_l(1 - y_l)$

ii) $dk_l = (T_{pl} - y_l)Sy_l$

iii) Untuk $m = 1 - M$, lakukan langkah berikut:

(1) $dw_{lm} = \alpha \times dk_l \times z_m$

iv) $dw_{l0} = \alpha \times dk_l$

5) Pembelajaran vektor nilai tengah dan lebar RBF

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) Untuk $n = 1 - N$, lakukan langkah berikut:

(1) $dc_{mn} = \alpha_c \dfrac{\sum_{l=1}^{L} dk_l w_{lm}}{\sigma^2} z_m (x_{pn} - c_{mn})$

b) $d\sigma_m = \alpha_c \dfrac{\sum_{l=1}^{L} dk_l w_{lm}}{\sigma^3} z_m \sum_{n=1}^{N} (x_{pn} - c_{mn})^2$

6) Perubahan parameter

a) $\boldsymbol{w}_{lm} = \boldsymbol{w}_{lm} + d\boldsymbol{w}_{lm}$

b) $\boldsymbol{w}_{l0} = \boldsymbol{w}_{l0} + d\boldsymbol{w}_{l0}$

c) $\boldsymbol{c}_{mn} = \boldsymbol{c}_{mn} + d\boldsymbol{c}_{mn}$

d) $\boldsymbol{\sigma}_{mn} = \boldsymbol{\sigma}_{mn} + d\boldsymbol{\sigma}_{mn}$

C. Kondisi henti

$$\sum_{p=1}^{P} \sum_{l=1}^{L} e_l^{\,2} \le 0.01$$

3.3.2 Algoritma JST RBF Fungsi *Error Cross-entropy*

Berikut algoritma yang dikembangkan dalam percobaan JST RBF untuk fungsi *error Cross-entropy* dengan momentum:

A. Inisialisasi

a) Inisialisasi bobot dengan metode Nguyen-Widrow

1) Bias antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{l0} = \begin{bmatrix} w_{10} & w_{20} & \cdots & w_{L0} \end{bmatrix}$$

2) Bobot antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{lm} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L1} & w_{L2} & \cdots & w_{LM} \end{bmatrix}$$

b) Vektor masukan

$$\boldsymbol{x}_{pn} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{P1} & x_{P2} & \cdots & x_{PN} \end{bmatrix}$$

c) Vektor target

$$\boldsymbol{T}_{pl} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1L} \\ t_{21} & t_{22} & \cdots & t_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ t_{P1} & t_{P2} & \cdots & t_{PL} \end{bmatrix}$$

d) Vektor nilai tengah RBF yang diambil dari rata-rata tiap kelas

$$\boldsymbol{c}_{mn} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{M1} & c_{M2} & \cdots & c_{MN} \end{bmatrix}$$

e) Lebar RBF dengan menggunakan persamaan berikut:

1) Untuk $m = 1 - M$, lakukan langkah berikut:

1. $\sigma_m = \dfrac{d_{\max}}{\sqrt{k}}$

B. Untuk epoh < *epoh_max* dan err_total > *err_max*, lakukan langkah berikut:

a) Untuk $p = 1 - P$, lakukan langkah berikut:

1) Perhitungan lapisan tersembunyi

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) $zin_m = \sum_{n=1}^{N} \dfrac{\left(x_{pn} - c_{mn}\right)^2}{2{\sigma_m}^2}$

ii) $z_m = \exp(-zin_m)$

2) Perhitungan pada lapisan keluaran

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $yin_l = w_{l0} + \sum_{m=1}^{M} z_m w_{lm}$

ii) $y_l = \frac{1}{1+\exp(-yin_l)}$

3) Perhitungan *error*

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $e_l = -T_{pl}\ln(y_l) - (1-T_{pl})\ln(1-y_l)$

b) $E_p = \sum_{l=1}^{L} e_l^{\,2}$

4) Pembelajaran bobot dan bias antara lapisan keluaran dengan tersembunyi

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $dk_l = (T_{pl} - y_l)$

ii) Untuk $m = 1 - M$, lakukan langkah berikut:

(1) $dw_{lm}(k) = \alpha \times dk_l \times z_m + \mu \times dw_{lm}(k-1)$

iii) $dw_{l0}(k) = \alpha \times dk_l + \mu \times dw_{l0}(k-1)$

5) Pembelajaran vektor nilai tengah dan lebar RBF

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) Untuk $n = 1 - N$, lakukan langkah berikut:

(1) $dc_{mn}(k) = \alpha_c \frac{\sum_{l=1}^{L} dk_l w_{lm}}{\sigma^2} z_m (x_{pn} - c_{mn}) + \mu \times dc_{mn}(k-1)$

b) $d\sigma_m(k) = \alpha_c \frac{\sum_{l=1}^{L} dk_l w_{lm}}{\sigma^3} z_m \sum_{n=1}^{N} (x_{pn} - c_{mn})^2 + \mu \times d\sigma_m(k-1)$

6) Perubahan parameter

a) $\boldsymbol{w}_{lm} = \boldsymbol{w}_{lm} + d\boldsymbol{w}_{lm}$

b) $\boldsymbol{w}_{l0} = \boldsymbol{w}_{l0} + d\boldsymbol{w}_{l0}$

c) $\boldsymbol{c}_{mn} = \boldsymbol{c}_{mn} + d\boldsymbol{c}_{mn}$

d) $\boldsymbol{\sigma}_{mn} = \boldsymbol{\sigma}_{mn} + d\boldsymbol{\sigma}_{mn}$

C. Kondisi henti

$$\sum_{p=1}^{P}\sum_{l=1}^{L} e_l^{\,2} \le 0.01$$

3.3.3 Algoritma JST BP Fungsi *Error* Kuadratis

Berikut algoritma yang dikembangkan dalam percobaan JST BP untuk fungsi *error* Kuadratis:

A. Inisialisasi

a) Inisialisasi bobot dengan metode Nguyen-Widrow

1) Bias antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{v}_{m0} = \begin{bmatrix} v_{10} & v_{20} & \cdots & v_{M0} \end{bmatrix}$$

2) Bobot antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{v}_{lm} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MN} \end{bmatrix}$$

3) Bias antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{l0} = \begin{bmatrix} w_{10} & w_{20} & \cdots & w_{L0} \end{bmatrix}$$

4) Bobot antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{lm} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L1} & w_{L2} & \cdots & w_{LM} \end{bmatrix}$$

b) Vektor masukan

$$\boldsymbol{x}_{pn} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{P1} & x_{P2} & \cdots & x_{PN} \end{bmatrix}$$

c) Vektor target

$$\boldsymbol{T}_{pl} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1L} \\ t_{21} & t_{22} & \cdots & t_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ t_{P1} & t_{P2} & \cdots & t_{PL} \end{bmatrix}$$

d) $\alpha = 0.1$

B. Untuk epoh < *epoh_max* dan err_total > *err_max*, lakukan langkah berikut:

a) Untuk $p = 1 - P$, lakukan langkah berikut:

1) Perhitungan lapisan tersembunyi

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) $zin_m = v_{m0} + \sum_{n=1}^{N} x_{pn} v_{mn}$

ii) $z_m = \dfrac{1}{1 + \exp(-zin_m)}$

2) Perhitungan pada lapisan keluaran

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $yin_l = w_{l0} + \sum_{m=1}^{M} z_m w_{lm}$

ii) $y_l = \dfrac{1}{1 + \exp(-yin_l)}$

3) Perhitungan *error*

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $e_l = y_l - T_{pl}$

b) $E_p = \sum_{l=1}^{L} e_l^{\,2}$

4) Pembelajaran bobot dan bias antara lapisan keluaran dengan tersembunyi

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $Sy_l = y_l(1 - y_l)$

ii) $dk_l = (T_{pl} - y_l)Sy_l$

iii) Untuk $m = 1 - M$, lakukan langkah berikut:

(1) $dw_{lm} = \alpha \times dk_l \times z_m$

iv) $dw_{l0} = \alpha \times dk_l$

5) Pembelajaran bobot dan bias antara lapisan tersembunyi dengan masukan

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) $Sm_m = z_m(1 - z_m)$

ii) $dm_m = \sum_{l=1}^{L}(w_{lm} dk_l) Sm_m$

iii) Untuk $n = 1 - N$, lakukan langkah berikut:

(1) $dv_{mn} = \alpha \times dm_m \times x_{pn}$

iv) $dv_{m0} = \alpha \times dm_m$

6) Perubahan parameter

a) $\boldsymbol{v}_{mn} = \boldsymbol{v}_{mn} + d\boldsymbol{v}_{mn}$

b) $\boldsymbol{v}_{m0} = \boldsymbol{v}_{m0} + d\boldsymbol{v}_{m0}$

c) $\boldsymbol{w}_{lm} = \boldsymbol{w}_{lm} + d\boldsymbol{w}_{lm}$

d) $\boldsymbol{w}_{l0} = \boldsymbol{w}_{l0} + d\boldsymbol{w}_{l0}$

C. Kondisi henti

$$\sum_{p=1}^{P} E_p \leq 0.01$$

3.3.4 Algoritma JST BP Fungsi *Error Cross-entropy*

Berikut algoritma yang dikembangkan dalam percobaan JST BP untuk fungsi *error Cross-entropy* dengan momentum:

A. Inisialisasi

a) Inisialisasi bobot dengan metode Nguyen-Widrow

1) Bias antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{v}_{m0} = \begin{bmatrix} v_{10} & v_{20} & \cdots & v_{M0} \end{bmatrix}$$

2) Bobot antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{v}_{lm} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MN} \end{bmatrix}$$

3) Bias antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{l0} = \begin{bmatrix} w_{10} & w_{20} & \cdots & w_{L0} \end{bmatrix}$$

4) Bobot antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{lm} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L1} & w_{L2} & \cdots & w_{LM} \end{bmatrix}$$

b) Vektor masukan

$$\boldsymbol{x}_{pn} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{P1} & x_{P2} & \cdots & x_{PN} \end{bmatrix}$$

c) Vektor target

$$\boldsymbol{T}_{pl} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1L} \\ t_{21} & t_{22} & \cdots & t_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ t_{P1} & t_{P2} & \cdots & t_{PL} \end{bmatrix}$$

d) $\alpha = 0.1$

B. Untuk epoh < *epoh_max* dan err_total > *err_max*, lakukan langkah berikut:

a) Untuk $p = 1 - P$, lakukan langkah berikut:

1) Perhitungan lapisan tersembunyi

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) $$zin_m = v_{m0} + \sum_{n=1}^{N} x_{pn} v_{mn}$$

ii) $$z_m = \frac{1}{1 + \exp(-zin_m)}$$

2) Perhitungan pada lapisan keluaran

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $$yin_l = w_{l0} + \sum_{m=1}^{M} z_m w_{lm}$$

ii) $$y_l = \frac{1}{1 + \exp(-yin_l)}$$

3) Perhitungan *error*

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $e_l = -T_{pl}\ln(y_l) - (1 - T_{pl})\ln(1 - y_l)$

b) $E_p = \sum_{l=1}^{L} e_l^{\,2}$

4) Pembelajaran bobot dan bias antara lapisan keluaran dengan tersembunyi

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $dk_l = (T_{pl} - y_l)$

ii) Untuk $m = 1 - M$, lakukan langkah berikut:

(1) $dw_{lm}(k) = \alpha \times dk_l \times z_m + \mu \times dw_{lm}(k-1)$

iii) $dw_{l0}(k) = \alpha \times dk_l + \mu \times dw_{l0}(k-1)$

5) Pembelajaran bobot dan bias antara lapisan tersembunyi dengan masukan

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) $Sm_m = z_m(1 - z_m)$

ii) $dm_m = \sum_{l=1}^{L}(w_{lm} dk_l) Sm_m$

iii) Untuk $n = 1 - N$, lakukan langkah berikut:

(1) $dv_{mn}(k) = \alpha \times dm_m \times x_{pn} + \mu \times dv_{mn}(k-1)$

iv) $dv_{m0}(k) = \alpha \times dm_m + \mu \times dv_{m0}(k-1)$

6) Perubahan parameter

a) $\boldsymbol{v}_{mn} = \boldsymbol{v}_{mn} + d\boldsymbol{v}_{mn}$

b) $\boldsymbol{v}_{m0} = \boldsymbol{v}_{m0} + d\boldsymbol{v}_{m0}$

c) $\boldsymbol{w}_{lm} = \boldsymbol{w}_{lm} + d\boldsymbol{w}_{lm}$

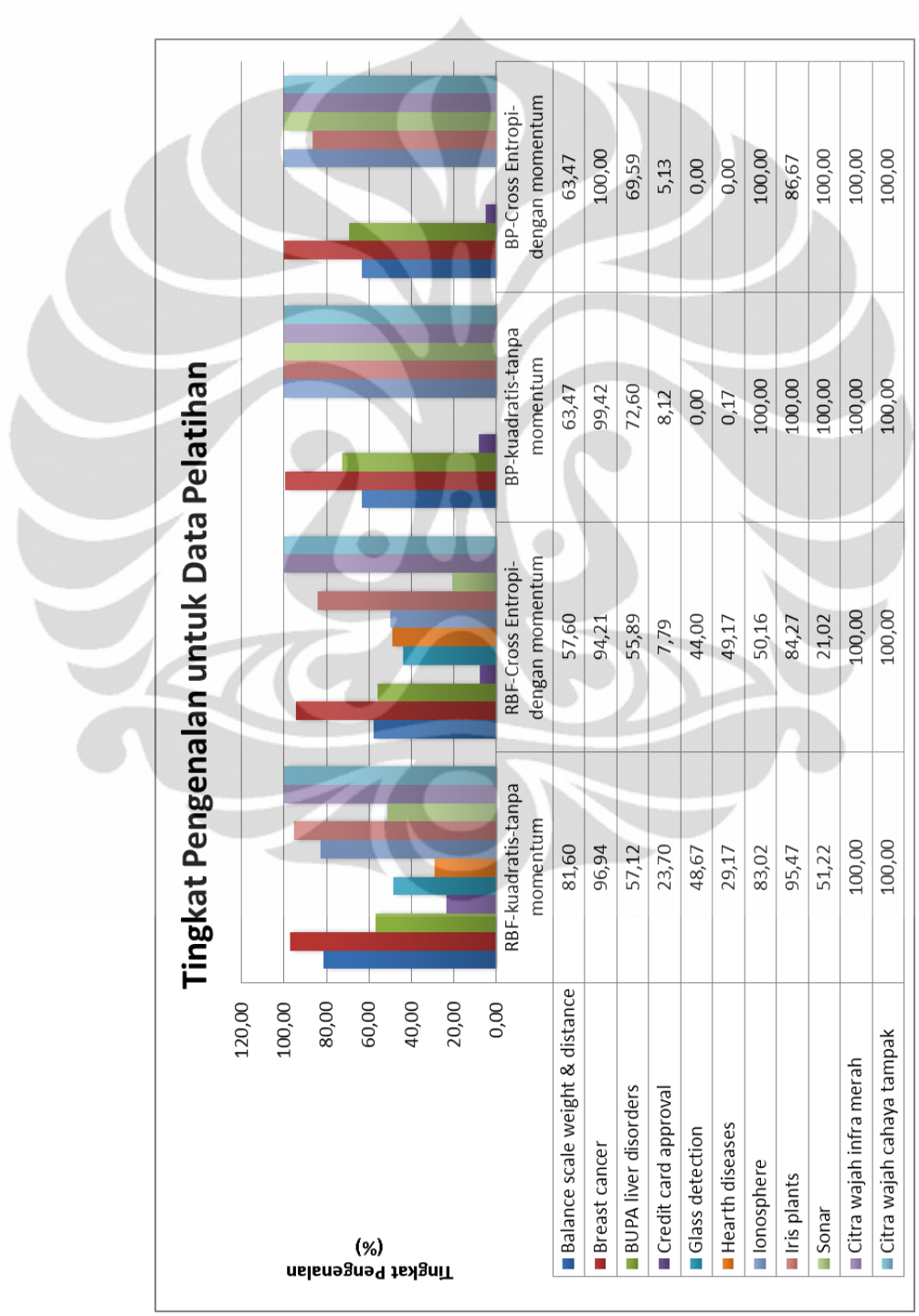
d) $\boldsymbol{w}_{l0} = \boldsymbol{w}_{l0} + d\boldsymbol{w}_{l0}$

C. Kondisi henti

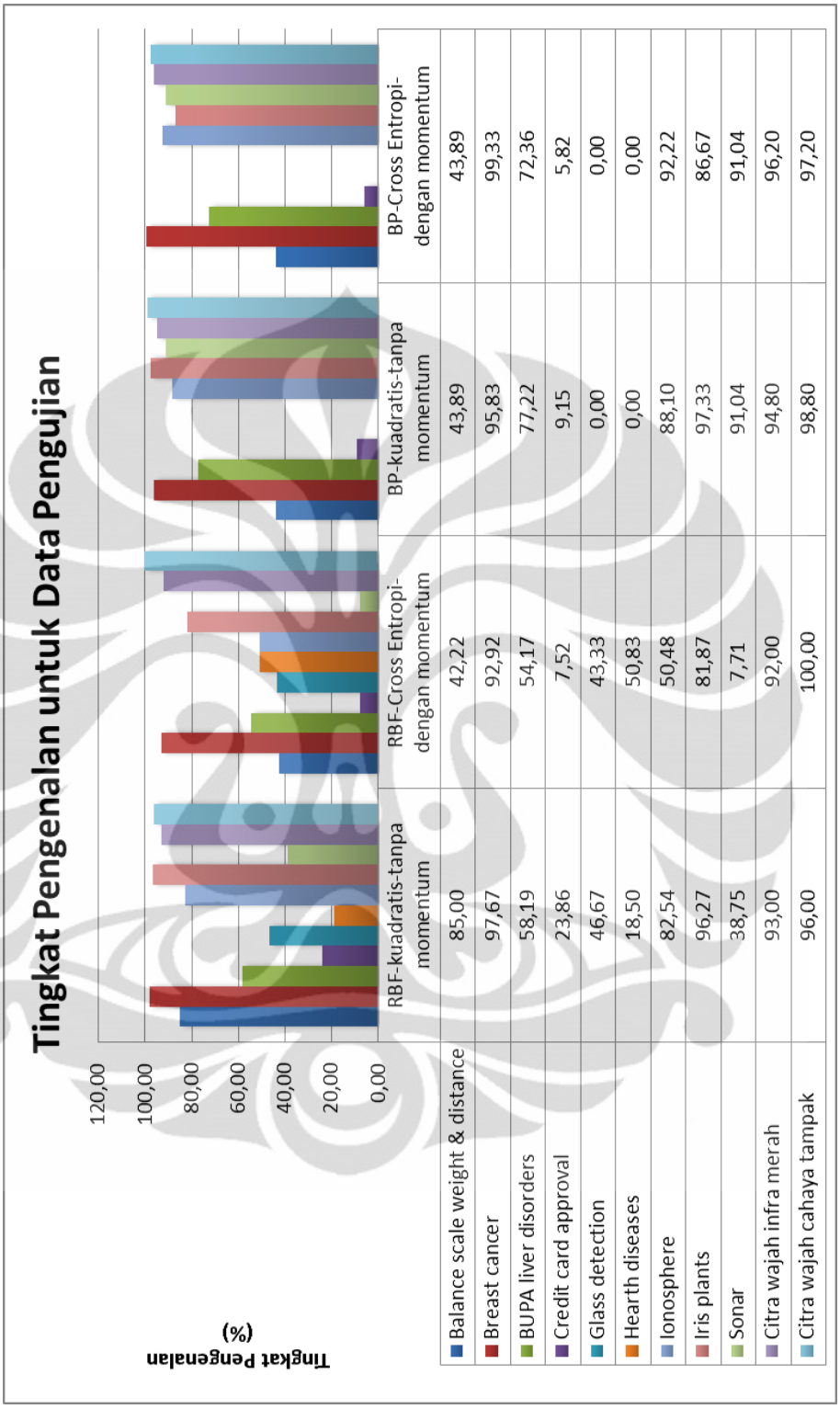
$$\sum_{p=1}^{P} E_p \le 0.01$$

3.4 Hasil Percobaan JST RBF Tunggal

Pada bagian ini akan ditampilkan hasil percobaan JST RBF dan JST BP dengan menggunakan kedua fungsi *error*. Hasil percobaan yang ditampilkan merupakan rata-rata dari 5 kali percobaan untuk masing-masing metode yang terdapat pada gambar 3.3 hingga 3.8.

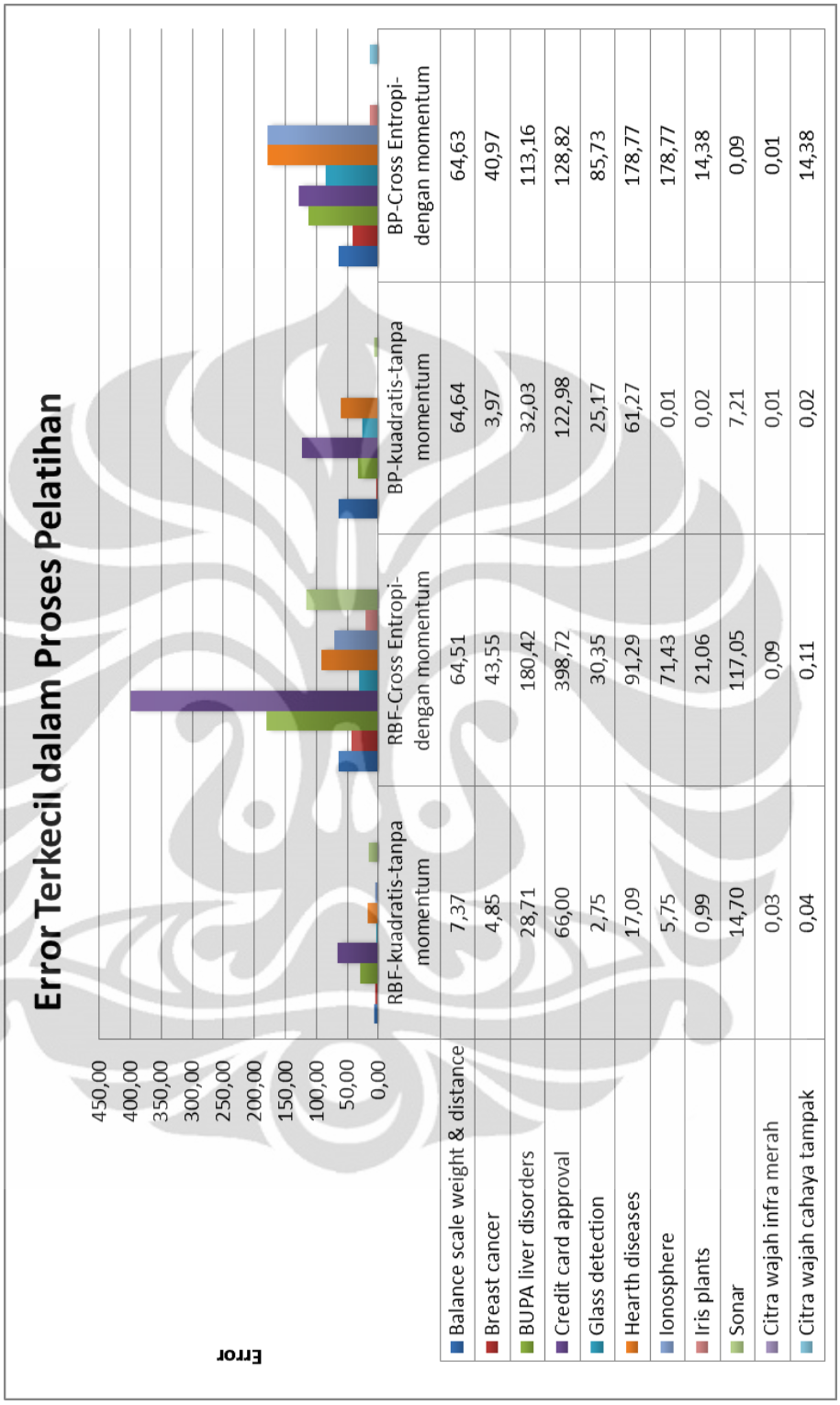


Gambar 3.3. Tingkat Pengenalan untuk Data Pelatihan Pada JST Tunggal

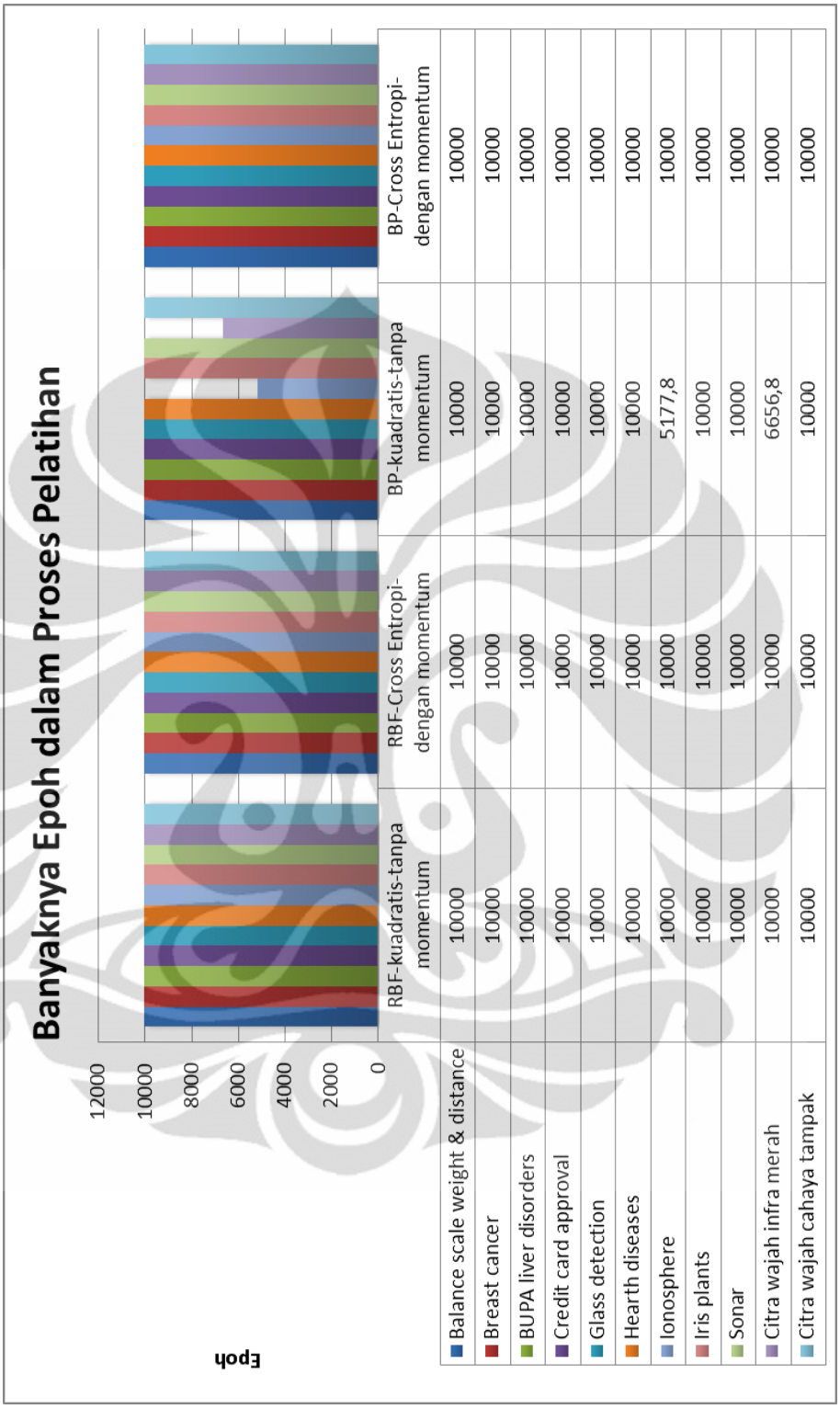


	RBF-kuadratis-tanpa momentum	RBF-Cross Entropi-dengan momentum	BP-kuadratis-tanpa momentum	BP-Cross Entropi-dengan momentum
Balance scale weight & distance	85,00	42,22	43,89	43,89
Breast cancer	97,67	92,92	95,83	99,33
BUPA liver disorders	58,19	54,17	77,22	72,36
Credit card approval	23,86	7,52	9,15	5,82
Glass detection	46,67	43,33	0,00	0,00
Hearth diseases	18,50	50,83	0,00	0,00
Ionosphere	82,54	50,48	88,10	92,22
Iris plants	96,27	81,87	97,33	86,67
Sonar	38,75	7,71	91,04	91,04
Citra wajah infra merah	93,00	92,00	94,80	96,20
Citra wajah cahaya tampak	96,00	100,00	98,80	97,20

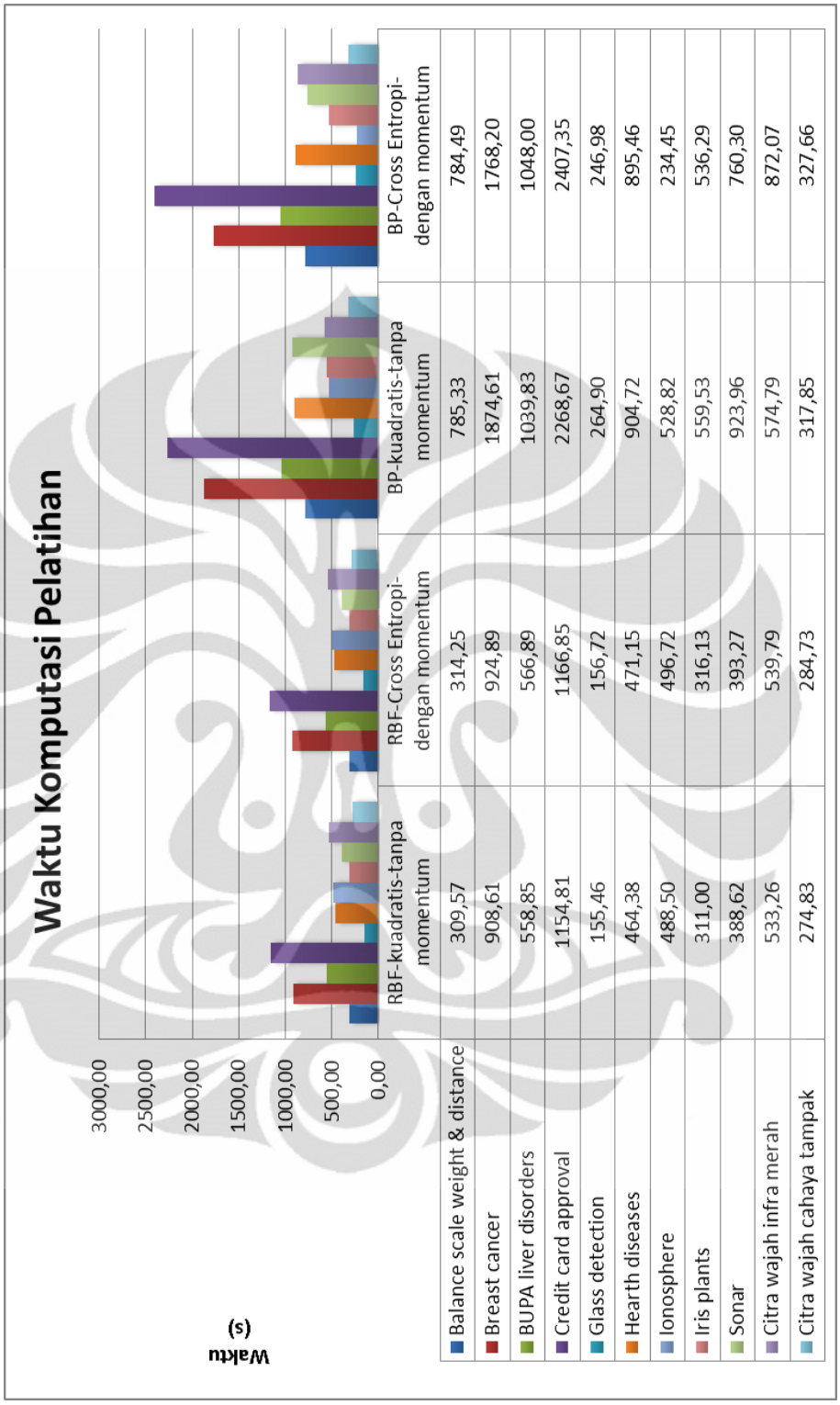
Gambar 3.4. Tingkat Pengenalan untuk Data Pengujian pada JST Tunggal



Gambar 3.5. Error Terkecil dalam Proses Pelatihan JST Tunggal

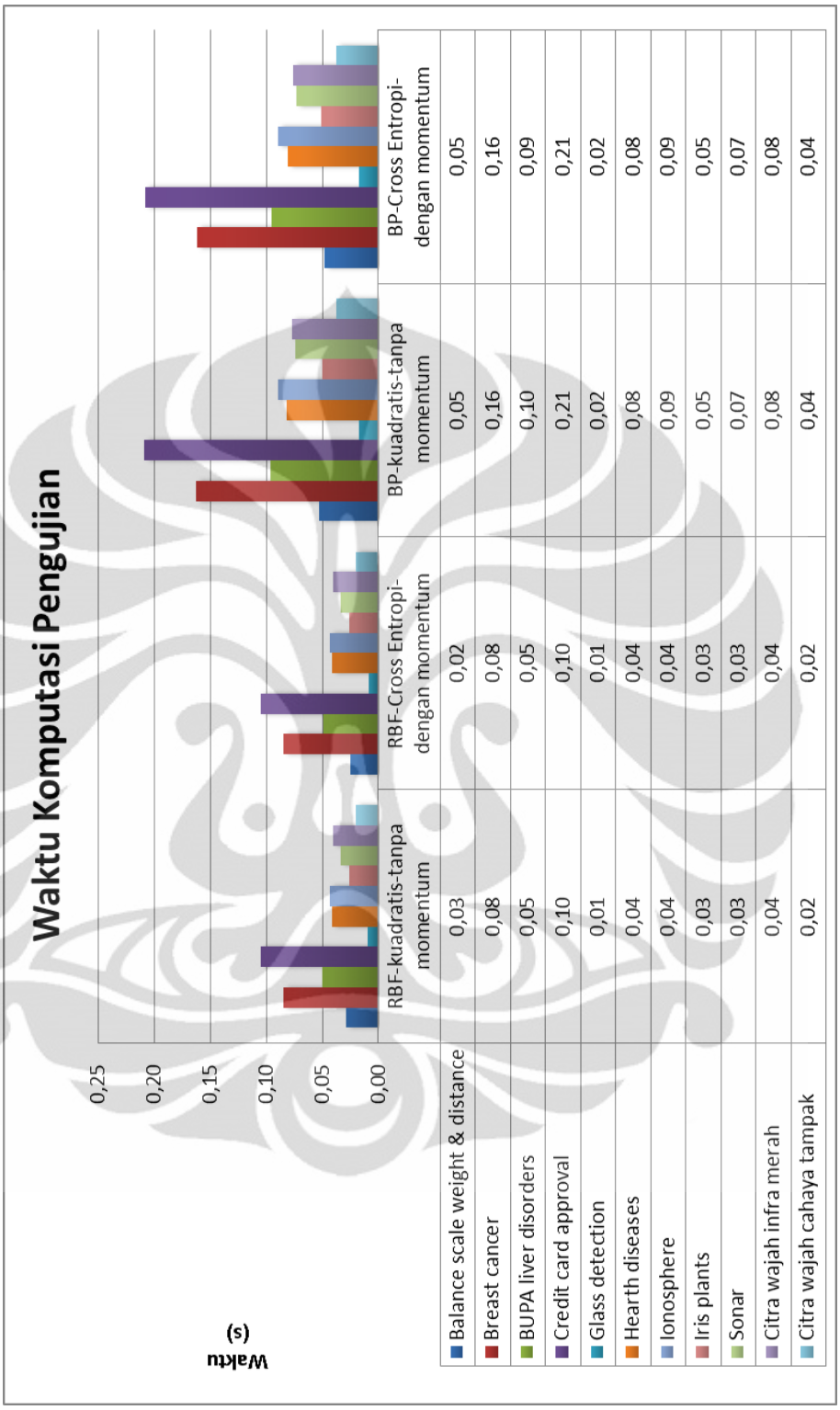


Gambar 3.6. Banyaknya Epoh dalam Proses Pelatihan JST Tunggal



	RBF-kuadratis-tanpa momentum	RBF-Cross Entropi-dengan momentum	BP-kuadratis-tanpa momentum	BP-Cross Entropi-dengan momentum
Balance scale weight & distance	309,57	314,25	785,33	784,49
Breast cancer	908,61	924,89	1874,61	1768,20
BUPA liver disorders	558,85	566,89	1039,83	1048,00
Credit card approval	1154,81	1166,85	2268,67	2407,35
Glass detection	155,46	156,72	264,90	246,98
Hearth diseases	464,38	471,15	904,72	895,46
Ionosphere	488,50	496,72	528,82	234,45
Iris plants	311,00	316,13	559,53	536,29
Sonar	388,62	393,27	923,96	760,30
Citra wajah infra merah	533,26	539,79	574,79	872,07
Citra wajah cahaya tampak	274,83	284,73	317,85	327,66

Gambar 3.7. Waktu Komputasi Pelatihan JST Tunggal



Gambar 3.8. Waktu Komputasi Pengujian JST Tunggal

3.5 Analisis Performa JST RBF dan JST BP

Setelah mendapatkan hasil dari percobaan, penulis akan mencoba untuk menganalisis hasil yang diperoleh dari percobaan untuk mendapatkan pengetahuan mengenai penyebab dan perbandingannya dengan metode lainnya. Dalam analisis ini, penulis akan menganalisis hasil dari tingkat pengenalan dan waktu komputasi. Selain itu, akan dilakukan analisis uji logika dan analisis pengaruh dari fungsi *error* terhadap performa dari JST RBF sebagai penunjang analisis.

3.5.1 Analisis Uji Logika

Sebelum melakukan analisis lebih lanjut, akan dibahas mengenai uji logika terhadap RBF dengan kedua fungsi *error* kuadratis dan *Cross-entropy*. Uji logika merupakan sebuah tes yang menggunakan data-data Boolean untuk mengetahui kemampuan jaringan untuk mengenali data-data yang ambigu. Pada uji ini, data yang digunakan dapat dilihat pada tabel 3.1.

Tabel 3.1. Data Tes Logika

N	x_1	x_2	x_3	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	t_1	t_2
1	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
2	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
3	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
4	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0
5*	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0
6*	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0
7*	0.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0
8*	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0
9*	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
10*	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0
11*	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
12*	0.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0

Catatan : * menandakan data dengan target yang berlawanan

Dengan menggunakan data Boolean yang ditunjukkan pada tabel 3.1 sebagai data pelatihan JST RBF, diharapkan performa JST RBF terhadap data-data ambigu dapat diketahui. Dari hasil pelatihan JST terhadap data pada tabel 3.1, diperoleh hasil berikut:

Tabel 3.2. Hasil Uji Logika JST RBF Tunggal

N	Kuadratis		*Cross-entropy*	
	y_1	y_2	y_1	y_2
1	0.9329 ■	0.0670	0.9165 ■	0.0835
2	0.9331 ■	0.0669	0.9200 ■	0.0800
3	0.9328 ■	0.0671	0.9129 ■	0.0871
4	0.9327 ■	0.0673	0.9090 ■	0.0910
5	0.5200	0.4800	0.5246	0.4754
6	0.5200	0.4800	0.5246	0.4754
7	0.5209	0.4791	0.5383	0.4617
8	0.5209	0.4791	0.5383	0.4617
9	0.5217	0.4783	0.5517	0.4483
10	0.5217	0.4783	0.5517	0.4483
11	0.5226	0.4774	0.5651	0.4349
12	0.5226	0.4774	0.5651	0.4349

Catatan: ■ menandakan data yang dikenali

Pada tabel 3.2, dapat dilihat bahwa JST RBF baik digunakan untuk set data yang tidak ambigu. Hal ini terbukti dari dapat dikenalinya dengan baik data nomor 1 hingga 4 yang merupakan data-data non-ambigu sedangkan untuk data-data ambigu, JST RBF belum mampu untuk mengenali baik pada fungsi *error* kuadratis maupun fungsi *error Cross-entropy*.

Untuk memahami penyebab ketidak mampuan JST RBF untuk mengenali data-data ambigu, diperlukan analisis mengenai 2 metode yang digunakan pada metode pelatihan JST RBF. Pada dasarnya, algoritma pelatihan dengan menggunakan RBF merupakan gabungan dari 2 metode, yaitu fungsi Radial Basis dan *backpropagation*. Fungsi Radial Basis merupakan fungsi untuk mencari jarak antar 2 nilai. Pada percobaan yang dilakukan, fungsi Radial Basis yang digunakan adalah fungsi Gaussian yang memiliki nilai antara 0 dan 1. Semakin dekat jarak antar 2 nilai, maka semakin dekat nilai keluaran fungsi ini dengan nilai 1. Sedangkan semakin jauh jarak antar 2 nilai, maka semakin dekat nilai keluaran fungsi ini dengan nilai 0. Dengan demikian, pengaruh dengan digunakannya fungsi Radial Basis di lapisan awal adalah untuk mencari nilai Gaussian antara data dan nilai tengah. Dengan kata lain, keluaran dari lapisan ini berupa nilai seberapa dekat suatu data dengan suatu nilai tengah.

Metode kedua yang digunakan setelah metode fungsi Radial Basis adalah *backpropagation*. Seperti yang telah dibahas, metode Backpropagation terkenal akan kemampuannya mempelajari pola-pola. Dengan digunakannya metode ini setelah metode Radial Basis, membuat JST mampu untuk mengenali suatu pola berdasarkan jaraknya dengan nilai tengah.

Permasalahan dari digabungnya kedua metode ini akan muncul apabila data yang digunakan memiliki masalah ambiguitas seperti yang ditunjukkan oleh data 5 dan 6, 7 dan 8, 9 dan 10, serta 11 dan 12. Seperti yang telah dijelaskan, penerapan fungsi Radial Basis untuk data-data jenis ini akan menghasilkan nilai yang kurang lebih sama namun memiliki target pelatihan yang berbeda. Keluaran yang bernilai sama dari metode Radial Basis ini selanjutnya harus dibedakan oleh metode *backpropagation*. Karena kesamaan dari 2 nilai, membuat metode *backpropagation* sulit untuk membedakan kedua data ini walaupun telah diberikan target pelatihan maupun fungsi *error* yang berbeda.

3.5.2 Analisis Pengaruh Fungsi *Error*

Pada proses pelatihan JST terhadap suatu set data, akan terdapat *error* yang harus diminimalisasi untuk meningkatkan performa JST. Semakin kecil *error* yang dihasilkan, maka semakin baik performa JST. Namun untuk meminimalisasi *error*, dibutuhkan pengetahuan mengenai distribusi dari *error* yang dihasilkan data tersebut. Permasalahan yang timbul adalah sulitnya untuk mengetahui distribusi *error* karena tidak dapat dilakukannya analisis statistik.

Karena sulitnya mengetahui distribusi *error* dari data pelatihan, maka dilakukan asumsi-asumsi guna mendekati distribusi *error* yang sebenarnya. Secara umum, distribusi *error* yang paling banyak ditemukan adalah distribusi Gaussian. Oleh karena itu, fungsi *error* yang tepat digunakan adalah *Mean Square Error* (MSE) atau dalam bahasa Indonesia dikenal dengan fungsi *error* kuadratis. Selain fungsi *error* kuadratis, fungsi *error* lainnya yang umum digunakan adalah fungsi *error Cross-entropy*. Apabila asumsi distribusi *error* pelatihan dapat didekati dengan baik oleh fungsi probabilitas *Likelihood*, maka fungsi *error Cross-entropy* merupakan fungsi *error* yang paling tepat diterapkan.

Dari pembahasan sebelumnya, maka dapat diambil kesimpulan pengaruh dari fungsi *error* sangatlah dominan terhadap performa yang ditunjukkan JST. Namun pemilihan fungsi *error* sangat bergantung dari data yang hendak dipelajari. Dengan demikian, pengaruh dari fungsi *error* terhadap hasil percobaan yang dilakukan tidak dapat ditebak dan dianalisis tetapi hanya dapat disimpulkan.

Karena fungsi *error* digunakan di dalam proses pelatihan, maka suatu data lebih baik didekati dengan fungsi *error* kuadratis atau *Cross-entropy* akan dilakukan berdasarkan performa tingkat pengenalan data pelatihan. Dari gambar 3.3, set data yang menunjukkan performa yang lebih baik ketika didekati dengan fungsi *error Cross-entropy* adalah data *Hearth diseases*. Sedangkan data lainnya menunjukkan pendekatan *error* Kuadratis menghasilkan performa yang lebih baik. Namun demikian, kualitas performa tingkat pengenalan data pelatihan tidak menunjukkan hasil yang maksimal. Hal ini mengindikasikan bahwa terdapat kesalahan pendekatan fungsi *error* ataupun dari segi data percobaan yang terlalu rumit untuk dikenali.

Dari segi waktu komputasi, pemilihan fungsi *error* juga turut menentukan lamanya waktu komputasi. Sama halnya dengan tingkat pengenalan, pengaruh dari fungsi *error* juga hanya dapat dilihat pada waktu komputasi pelatihan. Dari gambar 3.7, terlihat waktu komputasi pelatihan dengan fungsi *error Cross-entropy* lebih lama dibandingkan dengan fungsi *error* kuadratis. Secara teoritis, perhitungan yang ditunjukkan oleh fungsi *error Cross-entropy* lebih sulit dibandingkan dengan fungsi *error* kuadratis. Pada percobaan ini, fungsi *error Cross-entropy* juga dilengkapi dengan momentum. Dengan demikian, pelatihan dengan fungsi *error Cross-entropy* lebih lama dibandinkan dengan fungsi *error* kuadratis.

3.5.3 Analisis Tingkat Pengenalan

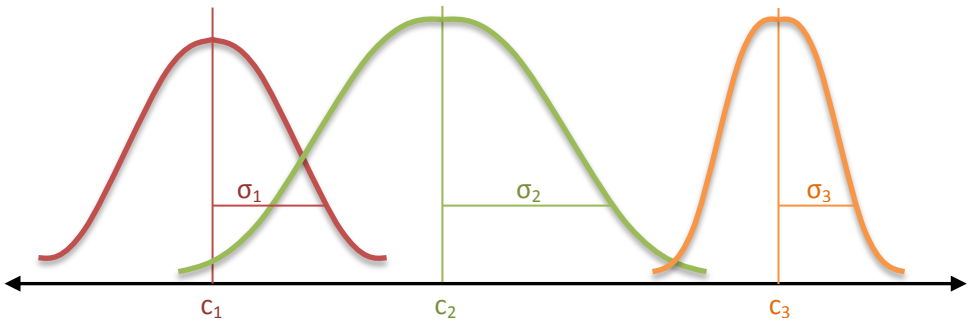
Sebelum melakukan analisis lebih lanjut, terlebih dahulu akan dilakukan analisis pengelompokan data dengan menggunakan metode *Self-Organizing Maps* (SOM) di mana penjelasannya terdapat pada lampiran D. Tujuan dilakukannya analisis SOM adalah untuk mengetahui ambiguitas di dalam setiap setiap set data

percobaan. Hasil percobaan dengan menggunakan metode SOM dapat dilihat pada lampiran E.

Pada lampiran E, terlihat untuk semua set data UCI dan set data citra wajah infra merah memiliki ambiguitas pada kedua set data pelatihan dan pengujian, sedangkan set data citra wajah manusia cahaya tampak tidak mengalami ambiguitas pada kedua data pelatihan dan pengujian. Dari gambar 3.3, dapat dilihat bahwa secara keseluruhan hasil tingkat pengenalan terbaik diperoleh dari data ke-10 dan ke-11, yaitu set data citra wajah infra merah dan set data citra wajah cahaya tampak, namun pada gambar 3.4 dapat diperoleh fakta bahwa data ke-11 menunjukkan hasil yang terbaik. Dari hasil ini, dapat dikatakan bahwa untuk mendapatkan performa JST RBF yang tinggi, diperlukan set data pelatihan yang tidak ambigu. Hal ini berlaku untuk kedua fungsi *error* kuadratis dan *Cross-entropy*.

Pada gambar 3.3 dan 3.4, secara umum dapat dikatakan bahwa terdapat data-data yang secara SOM memiliki sifat ambiguitas namun memiliki tingkat pengenalan set dapat pelatihan yang baik. Definisi tingkat pengenalan yang baik adalah apabila tingkat pengenalan yang dihasilkan lebih dari 90%. Data-data yang dimaksud adalah set data *Breast cancer*, *Iris plants*, dan citra wajah infra merah. Dengan demikian, terdapat faktor-faktor lain yang juga mempengaruhi tingkat pengenalan selain tingkat ambiguitas.

Faktor lainnya yang juga mempengaruhi tingkat pengenalan data pelatihan adalah interaksi antar suatu kelas terhadap kelas lainnya. Sebagai ilustrasi, dapat dilihat pada gambar 3.9.



Gambar 3.9. Ilustrasi Interaksi Antar Kelas

Untuk dapat membedakan 2 buah data, maka perbedaan yang ada harus jelas terlihat. Pada dunia statistik, hal ini dapat dilihat dengan menggunakan nilai rata-rata dan besar penyimpangan suatu kelompok data. Sebagai contoh, apabila terdapat 3 buah set data yang dapat secara statistik dapat digambarkan seperti gambar 3.9. Dari gambar tersebut, set data yang terletak di antara nilai tengah kelas 1 dan 2 akan mengalami ambiguitas yang lebih besar dibandingkan data yang terletak di antara kelas 2 dan 3. Dengan demikian, untuk mendapatkan performa JST RBF yang baik, set data pelatihan yang digunakan harus memiliki interaksi antar kelas yang lemah seperti yang dicontohkan antara set data 2 dan 3. Akan tetapi, analisis ini sulit untuk diterapkan pada percobaan yang dilakukan penulis karena besarnya dimensi data yang digunakan. Pada 2 dimensi, penentuan apakah suatu nilai terdapat di antara 2 nilai dapat dilakukan dengan sangat muda yakni dengan membandingkannya. Sedangkan pada dimensi banyak, penentuan apakah suatu data terletak antara 2 set data sangat sulit dilakukan karena tidak dapat dilakukannya perbandingan seperti pada bidang 2 dimensi ataupun penggambaran data.

Faktor lain yang juga mempengaruhi tingkat pengenalan set data pelatihan dan juga telah dibahas pada 3.5.2 adalah pemilihan fungsi *error*. Seperti yang telah dijelaskan, bahwa fungsi *error* memainkan peranan yang sangat penting dalam proses pelatihan dan pemakaian fungsi *error* yang tepat akan menghasilkan performa pengenalan data pelatihan yang maksimal. Dengan demikian, apabila data yang digunakan untuk pelatihan cukup mewakili data yang nanti diujikan, maka tingkat pengenalan data pengujian akan memiliki hasil yang sejalan dengan tingkat pengenalan data pelatihan. Hal ini terbukti benar dari hasil percobaan yang dapat dilihat dari hasil perbandingan gambar 3.3 dan 3.4 yang menunjukkan set data *Hearth diseases* lebih cocok didekati dengan fungsi *error Cross-entropy*.

Untuk tingkat pengenalan set data pelatihan, faktor utama yang paling berpengaruh adalah hasil pelatihan karena untuk menguji set data pengujian diperlukan parameter-parameter yang diperoleh dari hasil pelatihan. Namun, hal ini bukannya faktor tunggal penentu tingkat pengenalan. Faktor lainnya yang mempengaruhi tingkat pengenalan data pengujian adalah pemilihan set data pelatihan. Dalam dunia teknik kendali, suatu pengendali selalu dirancang untuk

bekerja pada suatu kondisi tertentu yang telah ditentukan. Sama halnya dengan pemilihan data pelatihan dan pengujian, untuk medapatkan hasil pengenalan data pengujian yang baik diperlukan latihan terhadap data yang memiliki variasi yang serupa dengan data pengujian. Hal ini terlihat pada hasil tingkat pengenalan data pengujian (gambar 3.4) yang tidak jauh berbeda dengan tingkat pengenalan data pelatihan (gambar 3.3). Dengan kata lain, JST RBF hanya mampu mengenali data-data pengujian yang memiliki kemiripan dengan data pelatihan untuk kedua jenis fungsi *error*.

3.5.4 Analisis Waktu Komputasi

Waktu komputasi adalah waktu yang diperlukan untuk melakukan proses komputasi. Pada percobaan JST tunggal, terdapat 2 buah waktu komputasi yang diperhitungkan yaitu waktu komputasi pelatihan dan waktu komputasi pengujian. Untuk dapat memahami lebih baik, analisis waktu komputasi akan dibagi kedalam 2 bagian yaitu analisis waktu komputasi pelatihan dan analisis waktu komputasi pengujian.

Dari gambar 3.7, dapat dilihat waktu komputasi yang dibutuhkan untuk melatih data pelatihan bervariasi antar set data yang digunakan. Pada dasarnya, lamanya waktu komputasi pelatihan ini merupakan waktu yang diperlukan untuk melakukan komputasi selama proses pelatihan.

Dari gambar 3.8, dapat dilihat waktu komputasi yang dibutuhkan untuk menguji data pengujian bervariasi antar set data yang digunakan. Pada dasarnya, metode pengujian yang dilakukan merupakan suatu kumpulan operasi aritmatika. Dengan demikian, dapat ditentukan bahwa lamanya waktu komputasi akan sangat bergantung pada banyaknya operasi aritmatika yang dilakukan. Secara garis besar, proses pelatihan pada JST RBF dapat dibagi menjadi 2 bagian, yakni proses *feedforward* dan *backpropagation* yang dilakukan terus-menerus hingga kriteria berhenti terpenuhi. Yang menjadi perbedaan utama antara waktu komputasi pelatihan dan pengujian adalah proses pengujian hanya dilakukan 1 kali dan hanya menggunakan bagian *feedforward*. Dengan demikian telah dapat dipastikan bahwa waktu komputasi pengujian akan sangat jauh lebih cepat dibandingkan waktu komputasi pelatihan.

Pada tabel 2.2, dapat dilihat bahwa setiap set data memiliki keunikan pada jumlah data, jumlah kelas, dan jumlah atribut yang berbeda-beda. Seperti yang telah dijelaskan pada butir 3.2, banyaknya data yang digunakan sebagai set data pengujian adalah 50% dari total banyaknya data. Dengan demikian, banyaknya data pengujian sebanding dengan banyaknya total data yang diperoleh. Semakin banyak data yang digunakan untuk pengujian, maka semakin banyak iterasi perhitungan yang harus dilakukan. Semakin banyak iterasi, maka semakin banyak waktu yang diperlukan untuk komputasi.

Selain jumlah data, jumlah kelas yang ada pada set data pengujian juga memiliki pengaruh terhadap lamanya waktu komputasi. Semakin banyak jumlah kelas, semakin banyak perhitungan aritmatika yang harus dilakukan. Selain itu, banyaknya neuron pada lapisan tersembunyi pada JST RBF merupakan fungsi dari banyaknya kelas pada set data yang digunakan. Hal ini dikarenakan setiap neuron pada lapisan ini digunakan untuk menghitung jarak antara satu data terhadap nilai tengah dan lebar dari setiap kelas. Dengan demikian, banyaknya jumlah kelas akan mempengaruhi banyaknya jumlah neuron pada lapisan keluaran dan tersembunyi. Semakin banyak neuron-neuron pada lapisan ini, maka semakin banyak pula perhitungan aritmatika yang dilakukan. Semakin banyak perhitungan aritmatika yang dilakukan, semakin lama pula waktu komputasi yang dibutuhkan.

Faktor lain yang mempengaruhi lamanya waktu komputasi adalah jumlah atribut data pengujian. Hal ini dapat dipastikan karena banyaknya jumlah atribut sebanding dengan banyaknya jumlah neuron pada lapisan masukan. Semakin banyak jumlah atribut, maka perhitungan pada fungsi Radial Basis semakin banyak. Sama seperti sebelumnya, semakin banyak perhitungan yang harus dilakukan, semakin lama pula waktu komputasi.

Pada proses pelatihan, terdapat proses komputasi yang banyak karena pada proses ini *feedforward* dan *backpropagation* dilakukan berulang-ulang di mana lamanya waktu komputasi kedua bagian ini sangat dipengaruhi oleh 3 faktor yaitu jumlah data yang digunakan, jumlah atribut, dan jumlah kelas pada set data. Untuk proses *backpropagation*, lamanya waktu komputasi juga disebabkan oleh pemilihan fungsi *error* (analisis pengaruh fungsi *error* terhadap waktu komputasi pelatihan dapat dilihat pada butir 3.5.2). Dari gambar 3.6, dapat dilihat bahwa

banyaknya epoh yang digunakan dalam proses pelatihan JST RBF adalah 10000. Dari analisis ini, dapat disimpulkan bahwa lamanya waktu komputasi pelatihan bergantung pada 5 faktor, yaitu banyaknya jumlah data yang digunakan, jumlah kelas pada set data, jumlah atribut, pemilihan fungsi *error*, dan banyaknya epoh pelatihan. Analisis ini terbukti benar apabila mengacu pada hasil percobaan gambar 3.7.

Seperti yang telah dijelaskan sebelumnya, proses pengujian merupakan proses *feedforward* yang dilakukan sebanyak 1 kali dan waktu komputasi proses ini sangat bergantung dari 3 faktor yang telah dijelaskan, yaitu banyaknya data pengujian yang digunakan, banyaknya atribut pada data pengujian, dan banyaknya kelas set data pengujian. Hasil analisis ini terbukti benar apabila mengacu pada hasil percobaan gambar 3.8.

3.5.5 Perbandingan Performa JST Algoritma RBF Dengan BP

Setelah mengetahui performa dari JST RBF dengan kedua fungsi *error*, langkah selanjutnya adalah membandingkan performa yang ditunjukkan JST RBF dengan performa yang ditunjukkan JST BP. Perbandingan yang akan dilakukan adalah perbandingan mengenai kemampuan pengenalan JST dan lamanya waktu komputasi untuk kedua set data pelatihan dan pengujian.

Seperti yang telah disebutkan, tingkat pengenalan terhadap suatu data dipengaruhi oleh fungsi *error* yang digunakan. Dari gambar 3.3, kedua algoritma pembelajaran menunjukkan variasi tingkat pengenalan data pelatihan. Pada fungsi *error* kuadratis, JST RBF menunjukkan performa yang lebih baik pada data Balance scale weight and distance, Credit card approval, Glass detection, dan Hearth diseases. Sedangkan pada data lainnya, JST BP menunjukkan hasil yang lebih baik. Pada fungsi *error Cross-entropy*, JST RBF menunjukkan performa yang lebih baik pada data Credit card approval, Glass detection, dan Hearth diseases. Sedangkan pada data lainnya, JST BP menunjukkan hasil yang lebih baik. Asumsi suatu data menunjukkan hasil yang lebih baik apabila terdapat perbedaan tingkat pengenalan minimal sebanyak 2%. Dari hasil ini, terlihat bahwa mayoritas data dapat dikenali lebih baik oleh JST BP.

Seperti yang telah dibahas, JST RBF bekerja berdasarkan perhitungan jarak dan mempelajari jarak tersebut untuk mengenali data. Sedangkan JST BP bekerja berdasarkan pengenalan pola yang disimpan dalam bentuk bobot-bobot antar lapisan. Dengan demikian, algoritma pelatihan yang ditawarkan oleh kedua algoritma memang berbeda. Namun demikian, JST RBF tidak memiliki kemampuan adaptasi sebaik yang dimiliki oleh JST BP. Kemampuan adaptasi ini memungkinkan JST BP untuk dapat bekerja terhadap data yang memiliki tingkat ambiguitas yang lebih tinggi. Dengan kata lain, JST BP memiliki tingkat pengenalan yang lebih baik dibandingkan dengan JST RBF apabila data yang dipakai memiliki sifat ambigu. Akan tetapi, apabila data yang digunakan dalam proses pelatihan memiliki ambiguitas yang rendah, maka JST RBF akan memiliki tingkat pengenalan yang tidak kalah baik dengan JST BP. Dengan demikian dapat, ditarik kesimpulan bahwa kualitas baik tidaknya hasil pengenalan data pelatihan bergantung pada tingkat ambiguitas data tersebut. Apabila set data tersebut memiliki tingkat ambiguitas rendah, maka JST BF mungkin memiliki performa yang tidak kalah baikknya dibandingkan dengan JST BP. Namun apabila set data pelatihan memiliki tingkat ambiguitas yang lebih tinggi, maka pelatihan JST dengan algotitma pembelajaran *backpropagation* menunjukkan performa yang lebih baik dibandingkan JST RBF.

Seperti pada telah dijelaskan pada analisis butir 3.5.3, performa tingkat pengenalan data pengujian sangat bergantung pada 2 hal yaitu pemilihan data pelatihan dan hasil pelatihan. Kedua faktor ini juga merupakan faktor utama pada tingkat pengenalan JST BP terhadap data pengujian. Karena data yang digunakan untuk pengujian adalah sama antara pengujian JST RBF dan JST BP, maka faktor yang paling menentukan kualitas pengenalan data pengujian terletak pada hasil pelatihan. Berdasarkan analisis tersebut, maka kesimpulan yang dapat diperoleh adalah apabila set data pengujian memiliki ambuitas yang rendah dan cukup diwakili oleh data pelatihan, maka performa pengenalan data pengujian oleh JST RBF dapat lebih baik dibandingkan JST BP. Namun apabila set data pengujian memiliki ambiguitas yang lebih tinggi dan tetap cukup diwakili oleh data pelatihan, maka performa pengenalan data pengujian oleh JST BP akan lebih baik dibandingkan JST RBF.

Seperti yang telah dijelaskan pada butir 2.3, algoritma yang ditawarkan oleh RBF lebih sederhana dibandingkan algoritma pelatihan *backpropagation*. Kesederhanaan algoritma ini menyebabkan waktu komputasi RBF menjadi lebih singkat dibandingkan dengan waktu komputasi BP. Hal ini terbukti benar dari gambar 3.7 dan 3.8 yang menunjukkan fakta bahwa rata-rata waktu komputasi pelatihan dan pengujian BP kurang lebih 2 kali lebih lama dibandingkan waktu komputasi pengujian RBF.

Dari seluruh analisis yang telah dilakukan, dapat ditarik kesimpulan bahwa algoritma pembelajaran RBF lebih baik apabila dibandingkan dengan algoritma pembelajaran BP karena algoritma pembelajaran RBF memiliki keuntungan dari biaya komputasi yang setengah kali lebih rendah dari biaya komputasi algoritma pembelajaran BP dan memiliki kulitas performa tingkat pengenalan yang tidak lebih buruk dari algoritma BP. Namun apabila data yang hendak dipelajari memiliki sifat ambigu yang besar ataupun interaksi antar kelas yang kuat, maka biaya komputasi algoritma pembelajaran RBF tidak sebanding dengan kualitas tingkat pengenalannya. Dalam kasus ini, algoritma pembelajaran BP menjadi pilihan yang lebih baik.

3.6 Kesimpulan Hasil Percobaan JST RBF Tunggal

Dari percobaan dan analisis yang telah dilakukan, maka dapat ditarik beberapa kesimpulan berikut:

1. JST RBF baik digunakan untuk data yang memiliki tingkat ambiguitas yang rendah.
2. Pemilihan fungsi *error* yang tepat akan meningkatkan performa JST RBF.
3. Terdapat 4 faktor yang mempengaruhi tingkat pengenalan data set pelatihan pada JST RBF, yaitu:
   a. Ambiguitas data. Semakin rendah tingkat ambiguitas anggota set data, semakin tinggi tingkat pengenalan JST RBF.
   b. Interaksi antar kelas. Semakin rendah interaksi antar kelas, semakin tinggi tingkat pengenalan JST RBF.
   c. Pemilihan fungsi *error*.

4. Faktor yang mempengaruhi kualitas tingkat pengenalan set data pengujian adalah pemilihan data pelatihan yang tepat dan hasil pelatihan yang baik.
5. Waktu komputasi pelatihan dipengaruhi oleh 5 faktor, yaitu:
   a. Jumlah data penguji.
   b. Jumlah atribut data penguji.
   c. Jumlah kelas set data penguji.
   d. Pemilihan fungsi *error*.
   e. Banyaknya epoh pelatihan.
6. Waktu komputasi pengujian dipengaruhi oleh 3 faktor, yaitu:
   a. Jumlah data penguji.
   b. Jumlah atribut data penguji.
   c. Jumlah kelas set data penguji.
7. JST RBF memiliki kemampuan pengenalan yang tidak lebih buruk apabila dibandingkan dengan JST BP apabila terkait dengan set data yang memiliki tingkat ambiguitas dan interaksi antar kelas yang rendah. Sedangkan untuk data dengan tingkat ambiguitas dan interaksi antar kelas yang lebih tinggi, tingkat pengenalan yang ditunjukkan oleh JST BP lebih baik dibandingkan dengan JST RBF.
8. JST RBF memiliki waktu komputasi pelatihan dan pengujian kurang lebih 2 kali lebih cepat dibandingkan waktu komputasi pelatihan dan pengujian JST BP.

BAB 4
ANALISIS PERFORMA JST *ENSEMBLE*

Pada percobaan kedua ini, akan dilakukan pengembangan dari JST tunggal berbasis RBF untuk meningkatkan performa dari JST. Pengembangan yang dilakukan adalah dengan menggunakan metode *ensemble* pada JST. Struktur penulisan pada bab ini sama seperti bab sebelumnya, yaitu tujuan yang hendak dicapai, prosedur percobaan yang dilakukan, algoritma program yang digunakan, hasil percobaan, analisis hasil data percobaan, dan kesimpulan yang dapat ditarik dari hasil percobaan yang dilakukan.

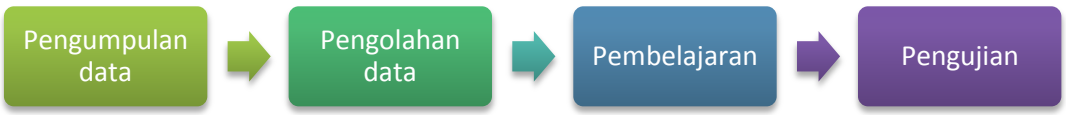
4.1 Tujuan Percobaan JST RBF *Ensemble*

Berikut ini merupakan tujuan yang hendak dicapai dari percobaan yang dilakukan pada bab ini:

1. Mengembangkan algoritma JST *ensemble* berbasis RBF.
2. Memahami pengaruh dari metode *ensemble* terhadap performa JST berbasis RBF.
3. Memahami pengaruh banyaknya jumlah *ensemble* yang digunakan terhadap performa dari JST berbasis RBF.

4.2 Prosedur Percobaan JST RBF *Ensemble*

Pada percobaan yang akan dilakukan, terdapat langkah-langkah yang dikembangkan guna memperoleh tahapan percobaan yang terstruktur. Berikut langkah-langkah percobaan yang digunakan:



Gambar 4.1. Prosedur Percobaan JST *Ensemble*

Seperti yang terlihat pada gambar 4.1, terdapat 4 tahap yang dilakukan pada percobaan ini. Berikut penjelasan untuk masing-masing tahap percobaan:

1. Pengumpulan data

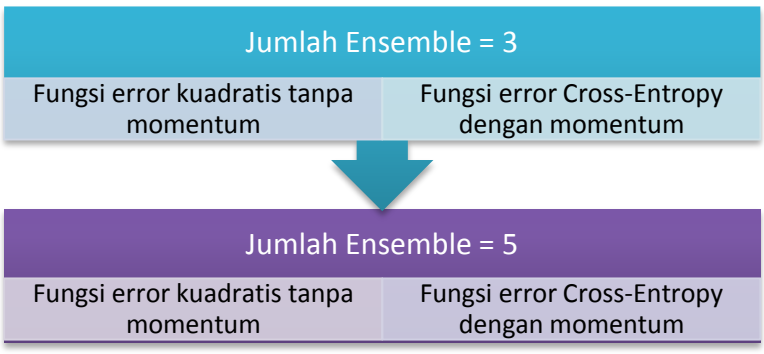
   Seperti yang telah dijelaskan pada butir 3.2, tahap ini merupakan tahap di mana data-data yang akan digunakan pada percobaan dikumpulkan dan ditata ulang untuk mempermudah pemakaian. Data yang digunakan pada percobaan ini sama dengan data yang digunakan pada percobaan pertama, yaitu berjumlah 11 set data di mana 9 set data berasal dari UCI, satu set data citra wajah cahaya tampak yang diperoleh dari Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng. selaku pembimbing penulis, dan satu set data citra wajah infra merah yang diperoleh dari laporan skripsi yang disusun oleh Stephen Roy Imantaka, ST. dengan judul Sistem Pengenal Wajah Berbasis *Ensemble Neural Network* untuk Citra Infra Merah (2010). Untuk memahami tahap ini, pembaca dapat melihat kembali penjelasan pada butir 3.2.

2. Pengolahan data

   Pengolahan data merupakan suatu tahap di mana data-data yang telah diperoleh mengalami pengolahan awal berupa normalisasi atau PCA seperti yang dilakukan pada percobaan pertama. Setelah mengalami normalisasi atau PCA, data kemudian dibagi menjadi 2 bagian yakni data yang digunakan untuk pelatihan dan data yang digunakan untuk pengujian. Perbandingan antara data pelatihan dan data pengujian yang digunakan pada percobaan ini adalah sama dengan perbandingan yang digunakan pada percobaan pertama yakni 50:50.

3. Pelatihan

   Pelatihan merupakan tahap di mana JST dilatih untuk mengenali data-data yang telah mengalami normalisasi atau PCA. Pada percobaan ini, terdapat 4 metode pelatihan yang dapat dilihat pada gambar 4.2 berikut:



Gambar 4.2. Metodologi Percobaan JST *Ensemble*

Seperti yang telah disebutkan pada tujuan percobaan, percobaan kedua ini dilakukan untuk memahami pengaruh metode *ensemble* terhadap performa dari JST berbasis RBF. Dengan demikian, pada percobaan ini JST *ensemble* dikembangkan dan dipelajari. Sesuai dengan gambar 4.2, terdapat 4 percobaan yang dilakukan yakni pelatihan JST *ensemble* dengan pilihan jumlah *ensemble* yang digunakan adalah sebanyak 3 dan 5 serta menggunakan 2 jenis fungsi *error* untuk masing-masing jumlah *ensemble*.

Seperti yang dilakukan pada percobaan pertama, untuk masing-masing percobaan akan dijalankan sebanyak 5 kali. Tujuan dari melakukan percobaan lebih dari 1 kali adalah untuk mengetahui kosistensi dari performa JST dan memperoleh *sample* untuk dilakukan analisis.

Hasil yang diperoleh dari tahap pelatihan adalah besarnya vektor nilai tengah RBF, lebar RBF serta bobot-bobot dan bias-bias JST yang akan digunakan pada tahap pengujian. Selain itu, *error* terkecil, banyaknya epoh pelatihan, serta waktu komputasi pelatihan juga diperoleh dari tahap ini.

4. Pengujian

Pengujian adalah tahap di mana nilai-nilai keluaran dari hasil pelatihan JST terhadap data pelatihan, diterapkan kembali terhadap set data pengujian untuk setiap percobaan. Hasil yang diperoleh dari tahap pengujian adalah performa dari JST yang meliputi tingkat pengenalan terhadap data pengujian serta waktu komputasi dari proses pengujian.

4.3 Algoritma Program

Berikut keempat algoritma yang digunakan untuk keempat percobaan JST *Ensamble*:

4.3.1 Algoritma JST RBF *Ensemble* Fungsi *Error* Kuadratis

Berikut algoritma yang dikembangkan dalam percobaan JST *Ensamble* dengan algoritma RBF untuk fungsi *error* kuadratis:

A. Inisialisasi

a) Inisialisasi bobot dengan menggunakan Nguyen-Widrow

1) Bias antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{i,J0} = \begin{bmatrix} w_{10} & w_{20} & \cdots & w_{L0} \end{bmatrix}$$

2) Bobot antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{i,lm} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L1} & w_{L2} & \cdots & w_{LM} \end{bmatrix}$$

b) Vektor masukan

$$\boldsymbol{x}_{pn} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{P1} & x_{P2} & \cdots & x_{PN} \end{bmatrix}$$

c) Vektor target

$$\boldsymbol{T}_{pl} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1L} \\ t_{21} & t_{22} & \cdots & t_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ t_{P1} & t_{P2} & \cdots & t_{PL} \end{bmatrix}$$

d) Vektor nilai tengah RBF yang diambil dari rata-rata tiap kelas

$$\boldsymbol{c}_{i,mn} = \begin{bmatrix} c_{i,11} & c_{i,12} & \cdots & c_{i,1N} \\ c_{i,21} & c_{i,22} & \cdots & c_{i,2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{i,M1} & c_{i,M2} & \cdots & c_{i,MN} \end{bmatrix}$$

e) Lebar RBF dengan menggunakan persamaan berikut:

1) Untuk $m = 1 - M$, lakukan langkah berikut:

1. $\sigma_{i,m} = \dfrac{d_{\max}}{\sqrt{k}}$

B. Untuk epoh < *epoh_max* dan err_total > *err_max*, lakukan langkah berikut:

1. Untuk $i = 1 - I$, lakukan langkah berikut

a) Untuk $p = 1 - P$, lakukan langkah berikut:

1) Perhitungan lapisan tersembunyi

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) $zin_{i,m} = \sum_{n=1}^{N} \dfrac{\left(x_{pn} - c_{i,mn}\right)^2}{2\sigma_{i,m}{}^2}$

ii) $z_{i,m} = \exp\left(-zin_{i,m}\right)$

2) Perhitungan pada lapisan keluaran

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $yin_{i,l} = w_{i,l0} + \sum_{m=1}^{M} z_{i,m} w_{i,lm}$

ii) $y_{i,l} = \frac{1}{1 + \exp(-yin_{i,l})}$

2. Pengambilan keputusan

$F_l = \frac{1}{I} \sum_{i=1}^{I} y_{i,l}$

3. Perhitungan *error*

a) Untuk $l = 1 - L$, lakukan langkah berikut:

1) $e_l = F_l - T_{pl}$

b) $E_p = \sum_{l=1}^{L} e_l^{\,2}$

4. Untuk $i = 1 - I$, lakukan langkah berikut:

a) Pembelajaran bobot dan bias antara lapisan keluaran dengan tersembunyi

1) Untuk $l = 1 - L$, lakukan langkah berikut:

a) $Sy_{i,l} = y_{i,l}(1 - y_{i,l})$

b) $dk_{i,l} = \left[(1-\lambda)(T_{pl} - y_{i,l}) + \lambda(T_{pl} - F_l)\right] Sy_{i,l}$

c) Untuk $m = 1 - M$, lakukan langkah berikut:

i) $dw_{i,lm} = \alpha \times dk_{i,l} \times z_{i,m}$

d) $dw_{i,l0} = \alpha \times dk_{i,l}$

b) Pembelajaran vektor nilai tengah dan lebar RBF

1) Untuk $m = 1 - M$, lakukan langkah berikut:

a) Untuk $n = 1 - N$, lakukan langkah berikut:

(1) $dc_{i,mn} = \alpha_c \frac{\sum_{l=1}^{L} dk_{i,l} w_{i,lm}}{\sigma_{i,m}^{\,2}} z_{i,m} (x_{pn} - c_{i,mn})$

b) $d\sigma_{i,m} = \alpha_c \frac{\sum_{l=1}^{L} dk_{i,l} w_{i,lm}}{\sigma_{i,m}^{\,3}} z_{i,m} \sum_{n=1}^{N} (x_{pn} - c_{i,mn})^2$

c) Perubahan parameter

1) $\boldsymbol{w}_{i,lm} = \boldsymbol{w}_{i,lm} + d\boldsymbol{w}_{i,lm}$

2) $\boldsymbol{w}_{i,l0} = \boldsymbol{w}_{i,l0} + d\boldsymbol{w}_{i,l0}$

3) $\boldsymbol{c}_{i,mn} = \boldsymbol{c}_{i,mn} + d\boldsymbol{c}_{i,mn}$

4) $\boldsymbol{\sigma}_{i,m} = \boldsymbol{\sigma}_{i,m} + d\boldsymbol{\sigma}_{i,m}$

C. Kondisi henti

$$\sum_{p=1}^{P} E_p \leq 0.01$$

4.3.2 Algoritma JST RBF *Ensamble* Fungsi *Error Cross-entropy*

Berikut algoritma yang dikembangkan dalam percobaan JST *Ensamble* dengan algoritma RBF untuk fungsi *error Cross-entropy* dengan momentum:

A. Inisialisasi

a) Inisialisasi bobot dengan menggunakan Nguyen-Widrow

1) Bias antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{i,l0} = \begin{bmatrix} w_{10} & w_{20} & \cdots & w_{L0} \end{bmatrix}$$

2) Bobot antara lapisan tersembunyi dengan lapisan keluaran

$$\boldsymbol{w}_{i,lm} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L1} & w_{L2} & \cdots & w_{LM} \end{bmatrix}$$

b) Vektor masukan

$$\boldsymbol{x}_{pn} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{P1} & x_{P2} & \cdots & x_{PN} \end{bmatrix}$$

c) Vektor target

$$\boldsymbol{T}_{pl} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1L} \\ t_{21} & t_{22} & \cdots & t_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ t_{P1} & t_{P2} & \cdots & t_{PL} \end{bmatrix}$$

d) Vektor nilai tengah RBF yang diambil dari rata-rata tiap kelas

$$\boldsymbol{c}_{i,mn} = \begin{bmatrix} c_{i,11} & c_{i,12} & \cdots & c_{i,1N} \\ c_{i,21} & c_{i,22} & \cdots & c_{i,2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{i,M1} & c_{i,M2} & \cdots & c_{i,MN} \end{bmatrix}$$

e) Lebar RBF dengan menggunakan persamaan berikut:

1) Untuk $m = 1 - M$, lakukan langkah berikut:

1. $\sigma_{i,m} = \dfrac{d_{\max}}{\sqrt{k}}$

B. Untuk epoh < *epoh_max* dan err_total > *err_max*, lakukan langkah berikut:

1. Untuk $i = 1 - I$, lakukan langkah berikut

a) Untuk $p = 1 - P$, lakukan langkah berikut:

1) Perhitungan lapisan tersembunyi

a) Untuk $m = 1 - M$, lakukan langkah berikut:

i) $zin_{i,m} = \sum_{n=1}^{N} \dfrac{\left(x_{pn} - c_{i,mn}\right)^2}{2\sigma_{i,m}^{\ 2}}$

ii) $z_{i,m} = \exp\left(-zin_{i,m}\right)$

2) Perhitungan pada lapisan keluaran

a) Untuk $l = 1 - L$, lakukan langkah berikut:

i) $yin_{i,l} = w_{i,l0} + \sum_{m=1}^{M} z_{i,m} w_{i,lm}$

ii) $y_{i,l} = \dfrac{1}{1 + \exp\left(-yin_{i,l}\right)}$

2. Pengambilan keputusan

$$F_l = \frac{1}{I} \sum_{i=1}^{I} y_{i,l}$$

3. Perhitungan *error*

a) Untuk $l = 1 - L$, lakukan langkah berikut:

1) $e_l = -T_{pl} \ln\left(F_l\right) - \left(1 - T_{pl}\right) \ln\left(1 - F_l\right)$

b) $E_p = \sum_{l=1}^{L} e_l^{\ 2}$

4. Untuk $i = 1 - I$, lakukan langkah berikut:

a) Pembelajaran bobot dan bias antara lapisan keluaran dengan tersembunyi

1) Untuk $l = 1 - L$, lakukan langkah berikut:

a) $Sy_{i,l} = y_{i,l}\left(1 - y_{i,l}\right)$

b) $dk_{i,l} = \left(T_{pl} - y_{i,l}\right) + \lambda\left(F_l - y_l\right)Sy_{i,l}$

c) Untuk $m = 1 - M$, lakukan langkah berikut:

j) $dw_{i,lm}(k) = \alpha \times dk_{i,l} \times z_{i,m} + \mu \times dw_{i,lm}(k-1)$

d) $dw_{i,l0}(k) = \alpha \times dk_{i,l} + \mu \times dw_{i,l0}(k-1)$

b) Pembelajaran vektor nilai tengah dan lebar RBF

1) Untuk $m = 1 - M$, lakukan langkah berikut:

a) Untuk $n = 1 - N$, lakukan langkah berikut:

(1) $dc_{i,mn}(k) = \alpha_c \dfrac{\sum_{l=1}^{L} dk_{i,l} w_{i,lm}}{\sigma_{i,m}^2} z_{i,m}\left(x_{pn} - c_{i,mn}\right) + \mu \times dc_{i,mn}(k-1)$

b) $d\sigma_{i,m}(k) = \alpha_c \dfrac{\sum_{l=1}^{L} dk_{i,l} w_{i,lm}}{\sigma_{i,m}^3} z_{i,m} \sum_{n=1}^{N}\left(x_{pn} - c_{i,mn}\right)^2 + \mu \times d\sigma_{i,m}(k-1)$

c) Perubahan parameter

1) $\boldsymbol{w}_{i,lm} = \boldsymbol{w}_{i,lm} + d\boldsymbol{w}_{i,lm}$

2) $\boldsymbol{w}_{i,l0} = \boldsymbol{w}_{i,l0} + d\boldsymbol{w}_{i,l0}$

3) $\boldsymbol{c}_{i,mn} = \boldsymbol{c}_{i,mn} + d\boldsymbol{c}_{i,mn}$

4) $\boldsymbol{\sigma}_{i,m} = \boldsymbol{\sigma}_{i,m} + d\boldsymbol{\sigma}_{i,m}$

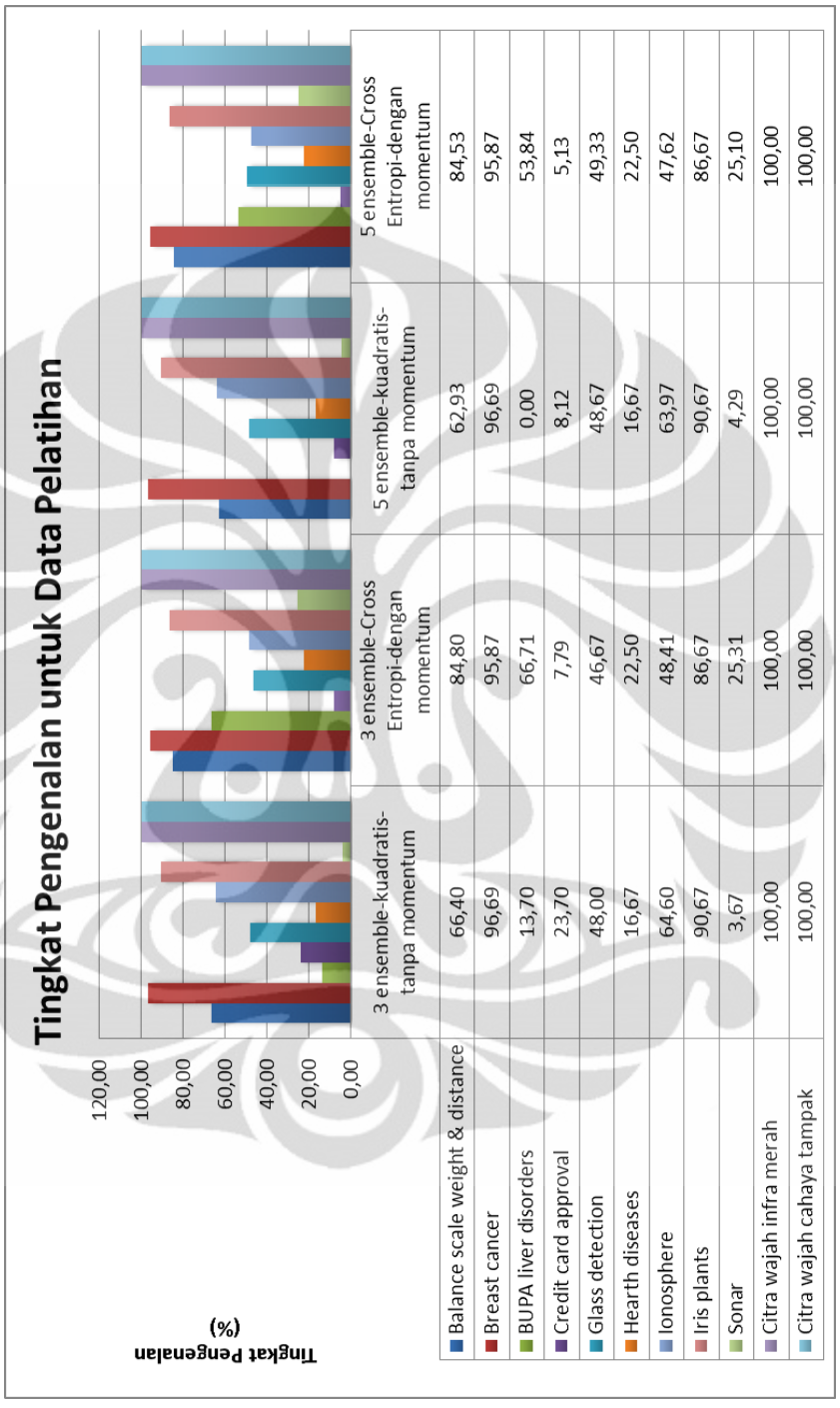
C. Kondisi henti

$$\sum_{p=1}^{P} E_p \le 0.01$$

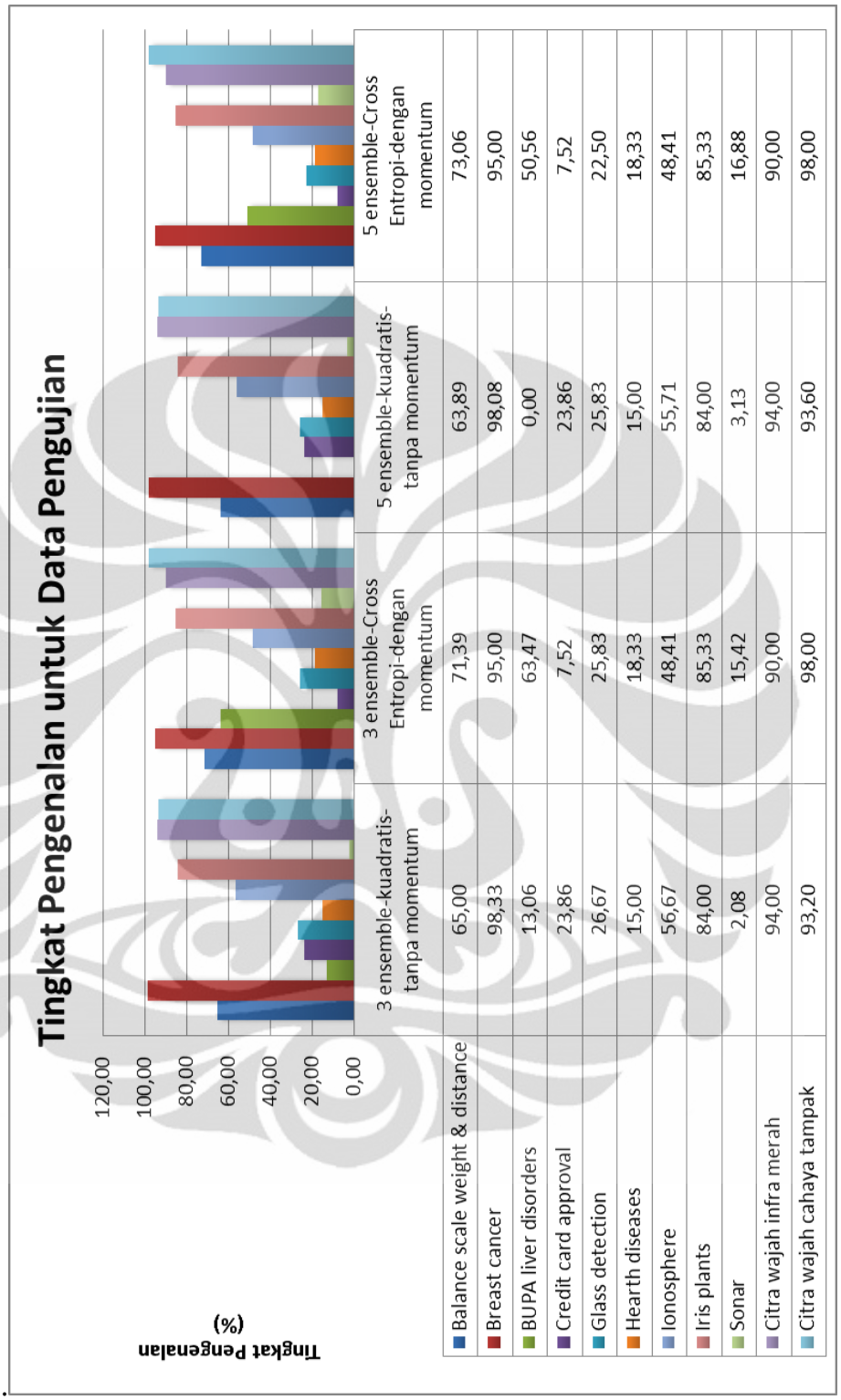
4.4 Hasil Percobaan JST RBF *Ensemble*

Pada bagian ini akan ditampilkan hasil percobaan JST RBF *ensemble* untuk jumlah *ensemble* 3 dan 5 dengan menggunakan kedua fungsi *error*. Hasil

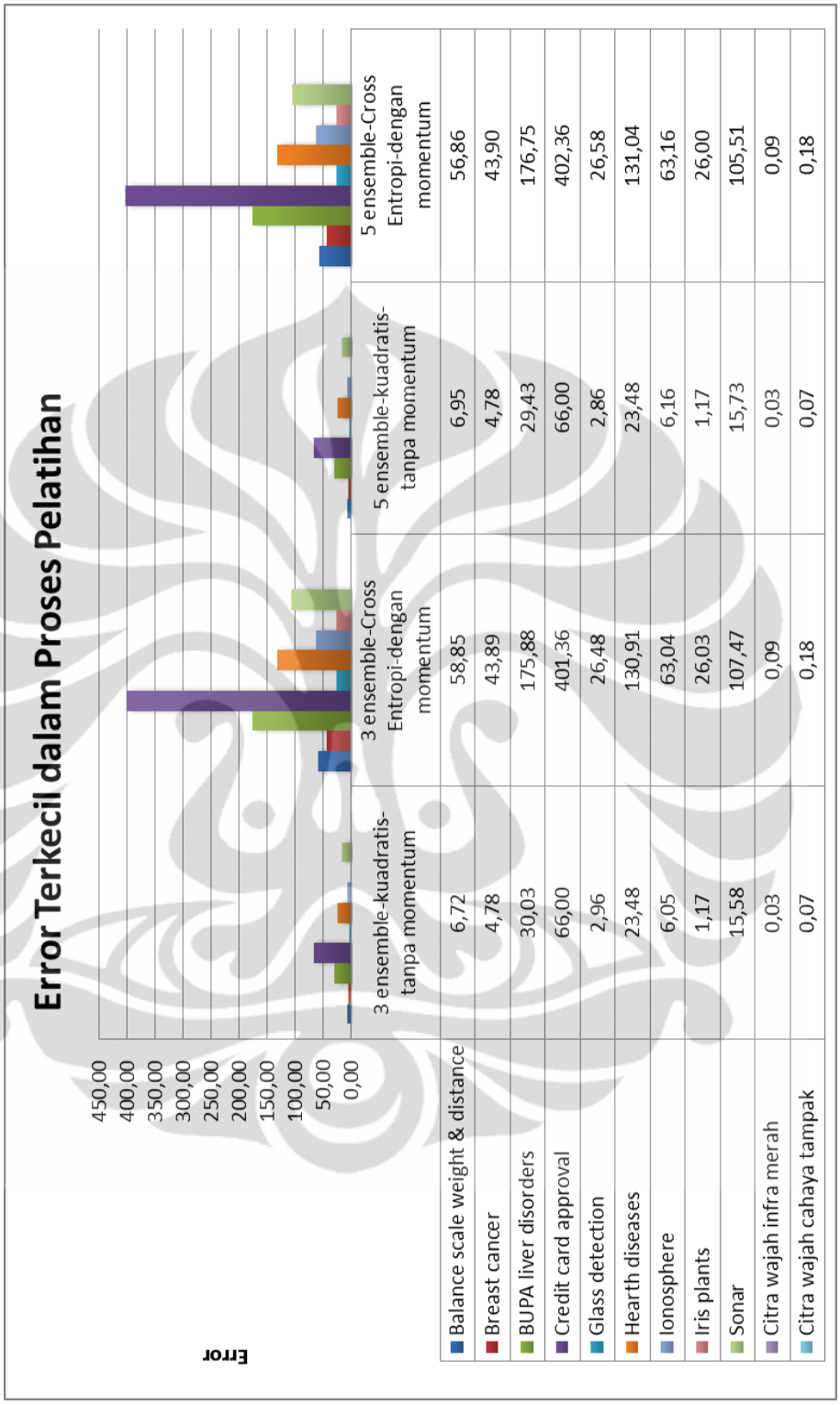
percobaan yang ditampilkan merupakan rata-rata dari 5 kali percobaan untuk masing-masing metode yang terdapat pada gambar 4.3 hingga 4.8.



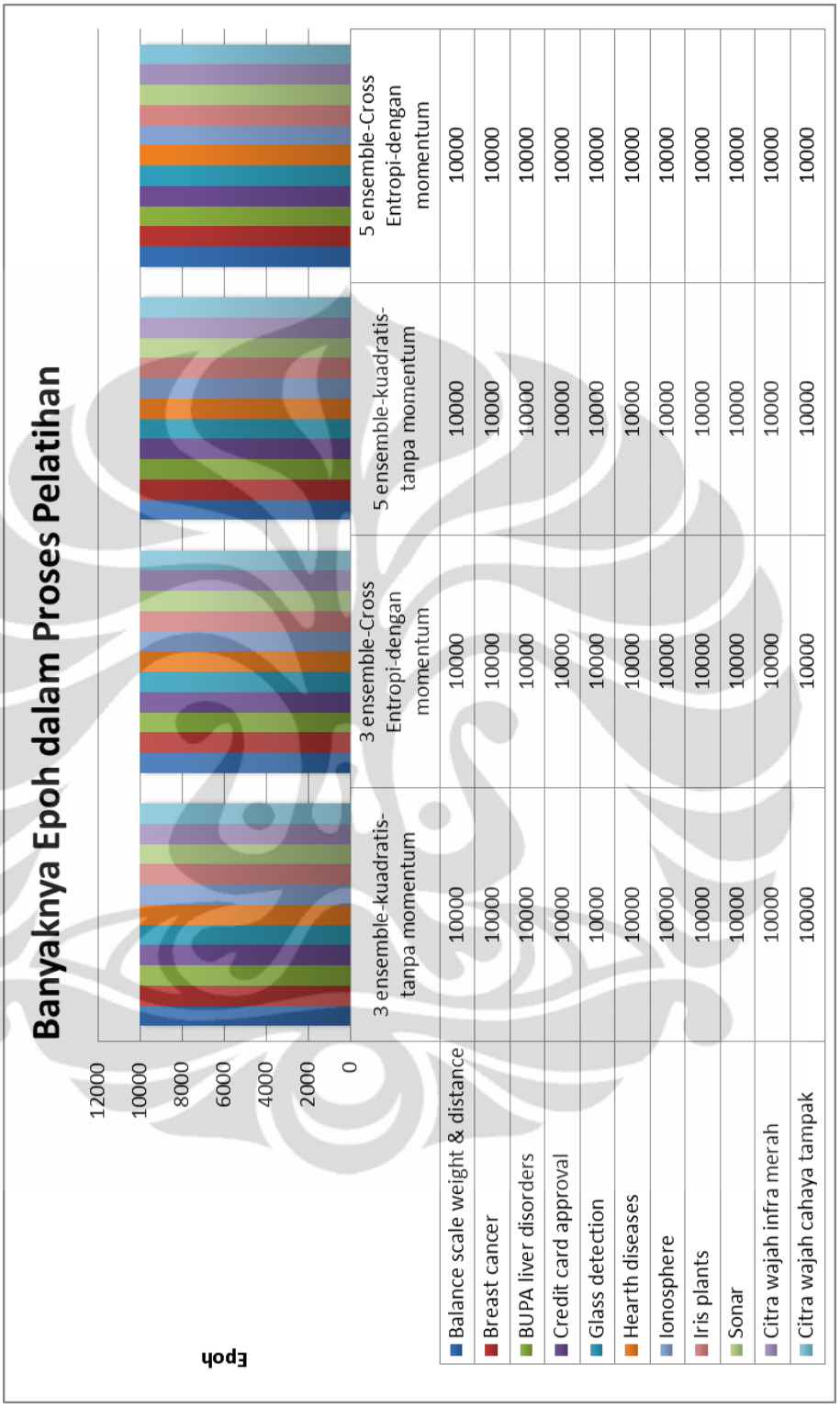
Gambar 4.3. Tingkat Pengenalan untuk Data Pelatihan pada JST *Ensemble*



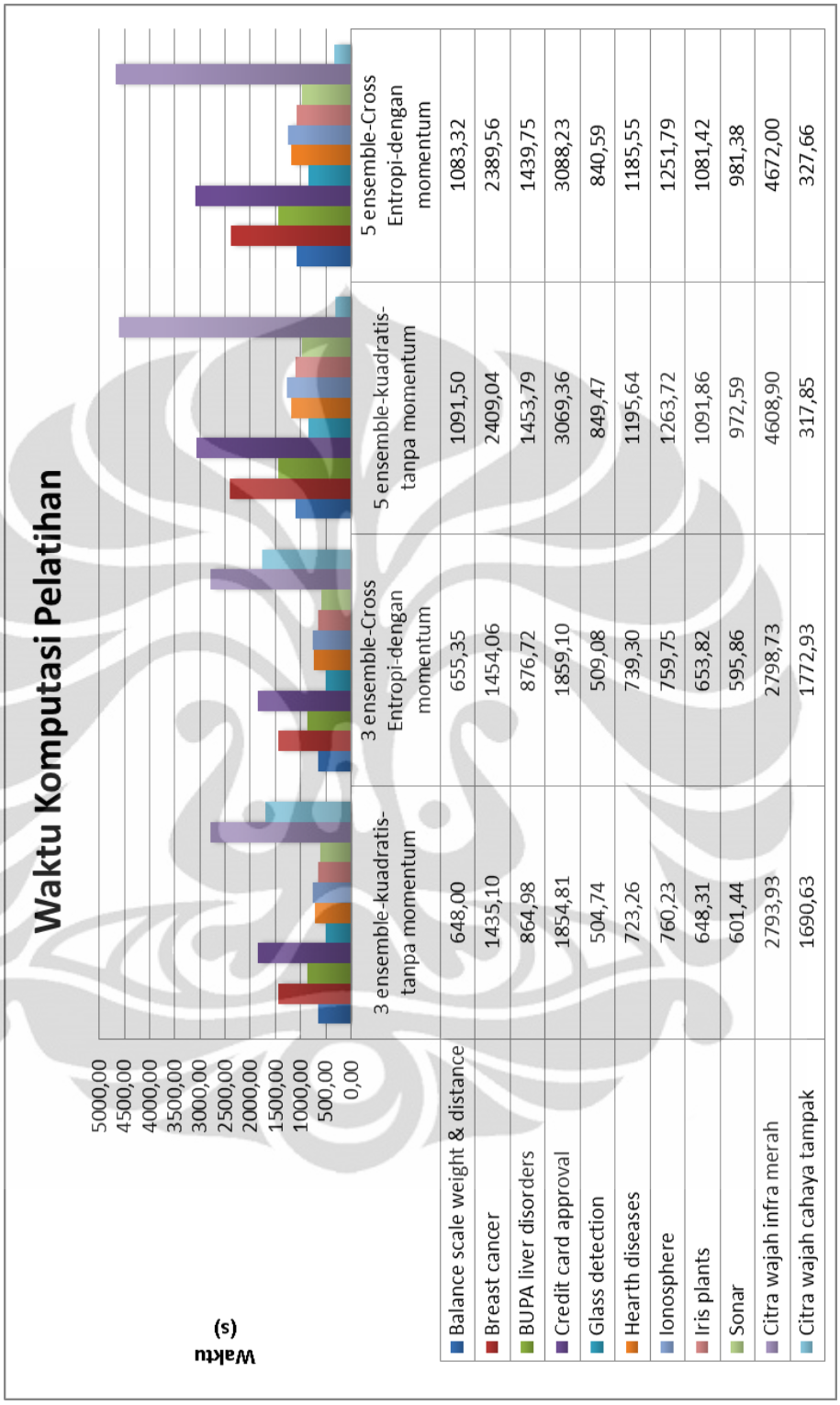
Gambar 4.4. Tingkat Pengenalan Data Pengujian pada JST *Ensemble*



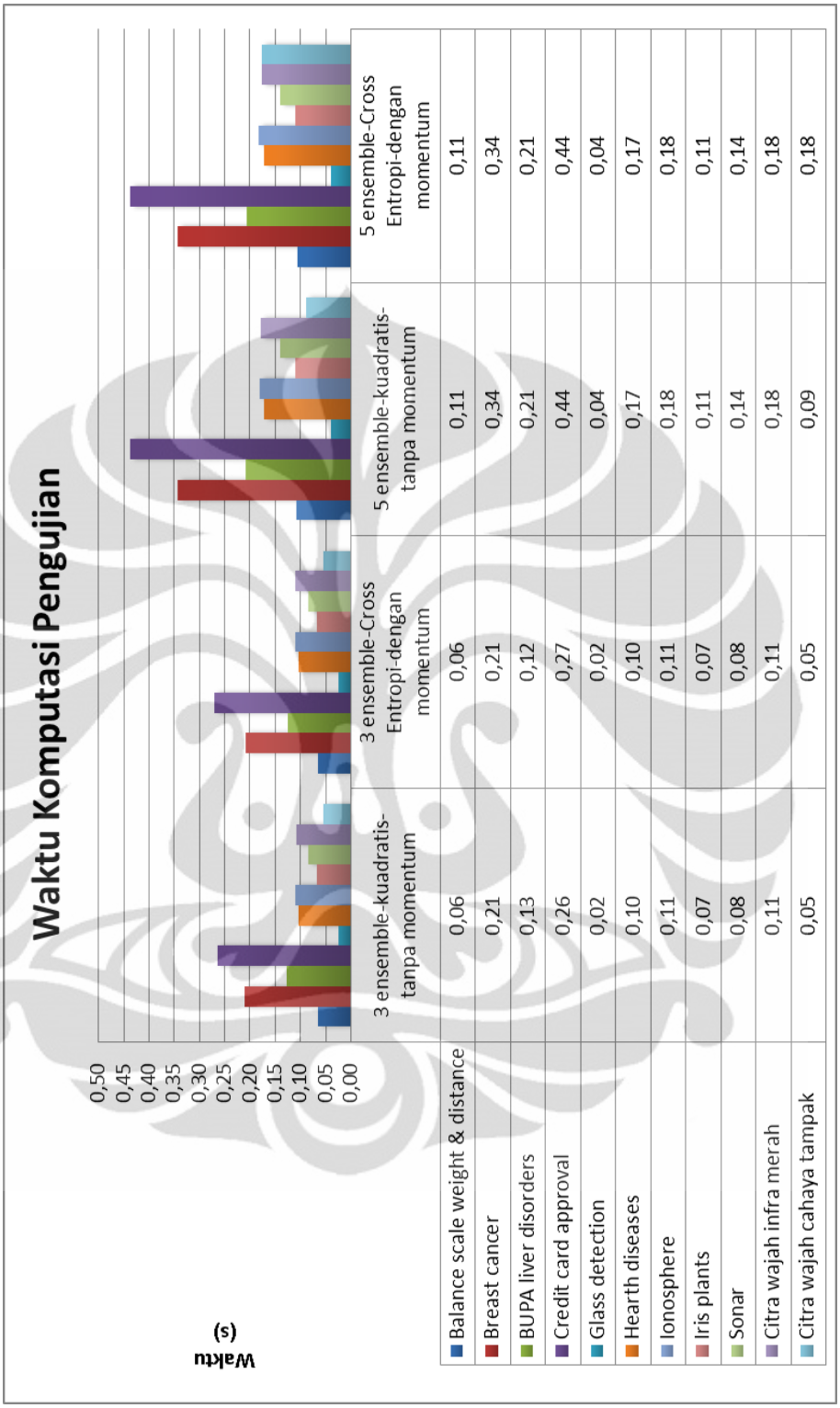
Gambar 4.5. Error Terkecil dalam Proses Pelatihan JST *Ensemble*



Gambar 4.6. Banyaknya Epoh dalam Proses Pelatihan JST *Ensemble*



Gambar 4.7. Waktu Komputasi Pelatihan JST *Ensemble*



Gambar 4.8. Waktu Komputasi Pengujian JST *Ensemble*

4.5 Analisis Performa JST RBF *Ensemble*

Setelah mendapatkan hasil percobaan, akan dilakukan analisis terhadap hasil percobaan untuk mendapatkan pengetahuan mengenai karakter dari JST RBF *ensemble*. Untuk mendapatkan pengetahuan tersebut, akan dilakukan analisis mengenai pengaruh dari jumlah *ensamble* dalam proses pelatihan terhadap tingkat pengenalan dan waktu komputasi. Selain itu juga akan dilakukan analisis uji logika dan pengaruh dari sifat ambiguitas data terhadap performa algoritma NCL sebagai alat bantu analisis.

4.5.1 Analisis Hasil Uji Logika

Seperti yang telah dilakukan pada percobaan JST RBF tunggal, uji logika digunakan untuk mengetahui kemampuan suatu JST dalam membedakan data yang ambigu. Pada uji logika terhadap JST tunggal, telah diperoleh hasil bahwa algoritma pelatihan RBF tidak mampu untuk mengenali data-data yang ambigu. Pada percobaan JST *ensemble*, uji logika digunakan kembali untuk mengetahui pengaruh dari jumlah jaringan terhadap kemampuannya dalam memperbaiki kekurangan jaringan tunggal, yaitu untuk mengenali data-data ambigu.

Tabel 4.1. Hasil Uji Logika JST RBF *Ensemble*

N	3 *ensemble*				5 *ensemble*			
	Kuadratis		***Cross-entropy***		**Kuadratis**		***Cross-entropy***	
	y_1	y_2	y_1	y_2	y_1	y_2	y_1	y_2
1	0.8825 ■	0.1176	0.7453	0.2547	0.8391 ■	0.1610	0.8883 ■	0.1117
2	0.8825 ■	0.1176	0.7455	0.2545	0.8391 ■	0.1611	0.8883 ■	0.1116
3	0.8930 ■	0.1073	0.7452	0.2548	0.8441 ■	0.1559	0.8949 ■	0.1051
4	0.8931 ■	0.1072	0.7451	0.2549	0.8842 ■	0.1559	0.8949 ■	0.1051
5	0.5301	0.4694	0.6306	0.3694	0.5710	0.4291	0.5651	0.4349
6	0.5301	0.4694	0.6306	0.3694	0.5710	0.4291	0.5651	0.4349
7	0.5304	0.4692	0.6358	0.3642	0.5681	0.4320	0.5697	0.4304
8	0.5304	0.4692	0.6358	0.3642	0.5681	0.4320	0.5697	0.4304
9	0.5304	0.4690	0.6411	0.3589	0.5716	0.4286	0.5670	0.4330
10	0.5304	0.4690	0.6411	0.3589	0.5716	0.4286	0.5670	0.4330
11	0.5307	0.4687	0.6529	0.3471	0.5687	0.4314	0.5700	0.4300
12	0.5307	0.4687	0.6529	0.3471	0.5687	0.4314	0.5700	0.4300

Catatan: ■ menandakan data yang dikenali

Pada percobaan yang dilakukan, data yang sama yang digunakan pada uji logika JST RBF tunggal diterapkan terhadap JST RBF *ensemble*. Data ini dapat

dilihat pada tabel 3.1. Dengan menggunakan set data tersebut sebagai masukan, maka diperoleh hasil keputusan bersama yang dapat dilihat pada tabel 4.1.

Dari hasil uji logika, dapat dilihat bahwa JST RBF *ensemble* masih memiliki sifat yang sama dengan JST RBF tunggal yaitu tidak mampu mengenali data-data ambigu dan dapat mengenali data-data yang tidak ambigu dengan baik. Hal ini dapat dilihat pada tabel 4.1, di mana untuk data nomor 1 – 4 dapat dikenali dengan baik, sedangkan data nomor 5 dan 6, 7 dan 8, 9 dan 10, serta 11 dan 12 dikenali sebagai data yang sama untuk semua jumlah *ensemble* dan fungsi *error*. Penyebab dari fenomena ini sama dengan fenomena yang terjadi pada uji logika JST RBF tunggal, yaitu lapisan Radial Basis menghasilkan angka yang sama untuk data nomor 5 dan 6, 7 dan 8, 9 dan 10, serta 11 dan 12 sedangkan lapisan *backpropagation* harus mengenali keluaran dari lapisan sebelumnya sebagai data yang berbeda. Hal demikian yang membuat JST RBF *ensemble* tidak mampu mengenali data-data yang memiliki tingkat ambiguitas yang tinggi.

Apabila dilakukan perbandingan antara tabel 4.1 dengan tabel 3.2**,** maka dapat dilihat bahwa terjadi penurunan kualitas tingkat pengenalan data nomor 1 – 4. Mengingat perbedaan antar kedua tabel terletak pada algoritma pelatihan yang berbeda, maka dapat dipastikan bahwa terdapat pengaruh dari algoritma pelatihan JST RBF *ensemble* terhadap kualitas tingkat pengenalan. Perbedaan algoritma pelatihan antar kedua tabel adalah jumlah jaringan yang digunakan dan penerapan algoritma NCL yang akan dibahas setelah butir ini.

4.5.2 Analisis Pengaruh Ambiguitas

Seperti yang telah dibahas, NCL merupakan metode yang digunakan untuk menimbulkan diversitas pada setiap jaringan dengan cara menambahkan penalti. Tujuan dilakukannya algortima NCl adalah agar setiap jaringan memiliki pengetahuan yang berbeda-beda terhadap data pelatihan. Dengan algoritma demikian, diharapkan sistem dapat menghasilkan performa yang lebih tinggi apabila semua jaringan digunakan untuk mengambil keputusan.

Besarnya pengaruh penalti pada algoritma NCL ditentukan oleh parameter konstanta penalti (γ). Semakin besar nilai konstanta ini, maka semakin besar pula pengaruh dari penalti ini. Pada percobaan ini, besarnya konstanta penalti yang

digunakan adalah sebesar 0.5 karena nilai ini menghasilkan respon yang paling baik untuk data UCI (Dam, Abbas, & Yao, 2008). Untuk data citra wajah manusia, besarnya nilai konstanta penalti juga sebesar 0.5 karena nilai ini menunjukkan performa yang terbaik. Berdasarkan alasan tersebut, penurunan kualitas hasil tingkat pengenalan JST RBF *ensemble* bukan dikarenakan besarnya nilai konstanta penalti.

Pada penggunaanya, algoritma NCL merupakan algoritma yang diterapkan pada proses pelatihan. Untuk memahami pengaruh dari NCL terhadap data ambigu, akan ditampilkan hasil pelatihan JST RBF terhadap data ambigu. Tabel 4.2 dan 4.3 merupakan tabel yang berisikan beberapa hasil pelatihan JST RBF tunggal dan *ensemble* tehadap set data *Iris plants* ambigu untuk kedua fungsi *error*. (Data ambigu diperoleh dari hasil SOM yang terdapat pada lampiran F).

Tabel 4.2. Hasil Pelatihan JST Tunggal untuk Data *Iris Plants* Ambigu

No. data	JST tunggal kuadratis			JST tunggal *Cross-entropy*			Target Pelatihan		
	y_1	y_2	y_3	y_1	y_2	y_3	t_1	t_2	t_3
6	0.00	0.00	1.00 ■	0.00	0.02	0.99 ■	0.00	0.00	1.00
21	0.00	0.09	0.84	0.00	0.06	0.94 ■	0.00	0.00	1.00
42	0.00	0.00	1.00 ■	0.00	0.01	0.99 ■	0.00	0.00	1.00
45	0.00	0.00	1.00 ■	0.00	0.00	1.00 ■	0.00	0.00	1.00
60	0.00	0.03	0.97 ■	0.00	0.05	0.95 ■	0.00	0.00	1.00
66	0.00	0.00	1.00 ■	0.00	0.02	0.98 ■	0.00	0.00	1.00
72	0.00	0.04	0.96 ■	0.00	0.10	0.90 ■	0.00	0.00	1.00

Catatan: ■ menandakan data yang dikenali

Tabel 4.3. Hasil Pelatihan JST *Ensemble* untuk Data *Iris Plants* Ambigu

No. data	JST *ensemble* kuadratis			JST *ensemble Cross-entropy*			Target Pelatihan		
	y_1	y_2	y_3	y_1	y_2	y_3	t_1	t_2	t_3
6	0.00	0.10	0.93 ■	0.00	0.11	0.91 ■	0.00	0.00	1.00
21	0.00	0.20	0.65	0.00	0.13	0.41	0.00	0.00	1.00
42	0.00	0.05	0.96 ■	0.00	0.09	0.92 ■	0.00	0.00	1.00
45	0.00	0.01	0.99 ■	0.00	0.02	0.98 ■	0.00	0.00	1.00
60	0.00	0.31	0.77	0.00	0.45	0.73	0.00	0.00	1.00
66	0.00	0.11	0.90 ■	0.00	0.09	0.84	0.00	0.00	1.00
72	0.00	0.41	0.66	0.00	0.03	0.68	0.00	0.00	1.00

Catatan: ■ menandakan data yang dikenali

Dengan membandingkan hasil pelatihan pada tabel 4.2 dan 4.3, dapat dilihat bahwa algoritma pelatihan NCL memperkuat efek ambiguitas data sehingga semakin sulit untuk dikenali. Hal ini terbukti dari semakin sedikitnya jumlah data ambigu yang dapat dikenali oleh JST. Pada jaringan tunggal dengan fungsi *error* kuadratis, 6 dari 7 data ambigu dapat dikenali dan semua data dapat dikenali dengan fungsi *error Cross-entropy*. Sedangkan pada JST *ensemble* dengan fungsi *error* kuadratis, hanya 4 dari 7 data ambigu yang dapat dikenali dan 3 dari 7 data dapat dikenali dengan fungsi *error Cross-entropy*.

Fenomena pada tabel 4.2 dan 4.3 dapat terjadi karena ketika jaringan *ensemble* memperlajari data ambigu, setiap jaringan memperoleh *error* empiris yang besar dan ditambah dengan efek penalti yang besar juga (efek penalti yang besar diperoleh dari kesalahan jaringan lainnya yang besar). Hal demikian akan membuat bobot, nilai tengah RBF, dan lebar RBF mengalami perubahan yang besar untuk beradaptasi dengan data tersebut. Ketika pelatihan dilanjutkan dengan menggunakan data non-ambigu, besarnya nilai *error* empiris dan penalti akan menjadi membesar kembali sehingga akan memperbaiki bobot, nilai tengah RBF, dan lebar RBF untuk beradaptasi data non-ambigu. Karena mayoritas data adalah data non-ambigu, maka pelatihan akan cenderung konvergen untuk mempelajari data majoritas yaitu data non-ambigu dan data ambigu tetap tidak dapat dikenali dengan baik. Dengan kata lain, data yang memiliki tingkat ambiguitas yang tinggi akan sulit untuk dikenali oleh jaringan *ensemble* dengan algoritma pelatihan RBF.

4.5.3 Analisis Pengaruh Jumlah Jaringan Terhadap Tingkat Pengenalan

Pada percobaan yang dilakukan, terdapat 2 tahap penting yang dilakukan yaitu tahap pelatihan dan pengujian dengan tingkat pengenalan masing-masing. Pada bagian ini akan dianalisis pengaruh jumlah JST RBF *ensemble* yang digunakan terhadap tingkat pengenalan data pelatihan dan pengujian.

Pada JST RBF tunggal, faktor-faktor yang mempengaruhi kualitas tingkat pengenalan adalah ambiguitas set data pelatihan, interaksi antar kelas set data pelatihan, dan pemilihan fungsi *error*. Faktor-faktor tersebut masih menjadi faktor utama dari tingkat pengenalan pada tiap jaringan pada sistem *ensemble*. Namun sebagai satu sistem, JST RBF *ensemble* juga dipengaruhi dari efek NCL yang

diterapkan pada proses pelatihan. Seperti yang telah dijelaskan, NCL dipengaruhi oleh ambiguitas data pelatihan. Semakin banyak data ambigu, maka semakin kuat pengaruh NCL yang akan berakibat pada penurunan tingkat pengenalan. Namun seperti yang diperoleh pada hasil percobaan (gambar 4.3), penambahan jumlah jaringan yang digunakan pada percobaan tidak memiliki pengaruh yang kuat terhadap tingkat pengenalan kecuali untuk data *BUPA liver disorders* pada fungsi *error* kuadratis dan *Cross-entropy* dan *Glass detection* pada fungsi *error Cross-entropy*. Asumsi bahwa jumlah jaringan memiliki pengaruh pada tingkat pengenalan apabila terjadi selisih lebih dari 2% pada tingkat pengenalan.

Pada proses pengujian, tingginya tingkat pengenalan data pengujian sangat dipengaruhi oleh performa yang ditunjukkan proses pelatihan. Hal ini dikarenakan dalam pengujian, seluruh parameter pengujian bersumber dari hasil pelatihan. Oleh karena itu, apa yang mempengaruhi hasil pelatihan juga akan mempengaruhi hasil pengujian. Sama halnya pada jaringan tunggal, faktor hasil pelatihan bukannya faktor tunggal penentu kualitas tingkat pengenalan set data pengujian. Faktor lainnya yang juga memainkan peranan penting adalah pemilihan data pelatihan. Hal ini menjadi penting karena pada dasarnya JST bekerja berdasarkan pola-pola. Untuk medapatkan hasil pengujian yang baik, maka diperlukan pola pelatihan yang cocok dengan pola pengujian. Dengan kata lain, kesimpulan yang dapat ditarik adalah penambahan jumlah jaringan tidak mempengaruhi tingkat pengenalan data pengujian selama data yang digunakan untuk pengujian terwakili dengan baik oleh data pelatihan. Kesimpulan ini terbukti benar dari hasil yang ditunjukkan percobaan pada gambar 4.4, di mana terlihat bahwa hampir seluruh data tidak menunjukkan peningkatan dalam bidang tingkat pengenalan data pengujian kecuali set data *BUPA liver disorders* pada fungsi *error* kuadratis dan *Cross-entropy* dan *Glass detection* pada fungsi *error Cross-entropy*. Asumsi bahwa jumlah jaringan memiliki pengaruh pada tingkat pengenalan apabila terjadi selisih lebih dari 2% pada tingkat pengenalan.

4.5.4 Analisis Pengaruh Jumlah Jaringan Terhadap Waktu Komputasi

Pada percobaan ini, terdapat 2 komputasi waktu yang diperhitungkan yaitu komputasi waktu pelatihan dan pengujian. Waktu komputasi pelatihan adalah

waktu yang dibutuhkan untuk melatih JST hingga terpenuhinya kondisi henti, yaitu *error* hasil pelatihan dan jumlah epoh yang telah ditentukan sebelumnya. Sedangkan komputasi waktu pengujian adalah waktu yang dibutuhkan jaringan untuk melakukan komputasi guna mengenali seluruh set data pengujian.

Dari hasil analisis yang telah dilakukan pada butir 3.5.4, faktor yang memperngaruhi lamanya waktu komputasi data pelatihan pada jaringan tunggal adalah banyaknya data yang digunakan untuk pelatihan, jumlah atribut pada tiap data pelatihan, jumlah kelas pada set data pelatihan, pemilihan fungsi *error* dan jumlah epoh pelatihan. Pada percobaan ini, algoritma program percobaan untuk pelatihan terlihat lebih kompleks dibandingan dengan jaringan tunggal karena formula matematis yang digunakan dalam proses perubahan nilai bobot-bobot, nilai tengah dan lebar RBF untuk tiap jaringan. Dari faktor-faktor yang telah disebutkan di atas, dapat dipastikan bahwa akan terjadi peningkatan waktu komputasi pelatihan pada tiap jaringan. Karena jumlah jaringan yang digunakan pada percobaan ini adalah 3 dan 5, maka dapat dipastikan bahwa akan terjadi peningkatan waktu komputasi pelatihan untuk keseluruhan jaringan karena semakin banyak jaringan, semakin banyak komputasi yang harus dilakukan. Semakin banyak komputasi yang harus dilakukan akan membuat waktu komputasi menjadi semakin lama. Analisis ini terbukti benar dari hasil percobaan yang ditunjukkan pada gambar 4.7.

Sama halnya dengan waktu komputasi pelatihan, waktu komputasi untuk set data pengujian juga akan mengalami peningkatan dibandingkan dengan waktu komputasi pengujian jaringan tunggal. Berbeda dengan penyebab peningkatan waktu komputasi pelatihan, lamanya komputasi pengujian tidak dipengaruhi oleh formula-formula yang digunakan pada proses propagasi balik. Dengan demikian, kesimpulan yang sama juga dapat diterapkan pada jaringan *ensemble*, yaitu lama waktu komputasi pengujian sebanding dengan banyaknya jumlah jaringan yang digunakan. Semakin banyak jumlah jaringan, maka semakin banyak komputasi yang harus dilakukan. Semakin banyak komputasi yang perlu dilakukan, maka akan semakin lama waktu komputasi yang diperlukan. Hal ini terbukti benar dari hasil yang ditunjukkan oleh percobaan yang dilakukan dan dapat dilihat pada gambar 4.8.

4.6 Kesimpulan Hasil Percobaan JST RBF *Ensemble*

Dari percobaan dan analisis yang telah dilakukan, maka dapat ditarik beberapa kesimpulan berikut:

1. JST RBF *ensemble* tidak dapat bekerja pada data yang memiliki sifat ambigu.
2. Algoritma pelatihan dengan NCL meningkatkan pengaruh dari ambiguitas data sehingga menurunkan performa tingkat pengenalan data pelatihan dan pengujian.
3. Faktor yang mempengaruhi tingkat pengenalan data pelatihan adalah:
   a. Ambiguitas data. Semakin rendah tingkat ambiguitas di dalam set data pelatihan, maka semakin tinggi tingkat pengenalan JST RBF *ensemble*.
   b. Interaksi antar kelas. Semakin rendah interaksi antar kelas, maka semakin baik tingkat pengenalan JST RBF *ensemble*.
   c. Pemilihan fungsi *error*.
   d. NCL.
4. Faktor yang mempengaruhi tingkat pengenalan hasil pengujian adalah pemilihan data pelatihan dan hasil pelatihan yang baik.
5. Penambahan jumlah jaringan tidak mempengaruhi kedua tingkat pengenalan set data pelatihan dan pengujian.
6. Faktor yang mempengaruhi waktu komputasi pelatihan adalah:
   a. Jumlah data yang digunakan pada pelatihan.
   b. Jumlah atribut data pelatihan.
   c. Jumlah kelas dalam set data pelatihan.
   d. Pemilihan fungsi *error*.
   e. Banyaknya epoh pelatihan.
   f. Jumlah jaringan (*ensemble*).
7. Faktor yang mempengaruhi waktu komputasi pengujian adalah:
   a. Jumlah data yang digunakan pada pelatihan.
   b. Jumlah atribut data pelatihan.
   c. Jumlah kelas dalam set data pelatihan.
   d. Jumlah jaringan.

BAB 5
KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari keseluruhan penelitian skripsi ini dan saran untuk penelitian selanjutnya.

5.1 Kesimpulan

Dari percobaan yang telah dilakukan, berikut beberapa kesimpulan yang dapt ditarik oleh penulis:

1. Tingkat pengenalan data pelatihan pada JST RBF bergantung pada 3 faktor, yaitu:
   a. Tingkat ambiguitas data
   b. Interaksi antar kelas
   c. Pemilihan fungsi *error*
2. Tingkat pengenalan data pengujian bergantung pada pemilihan data pelatihan. Untuk mendapatkan hasil pengujian JST RBF yang baik, maka perlu diperhatikan pemilihan data pelatihan yang cukup mewakili data pengujian.
3. JST RBF memiliki kekurangan dari segi adaptasi terhadap data yang bersifat ambigu dan memiliki interaksi antar kelas yang kuat. Apabila dihadapkan pada set data demikian, maka kualitas performa tingkat pengenalan JST RBF ini akan menurun cukup jauh.
4. Dibandingkan dengan JST BP, JST RBF memiliki keuntungan dari segi waktu komputasi yakni 2 kali lebih cepat dengan tingkat pengenalan yang tidak jauh lebih buruk. Namun, keuntungan tersebut dapat dirasakan jika set data pelatihan memiliki tingkat ambiguitas dan interaksi antar kelas yang rendah.
5. Faktor yang mempengaruhi tingkat pengenalan data pelatihan untuk jaringan JST *ensemble* RBF sama dengan faktor yang mempengaruhi tingkat pengenalan JST tunggal RBF. Namun pada jaringan *ensemble*, terdapat pengaruh dari algoritma NCL yang pengaruhnya akan terasa pada set data yang memiliki data ambigu.
6. Untuk data pengujian, faktor yang mempengaruhi tingkat pengenalan pada jaringan ensemble adalah set data yang digunakan pada proses pelatihan.

Tingkat pengenalan data pengujian akan meningkat apabila terdapat data yang cukup mewakili set data pengujian pada set data pelatihan.

7. Tidak terdapat pengaruh dari penambahan jumlah *ensemble* terhadap tingkat pengenalan data pelatihan dan pengujian.
8. Terdapat kesinambungan antara waktu yang diperlukan untuk komputasi, baik waktu komputasi pelatihan maupun pengujian, dan jumlah jaringan yang digunakan. Semakin banyak jumlah jaringan yang digunakan, maka semakin lama pula waktu komputasi yang dibutuhkan.
9. Pemilihan fungsi *error* pada JST RBF, baik tunggal maupun *ensemble*, tidak dapat ditentukan secara langsung melainkan dari proses percobaan karena setiap set data dan jenis JST memiliki karakter sendiri dan hanya dapat didekati dengan fungsi *error* yang unik.
10. Berikut ringkasan perbandingan performa tingkat pengenalan data pengujian untuk jaringan tunggal dan ensemble:

Tabel 5.1. Ringkasan Performa JST RBF Tunggal dan *Ensemble*

No.	Nama Data	Kuadratis			*Cross-Entropy*		
		A	B	C	A	B	C
1	*Balance scale weight & distance*	✓					✓
2	*Breast cancer*		✓			✓	✓
3	*BUPA liver disorders*	✓			✓		
4	*Credit card approval*	✓	✓	✓		✓	✓
5	*Glass detection*	✓			✓		
6	*Hearth diseases*	✓			✓		
7	*Ionosphere*	✓			✓		
8	*Iris plants*	✓				✓	✓
9	Sonar	✓					✓
10	Citra wajah infra merah		✓	✓	✓		
11	Citra wajah cahaya tampak	✓			✓		

Keterangan:

A = JST tunggal RBF

B = JST dengan jumlah *ensemble* 3

C = JST dengan jumlah *ensemble* 5

5.2 Saran

Penulis menyarankan untuk menggunakan proses fuzzifikasi pada percobaan selanjutnya untuk mengurangi pengaruh dari ambiguitas data. Dengan demikian diharapkan performa tingkat pengenalan dapat meningkat dengan tetap mempertahankan kelebihan dari JST RBF, yakni di waktu komputasi. Selain itu, penulis juga menyarankan untuk menggunakan metode *C-Means Clustering* untuk mencari nilai tengah awal.

DAFTAR REFERENSI

Dam, H. H., Abbas, H. A., & Yao, X. (2008). Neural-Based Learning Classifier Systems. *IEEE Transaction on Knowledge Neural Network*, 36.

Er, M. J., Wu, S., Lu, J., & Toh, H. L. (2002). Face Recognition With Radial Basis Function (RBF) Neural Networks. *IEEE Transaction on Neural Networks*, 697-710.

George, M. (2007). Radial Basis Function Neural Networks and Principal Component Analysis for Pattern Classification. *IEEE Computer Society*, 200-206.

Ham, F. M., & Kostanic, I. (2000). *Principal of Neuro Computing for Science and Engineering.* New York: McGraw-Hill.

Imantaka, S. R. (2010). *Sistem Pengenal Wajah Berbasis Neural Network Ensemble untuk Citra Infra Merah.* Depok: Universitas Indonesia.

Lee, H., Hong, S., & Kim, E. (2009). Neural Network Ensemble with Probabilistic Fusion and Its Application to Gait Recognition. *Neurocomputing*, 1558.

Lin, M., Tang, K., & Yao, X. (2008). Selective Negative Correlation Learning Algorithm for Incremental Learning. *IEEE*, 2526-2531.

Liu, Y., & Yao, X. (1999). Ensemble Learning via Negative Correlation. *Pergamon Neural Networks 12*, 1299-1404.

Liu, Z.-Q., & Yan, F. (1997). Fuzzy Neural Network in Case-Based Diagnostic System. *IEEE Transaction on Fuzzy System*, 209-222.

Nasr, G., Badr, E., & Joun, C. (2002). Cross Entropy *Error* Function in Neural Networks: Forcasting Gasoline Demand. *FLAIRS-02 Proceedings*, 1-4.

Oh, S. H., & Lee, Y. (1995). A Modified *Error* Function To Improve the *Error* Backproapagation Algorithm for Multilayer Percepteron. *ETRI Jurnal*, 11-22.

Oliveira Soares, R. P., Castro, A. R., Oliveira, R. C., & Miranda, V. (2008). *Error* Entropy and Mean Square *Error* Minimization Algorithm for Neural Indentification of Supercritical Extraction Process. *IEEE Computer Society*, 75-80.

Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Cambridge: MA: Harvard University.

LAMPIRAN

A. Set Data UCI

Berikut keterangan set data UCI berjumlah 9 set data yang digunakan pada percobaan:

1. *Balance Scale Weight & Distance Database*

   Penulis : Siegler, R. S. (1976)

   Informasi : Data ini dihasilkan dari hasil percobaan psikologis. Tiap datanya diklasifikasikan menjadi tiga kelas, yakni memiliki keseimbangan ke kanan, ke kiri, atau seimbang. Atribut yang digunakan adalah bobot kanan, jarak kanan, bobot kiri, dan jarak kiri.

   Banyak data : 625 (49 seimbang, 288 kiri, dan 288 kanan)

   Jumlah atribut : 5

   Informasi atribut : 1. Nama kelas : 3 (L, B, R)
   2. Bobot kiri : 1 – 5
   3. Jarak kiri : 1 – 5
   4. Bobot kanan : 1 – 5
   5. Jarak kanan: 1 – 5

2. *Breast Cancer Database*

   Penulis : Dr. William H. Wolberg. (8 Januari 1991)

   Informasi : Data ini merupakan data kanker payu dara yang diambil secara periodik di klinik Dr. William H. Wolberg. Data ini digunakan untuk menentukan 2 jenis kanker yakni *benign* dan *malignant* berdasarkan 9 atribut.

   Banyak data : 699 (458 *benign* dan 241 *malignant*)

   Jumlah atribut : 11

   Informasi atribut : 1. Nomor kode data : nomor id
   2. Ketebalan *Clump* : 1 – 10
   3. Kesamaan ukuran sel : 1 – 10

4. Kesamaan bentuk sel : 1 – 10
5. Batas adesi : 1 – 10
6. Ukuran sel tunggal *Epithelial* : 1 – 10
7. *Bare Nuclei* : 1 – 10
8. *Bland Chromatin* : 1 – 10
9. *Normal Nucleoli* : 1 – 10
10. Mitosis : 1 – 10
11. Kelas : 2 untuk *benign* dan 4 untuk *malignant*

3. *BUPA Liver Disorders Database*

Penulis : BUPA Medical Research Ltd. (15 Mei 1990)

Informasi : Data ini diambil dari tes darah yang dilakukan untuk mentukan kelainan dari hati yang mungkin diakibatkan dari kebiasaan meminum minuman keras.

Banyak data : 345 (145 kelompok 1 dan 200 kelompok 2)

Jumlah atribut : 7

Informasi atribut : 1. *Mean corpuscular volume*
2. *Alkaline phosphotase*
3. *Alamine aminotransferase*
4. *Aspartate aminotransferase*
5. *Gamma-glutamyl transpeptidase*
6. Jumlah *half-pint* minuman keras yang diminum setiap hari
7. Kelompok data

4. *Credit Approval Database*

Penulis : Rahasia

Informasi : Data ini mengenai persetujuan pengajuan kartu kredit. Semua tribut dan kelas disamarkan guna merahasiakan informasi-informasi yang dianggap rahasia.

Banyak data : 690 (307 untuk kelas + dan 383 untuk kelas -)

Jumlah atribut : 16

Informasi atribut : 1. A1 : b, a
2. A2 : kontinu
3. A3 : kontinu
4. A4 : u, y, l, t
5. A5 : g, p, gg
6. A6 : c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff
7. A7 : v, h, bb, j, n, z, dd, ff, o
8. A8 : kontinu
9. A9 : t, f
10. A10 : t, f
11. A11 : kontinu
12. A12 : t, f
13. A13 : g, p, s
14. A14 : kontinu
15. A15 : kontinu
16. A16 : +, -

5. *Glass Identification Database*

Penulis : B. German (September 1987)

Informasi : Data ini diperoleh dari sebuah eksperimen terhadap kaca di mana pecahan kaca digunakan untuk menentukan benda apakah itu berdasarkan atribut-atribut seperti kadar Sodium, Natrium, Magnesium, dll di mana satuan yang digunakan adalah persentase berat di dalam oksida.

Banyak data : 214 (70 untuk kelas 1, 76 untuk kelas 2, 17 untuk kelas 3, 0 untuk kelas 4, 13 untuk kelas 5, 9 untuk kelas 6, dan 29 untuk kelas 7)

Jumlah atribut : 11

Informasi atribut : 1. Nomor id : 1 – 214
2. RI : *Refrective Index*
3. Na : Sodium
4. Mg : Magnesium

5. Al : Aluminium
6. Si : Silikon
7. K : Potasium
8. Ca : Kalsium
9. Ba : Barium
10. Fe : Besi
11. Tipe kelas :
    - 1 : *building_windows_float_processed*
    - 2 : *building_windows_non_float_processed*
    - 3 : *vehicle_windows_float_processed*
    - 4 : *vehicle_windows_non_float_processed*
    - 5 : kontainer
    - 6 : peralatan meja
    - 7 : lampu

6. *Hearth Diseases Database*

Informasi : Data ini diperoleh dari sebuah eksperimen yang bertujuan untuk menganalisis apakah seorang pasien menderita penyakit jantung apa tidak dengan menggunakan atribut-atribut seperti umur, ada tidaknya sakit di dada, jenis kelamin, dll.

Banyak data : 270 (150 penderita jantung dan 120 tidak penderita jantung)

Jumlah atribut : 14

Informasi atribut : 1. Umur
2. Jenis kelamin
3. Jenis sakit di dada (terdapat 4 nilai)
4. tekanan darah saat istirahat
5. *Serum cholestoral* dalam mg/dl
6. Gula darah saat istirahat > 120 mg/dl
7. Hasil *electrocardiographic* saat istirahat (0 – 2)
8. Kecepatan detak jantung maksimum

9. *Exercise induced angina*
10. *Oldpeak* = depresi ST yang diakibatkan oleh olah raga relatif terhadap saat istirahat
11. Kemiringan dari puncak latihan segmen ST
12. Banyaknya nadi utama (0 – 3) yang diwarnai oleh *flourosopy*
13. Thal: 3 = normal, 6 = cacat tetap, 7 = cacat sementara
14. Tipe kelas: 1 untuk tidak menderita panyakit jantung dan 2 untuk menderita penyakit jantung

7. *Ionosphere Database*

Penulis : Johns Hopkins University (1989)

Informasi : Data ini diperoleh dari sistem di Goose Bay, Labrador. Sistem ini terdiri dari 16 antena frekuensi tinggi. Target dari sistem ini adalah elektron yang ada di lapisan ionosfer. "good" menandakan adanya struktur tertentu di ionosfer dan "bad" menandakan tidak ada struktur di sana.

Banyak data : 351

Jumlah atribut : 35

Informasi atribut : Atribut 1 – 34 merupakan imformasi hasil pengolahan data dan atribut ke-35 merupakan kelas yang berisi "g" atau "b".

8. *Iris Plants Database*

Penulis : R.A. Fisher (juli 1988)

Informasi : Data ini terdiri dari 3 kelas dengan 50 data untuk masing-masing kelas di mana tiap kelas mewakili sebuah jenis bunga iris yakni Iris Setosa, Iris Versicolour, dan Iris Virginica. Setiap kelas dapat dipisahkan secara linear dari yang lainnya.

Banyak data : 150 (50 Iris Setosa, 50 Iris Versicolour, dan 50 Iris Virginica)

Jumlah atribut : 5

Informasi atribut : 1. Panjang mahkota bunga dalam cm

2. Tebal mahkota bunga dalam cm

3. Panjang kelopak bunga dalam cm

4. Tebal kelopak bungan dalam cm

5. Jenis kelas : 1 untuk Iris Setosa, 2 untuk Iris Versicolour, dan 3 untuk Iris Virginica

9. *Sonar Database*

Informasi : Data ini merupakan rekaman tingkat energi pada 60 frekuensi yang berbeda untuk menentukan perbedaan antara benda metal dan batu.

Banyak data : 208 (111 benda metal dan 97 benda batu)

Jumlah atribut : 61

Informasi atribut : atribut 1 – 60 merupakan tingkat energi pada 60 frekuensi yang berbeda dan atribut ke-61 menunjukkan jenis kelas (M untuk metal dan R untuk batu)

B. Set Data Citra Wajah Infra Merah

Set data percobaan citra wajah manusia infra merah berjumlah 200 foto dalam ukuran aslinya:

1. Data A

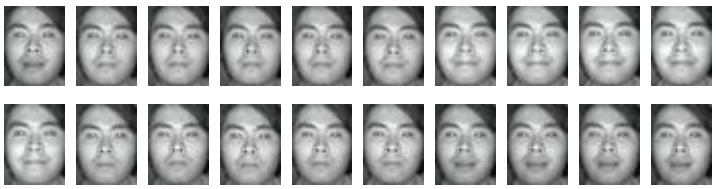
2. Data B

3. Data C

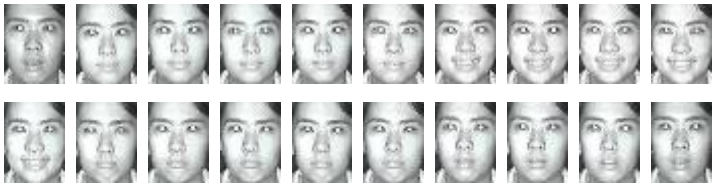
4. Data D

5. Data E

6. Data F



7. Data G



8. Data H

9. Data I

10. Data J

C. Set Data Citra Wajah Cahaya Tampak

Set data percobaan citra wajah manusia infra merah berjumlah 100 foto dalam ukuran aslinya:

1. Data A

2. Data B

3. Data C

4. Data D

5. Data E

6. Data F

7. Data G

8. Data H

9. Data I

10. Data J

D. Self-Organizing Map (SOM)

Self-Organizing Map (SOM) dikembangkan pertama kali oleh Kohenen merupakan sebuah suatu jaringan saraf tiruan yang dilatih secara tidak diarahkan (*unsupervised*) dan bekerja berdasarkan pengelompokan data di mana hanya terdapat satu neuron yang "nyala" dalam satu waktu (Ham & Kostanic, 2000). Pada penerapannya, SOM dapat mengelompokkan data dengan cara mencari jarak terdekat suatu data terhadap kumpulan vektor tiap kelompok yang diperoleh dari hasil pembelajaran. Berikut algoritma SOM:

1. Inisialisasi

   Vektor pewakil ($\boldsymbol{w}_i$), laju pembelajaran (α), dan konstanta laju pembelajaran (c)

2. Pembelajaran

   a. Mencari vektor pemenang

   $$d(\boldsymbol{x}) = \min_{\forall i} \|\boldsymbol{x} - \boldsymbol{w}_i\|$$

   b. Mengubah vektor pemenang

   $$w_i(k) = \begin{cases} w_i(k) + \alpha(k)[x - w_i(k)] & \text{jika } \|\boldsymbol{x} - \boldsymbol{w}_i\| = d(\boldsymbol{x}) \\ w_i(k) & \text{lainnya} \end{cases}$$

   c. Mengubah laju pembelajaran

   $$\alpha(k+1) = \alpha(k) \times c$$

   d. Ulangi pelajaran hingga nilai laju pembelajaran memenuhi kondisi henti

   $$\alpha(k) = 0.01$$

E. Hasil Pengelompokan SOM untuk Set Data Pelatihan dan Pengujian

No	Nama data	Data pelatihan		Data pengujian	
		Jumlah data	Jumlah data ambigu	Jumlah data	Jumlah data ambigu
1.	*Balance scale weight and distance*	75	51	72	37
2.	*Breast cancer*	242	17	240	11
3.	*BUPA liver disorders*	146	72	144	72
4.	*Credir card approval*	308	149	306	150
5.	*Glass detection*	30	16	24	16
6.	*Hearth diseases*	120	59	120	60
7.	*Ionosphere*	126	27	126	36
8.	*Iris plants*	75	7	75	8
9.	*Sonar*	98	41	96	48
10.	Citra wajah infra merah	100	0	100	5
11.	Citra wajah cahaya tampak	50	0	50	0

F. Hasil Pengelompokan SOM untuk Set Data *Iris Plants*

No. data	Data pelatihan		Data pengujian	
	Tanpa SOM	Hasil SOM	Tanpa SOM	Hasil SOM
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	1	1	1	1
5	2	2	2	2
6	3	2	3	2
7	1	1	1	1
8	2	2	2	2
9	3	3	3	2
10	1	1	1	1
11	2	2	2	2
12	3	3	3	3
13	1	1	1	1
14	2	2	2	2
15	3	3	3	3
16	1	1	1	1
17	2	2	2	2
18	3	3	3	3
19	1	1	1	1
20	2	2	2	2
21	3	2	3	3
22	1	1	1	1
23	2	2	2	2
24	3	3	3	3
25	1	1	1	1
26	2	2	2	2
27	3	3	3	2
28	1	1	1	1
29	2	2	2	2
30	3	3	3	2
31	1	1	1	1
32	2	2	2	2
33	3	3	3	3
34	1	1	1	1
35	2	2	2	2
36	3	3	3	3
37	1	1	1	1
38	2	2	2	2
39	3	3	3	3
40	1	1	1	1
41	2	2	2	2
42	3	2	3	2
43	1	1	1	1
44	2	2	2	2
45	3	2	3	3
46	1	1	1	1
47	2	2	2	2
48	3	3	3	3
49	1	1	1	1
50	2	2	2	2
51	3	3	3	3
52	1	1	1	1
53	2	2	2	2
54	3	3	3	2
55	1	1	1	1
56	2	2	2	2
57	3	3	3	3
58	1	1	1	1
59	2	2	2	2
60	3	2	3	3
61	1	1	1	1
62	2	2	2	2
63	3	3	3	3
64	1	1	1	1
65	2	2	2	2
66	3	2	3	2
67	1	1	1	1
68	2	2	2	2
69	3	3	3	3
70	1	1	1	1
71	2	2	2	2
72	3	2	3	3
73	1	1	1	1
74	2	2	2	2
75	3	3	3	2