# JARINGAN SARAF TIRUAN PROPAGASI BALIK UNTUK KLASIFIKASI DATA

**Giri Dhaneswara[1)] dan Veronica S. Moertini[2)]**
Jurusan Ilmu Komputer, Universitas Katolik Parahyangan, Bandung
Email: [1)]rebirth_82@yahoo.com, [2)]moertini@home.unpar.ac.id

***Intisari***

*Data yang tersimpan di basisdata dapat dianalisis dengan teknik klasifikasi data jaringan saraf tiruan (JST) propagasi balik dan dimanfaatkan untuk mengambil keputusan. Makalah ini akan membahas JST ini, analisis-perancangan-implementasi dari JST ini, eksperimen-eksperimen untuk mendapatkan konfigurasi JST yang terbaik dan penggunaannya untuk mengklasifikasi data aplikasi kredit di sebuah bank.*

*Kata kunci: propagasi balik, jaringan saraf tiruan, klasifikasi data*

***Abstract***

*Data strored in databases could be analized using a classification technique of back-propagation neural networks (BP NN) and utilized in supporting decission making processes. This paper discusses BP NN, the experiments conducted to discover the best configuration of the NN and its use in classifying banking credit application data.*

*Keywords: backpropagation, neural network, data classification*

Diterima : 6 Agustus 2004
Disetujui untuk dipublikasikan : 18 Agustus 2004

## 1. Pendahuluan

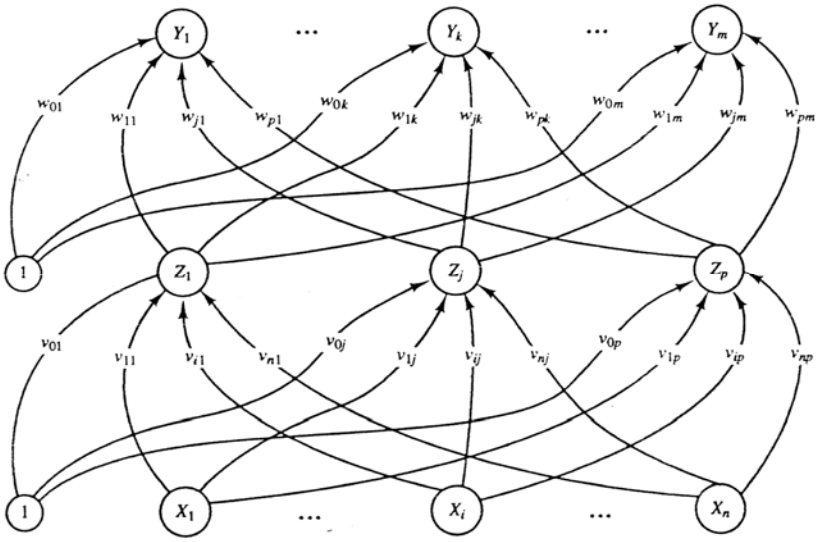
Banyak data yang tersimpan di basisdata yang dapat dianalisis dan dimanfaatkan untuk membantu proses pengambilan keputusan. Misalnya: (a) Untuk menentukan apakah sebuah aplikasi kredit di bank akan diterima atau tidak. (b) Untuk memilih jenis servis asuransi yang paling sesuai untuk seorang kustomer. (c) Di lingkungan perguruan tinggi: untuk memprediksi calon mahasiswa apakah kelak akan berprestasi atau *drop-out*. Salah satu teknik yang digunakan untuk menganalisis data untuk keperluan ini adalah teknik klasifikasi data.

Klasifikasi data terdiri dari dua langkah yang utama, yaitu: (1) Pembangunan model dari set data pelatihan. Pada jaringan saraf tiruan (JST), ini dilakukan dengan "pembentukan" jaringan dan penghitungan nilai-nilai parameter jaringan (bobot, *bias*, dll.,). (2) Penggunaan model untuk mengklasifikasi data baru. Di sini, sebuah rekord "diumpankan" ke model, dan model akan memberikan jawaban "kelas" hasil perhitungannya.

Makalah ini akan membahas JST propagasi balik *(backpropagation)*, analisis-perancangan-implementasi sistem JST ini, eksperimen-eksperimen untuk mendapatkan konfigurasi JST yang terbaik dan penggunaannya untuk mengklasifikasi data aplikasi kredit bank, serta kesimpulan dan arah penelitian lanjutan.

## 2. Jaringan Saraf Tiruan (JST) Propagasi Balik

JST propagasi balik adalah JST dengan topologi multi-lapis (*multilayer)* dengan satu lapis masukan (lapis *X),* satu atau lebih lapis *hidden* atau tersembunyi (lapis *Z*) dan satu lapis keluaran (lapis *Y*). Setiap lapis memiliki neuron-neuron (unit-unit) yang dimodelkan dengan lingkaran (lihat Gambar 1). Di antara neuron pada satu lapis dengan neuron pada lapis berikutnya dihubungkan dengan model koneksi yang memiliki bobot-bobot (*weights*), **w** dan **v**. Lapis tersembunyi dapat memiliki *bias*, yang memiliki bobot sama dengan satu [1].




Gambar 1. Jaringan saraf propagasi balik dengan satu lapis tersembunyi.

### 2.1. Algoritma Pelatihan JST Propagasi balik

Algoritma pelatihan JST Propagasi balik pada dasarnya terbagi menjadi 2 langkah, yaitu: langkah maju (*feedforward)*) dan propagasi balik (*back propagation)*. Pada langkah maju, perhitungan bobot-bobot neuron hanya didasarkan pada vektor masukan, sedangkan pada propagasi balik, bobot-bobot diperhalus dengan memperhitungkan nilai target atau keluaran. Algoritma pelatihan maju dan propagasi balik ini ditunjukkan pada Algoritma 1 [1].

Nilai *mean square error* (MSE) pada satu siklus pelatihan (langkah 2 – 10, dimana seluruh rekord dipresentasikan satu kali) adalah nilai kesalahan (*error* = nilai keluaran - nilai masukan) rata-rata dari seluruh rekord (tupple) yang dipresentasikan ke JST dan dirumuskan sebagai:

$$MSE = \left(\frac{\sum error^2}{jumlah\ rekord}\right)$$

Semakin kecil MSE, JST semakin kecil kesalahannya dalam memprediksi kelas dari rekord yang baru. Maka, pelatihan JST ditujukan untuk memperkecil MSE dari satu siklus ke siklus berikutnya sampai selisih nilai MSE pada siklus ini dengan siklus sebelumnya lebih kecil atau sama dengan batas minimal yang diberikan (epsilon).

### 2.2. Pengembangan Algoritma Pelatihan dengan Menggunakan Momentum

Salah satu pengembangan algoritma pelatihan propagasi balik dilakukan dengan penggunaan momentum pada perhitungan perubahan bobot-bobot. Tujuan dari pengembangan ini adalah untuk melancarkan pelatihan dan mencegah agar bobot tidak "berhenti" di sebuah nilai yang belum optimal. Perubahan bobot pada Algoritma 1 diubah menjadi [1]: $\Delta w_{jk}(t+1) = \alpha\delta_k z_j + \mu\Delta w_{jk}(t)$ dan $\Delta v_{ij}(t+1) = \alpha\delta_j x_i + \mu\Delta v_{ij}(t)$, dimana μ merupakan konstanta dari momentum dengan rentang [0,1].

### 2.3. Klasifikasi Data

Setelah pelatihan selesai dilakukan, maka model JST siap digunakan untuk mengklasifikasi rekord baru. Klasifikasi ini cukup dilakukan langkah *feedforward* dan hasilnya adalah berupa kelas yang diprediksi JST untuk rekord yang baru ini.

**Narasi**: Melatih JST dengan data pelatihan (berupa vektor atau rekord-rekord tabel) yang diberikan sampai bobot-bobot tidak berubah lagi (atau dicapai kondisi konvergen).
**Input**: Set data pelatihan , jumlah lapis, jumlah neuron, *learning rate*, epsilon
**Output**: Model JST yang siap untuk mengklasifikasi data (vektor) baru.
**Algoritma**:
(1) Menginisialisasi bobot awal dengan nilai acak yang sangat kecil, hitung MSE dan ΔMSE inisialisasi.
(2) Selama ΔMSE > epsilon lakukan:
(3) Untuk setiap *tupple* pada set data pelatihan lakukan

***Feedforward:***

(4) Setiap unit masukan ($X_i$, i=1..n) menerima vektor masukan $X_i$ dan mengirimkan vektor ini ke seluruh unit pada lapis diatasnya (*hidden layer*).

(5) Setiap unit *hidden* ($Z_j$, j=1..p) menjumlahkan bobot dari vektor masukan:

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$ Hitung keluaran fungsi aktivasi: $z_j = f(z\_in_j)$

Kirimkan vektor ini ke unit-unit pada *lapis* diatasnya (lapis keluaran)

(6) Setiap unit keluaran ($Y_k$, k=1..m) menjumlahkan vektor masukan:.

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

Hitung keluaran dari fungsi aktivasi: $y_k = f(y\_in_k)$

**Propagasi balik dari *error:***

(7) Setiap unit keluaran ($Y_k$, k=1..m) menerima vektor hasil yang diinginkan ($t_k$) untuk data masukan tersebut, hitung *error*-nya $(t_k - y_k)$: $\delta_k = (t_k - y_k) f'(y\_in_k)$

Hitung nilai koreksi bobotnya dengan α sebagai *learning rate*nya: $\Delta w_{jk} = \alpha \delta_k z_j$

Hitung nilai koreksi *bias*nya: $\Delta w_{0k} = \alpha \delta_k$

Kirimkan $\delta_k$ ke unit pada *lapis* dibawahnya.

(8) Setiap unit *hidden* ($Z_j$, j=1..p) menjumlahkan delta masukannya (dari unit-unit pada lapis diatasnya):

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

Kalikan dengan turunan dari fungsi aktivasinya untuk menghitung *error*nya: $\delta_j = \delta\_in_j f'(z\_in_j)$ Hitung nilai koreksi bobotnya: $\Delta v_{ij} = \alpha \delta_j x_i$ Hitung nilai koreksi *bias*nya: $\Delta v_{0j} = \alpha \delta_j$

**Perbaharui bobot dan *bias*:**

(9) Setiap unit keluaran ($Y_k$, k=1..m) memperbaharui *bias* dan bobotnya (j=0..p).

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

Setiap unit tersembunyi ($Z_j$, j=1..p) memperbaharui *bias* dan bobotnya (i=0..n).

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$

MSEold = MSE. Hitung MSE, ΔMSE = MSE – MSEold.

(10) Uji kondisi berhentinya

Algoritma 1. Algoritma pelatihan pada JST Propagasi balik.

Keterangan Notasi untuk Algoritma 1: $x$ = vektor masukan = $(x_1,...,x_i,...,x_n)$, $t$ = vektor keluaran= $(t_1,...,t_k,...,t_m)$. $\delta_k$ = nilai koreksi bobot *error* untuk $w_{jk}$ yang disebabkan oleh *error* pada unit keluaran $Y_k$. $\delta_j$ = nilai koreksi bobot *error* untuk $v_{ij}$ yang disebabkan oleh informasi propagasi balik dari *error* pada *lapis* keluaran ke unit tersembunyi $Z_j$. $\alpha$ = konstanta laju pembelajaran (*learning rate*). $X_i$ =unit masukan i. $v_{0j}$ = bias pada unit tersembunyi j. $Z_j$ =unit tersembunyi j. $w_{0k}$ =bias pada unit keluaran k. $Y_k$ = unit keluaran k. Fungsi aktivasi yang dapat digunakan pada propagasi balik antara lain adalah: Binary sigmoid: $f(x)=\frac{1}{1+e^{-x}}$, arc tangen: $f(x)=\frac{2}{\pi}\arctan(x)$, radial basis: $f(x)=e^{-\sqrt{x}}$.

## 3. Analisis, Perancangan dan Implementasi JST Propagasi Balik

### 3.1. Analisis Sistem JST

Sistem JST Propagasi balik yang dibangun dimaksudkan untuk mengklasifikasi data (apa saja) yang tersimpan di tabel-tabel basisdata. Agar sistem ini berfungsi dan berperformansi dengan baik, ada beberapa isu dan solusi yang perlu dipertimbangkan yang terkait dengan set data pelatihan dan performansi sistem.

**a. Set Data Pelatihan**

Isu: (a) Bebas. (b) Jumlah, tipe kolom tabel, dan jumlah kelas pada kolom keluaran (target) bervariasi. (c) Tipe kolom: kontinyu (dengan rentang bervariasi) atau diskret, sedangkan JST hanya mengolah tipe kontinyu. (d) Perlu data validasi yang diambil dari data pelatihan.

Solusi sistem: (a) Fitur untuk memilih tabel yang berisi dara pelatihan. (b) Fitur untuk memilih kolom yang akan digunakan untuk melatih JST dan mendefinisikan tipe dari tiap kolom (c) Sistem JST secara otomatis menghitung jumlah unit (neuron) di lapis masukan dan keluaran sesuai dengan data pelatihan. (d) Fitur untuk transformasi data. (e) Fitur untuk mendefinisikan prosentase dan cara pemilihan data validasi dari tabel. (f) Fitur untuk memvalidasi model JST hasil pelatihan.

**b. Performansi Sistem JST**

Isu: (a) Jumlah atau nilai lapis tersembunyi, unit di lapis(-lapis) ini, fungsi aktivasi, *learning-rate*, dan momentum yang optimal bergantung kepada karakteristik set data pelatihan. (b) Tingkat ketelitian yang diperlukan mungkin bergantung kepada tipe data pelatihan. (c) Model hasil pelatihan dapat digunakan untuk memprediksi data baru kapan saja.

Solusi sistem: (a) Fitur untuk mengambil nilai-nilai ini. (b) Fitur untuk mengamati statistik proses dan hasil pelatihan. (c) Fitur untuk mendefinisikan nilai epsilon dan jumlah siklus pelatihan. (d) Fitur untuk menyimpan model JST hasil pelatihan.

Transfomasi data kontinyu dimaksudkan untuk menormalisasi data dan dilakukan dengan rumus [BER97]

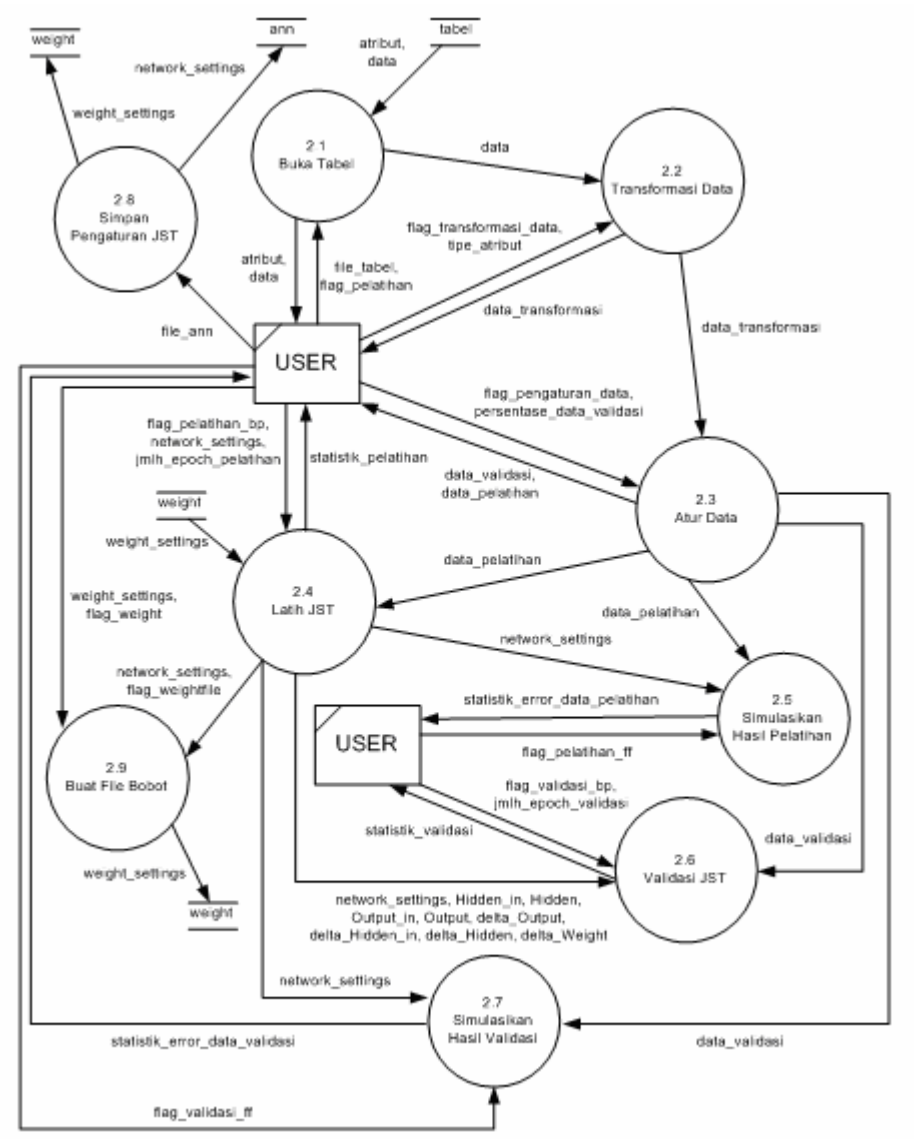
$$nilai\_baru = \frac{nilai\_asli - batas\_bawah}{batas\_atas - batas\_bawah}$$

. Sedangkan kolom tipe diskret (kategorial) ditransformasi menjadi kontinyu. Transformasi pada sistem JST ini dilakukan dengan cara sederhana, yaitu rentang nilai [0,1] dibagi menjadi rentang-rentang homogen sebanyak nilai pada kolom bertipe diskret dikurangi 1, lalu setiap nilai diskret diasosiasikan dengan nilai pada batas rentang-rentang baru ini.

Jumlah unit (neuron) lapis keluaran bergantung kepada jumlah kelas yang ada di kolom kelas dan didapat dengan rumus: *m* = pembulatan ke atas ($log_2$(jumlah kelas))

***Data flow diagram***

*Data flow diagram* (DFD) dari proses pelatihan pada sistem JST ditunjukkan pada Gambar 2 (pemodelan sistem yang lengkap dan keterangan dari DFD ini dapat dilihat di [DAN04]). Pada gambar ini ditunjukkan bahwa ada proses-proses lain yang terkait dengan proses Latih JST, karena sebelum set data pelatihan siap untuk dipresentasikan, data harus ditransformasi menjadi nilai kontinyu (proses Transformasi Data) dan ada pemilihan data yang akan digunakan untuk validasi (proses Atur Data). Pengguna juga dapat menginisialisasi bobot-bobot JST secara manual (proses Buat File Bobot). Setelah proses Latih JST dijalankan, pengguna dapat mengeksekusi proses Validasi untuk memperhalus bobot-bobot JST dan Simulasikan Hasil Validasi untuk memvisualisasi statistik proses validasi. Pengguna juga dapat menyimpan bobot-bobot JST ke sebuah file untuk keperluan klasifikasi rekord baru.




Gambar 2.DFD untuk proses pelatihan JST *Propagasi balik*.

### 3.2. Perancangan Struktur Data

Semua struktur data yang digunakan untuk menyimpan bobot, bias, masukan-keluaran fungsi aktivasi, kesalahan dan perubahan bobot adalah *array* dinamis bertipe real, dimana dimensi dari array ini dibuat sesuai dengan parameter masukan pengguna. Sebagai contoh, berikut ini adalah deklarasi array dinamis untuk bobot-bobot di lapis masukan dan tersembunyi (untuk JST dengan 1 lapis tersembunyi):

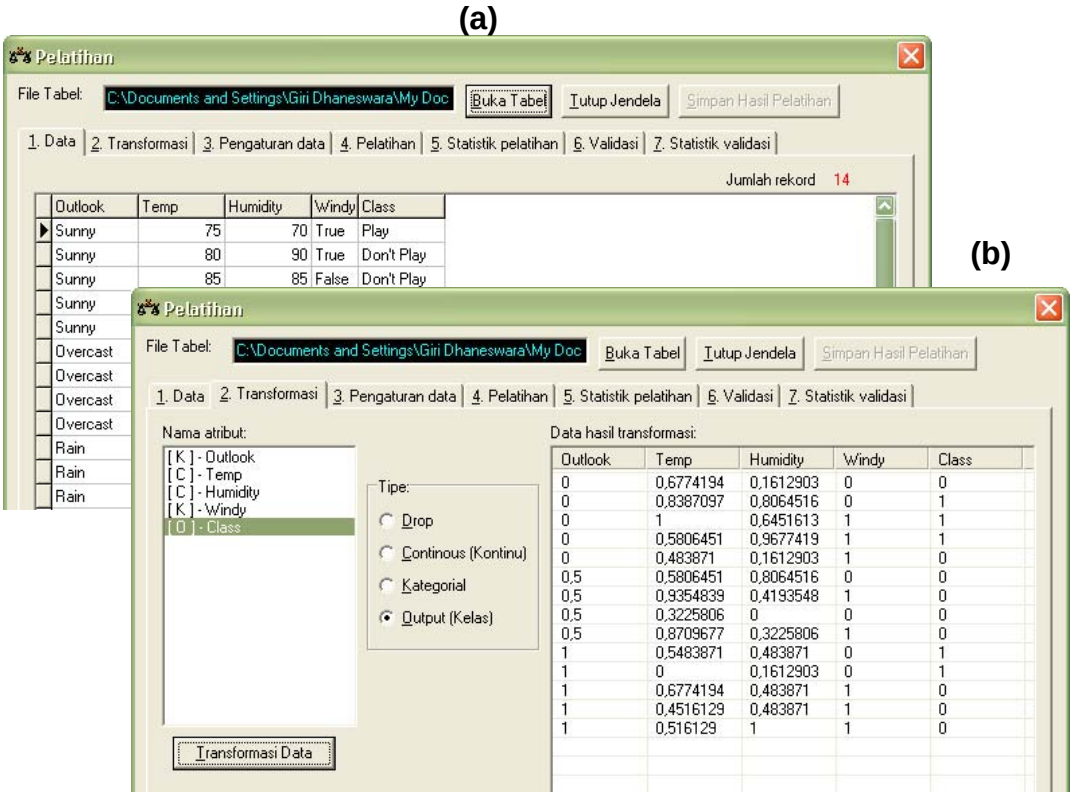
```
InWeights: Array of Array
of Single;
H1Weights: Array of Array
of Single;
SetLength(InWeights,
NrOfInputs, NrOfHidden1);
```

```
SetLength(H1Weights,
NrOfHidden1,
NrOfOutputs);
```

Dengan NrOfInputs adalah jumlah neuron lapis masukan, NrOfHidden1 adalah jumlah neuron lapis tersembunyi 1 dan NrOfOutputs adalah jumlah neuron lapis keluaran.

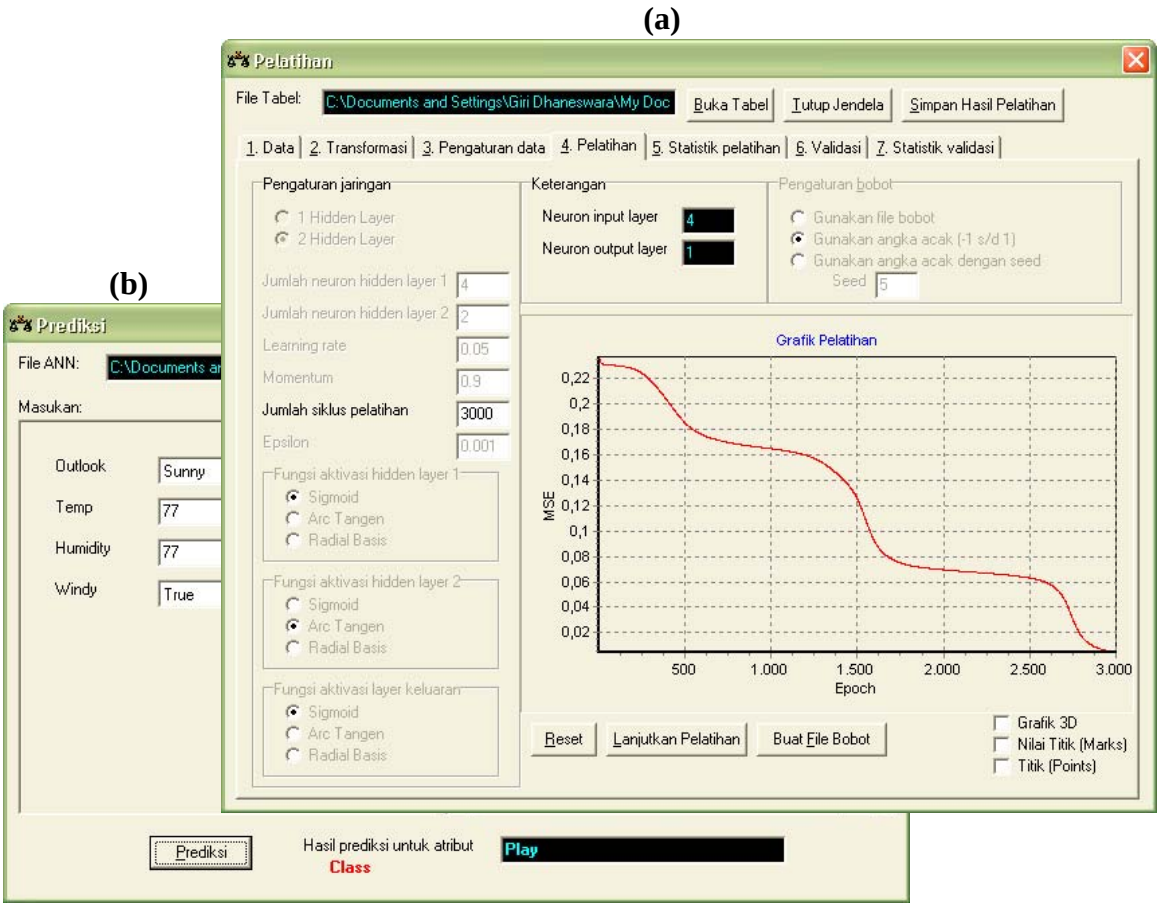
### 3.3. Implementasi Jendela-Jendela Sistem JST

Beberapa dari jendela utama pada sistem JST ini diberikan pada Gambar 3 dan Gambar 4. Jendela pada Gambar 3 berkaitan dengan fitur penyiapan data pelatihan dengan fitur untuk: (1) Membuka atau memilih tabel yang akan digunakan untuk melatih JST (Gambar 3.a.) (2) Mendefinisikan tipe-tipe kolom tabel, memilih kolom sebagai kelas (target) dan mentransformasi isi-isi kolom ke dalam nilai kontinyu yang siap dipresentasikan ke JST (Gambar 3.b). Setelah langkah ini, pengguna dapat memilih data yang akan digunakan untuk validasi melalui tombol (menu) Pengaturan data.




Gambar 3. (a) Pembukaan tabel set data pelatihan. (b). Pendefinisian tipe kolom dan transformasi data.

Setelah data ditransformasi, maka pengguna dapat melatih JST dengan memilih menu Pelatihan dan selama proses pelatihan berlangsung, pengguna dapat mengamati perubahan MSE dari siklus (*epoch*) ke ke siklus (Gambar 4.a). Setelah tahap ini selesai, pengguna dapat melihat statistik hasil pelatihan (MSE rata-rata, MSE terendah, waktu pelatihan, dll.), melakukan validasi dengan memilih menu Validasi dan mengamati statistik hasil validasi. Jika semua statistik dapat diterima, maka JST siap digunakan untuk mengklasifikasi rekord baru (Gambar 4.b.).




Gambar 4. (a). Jendela pelatihan JST. (b) Jendela prediksi untuk rekord baru.

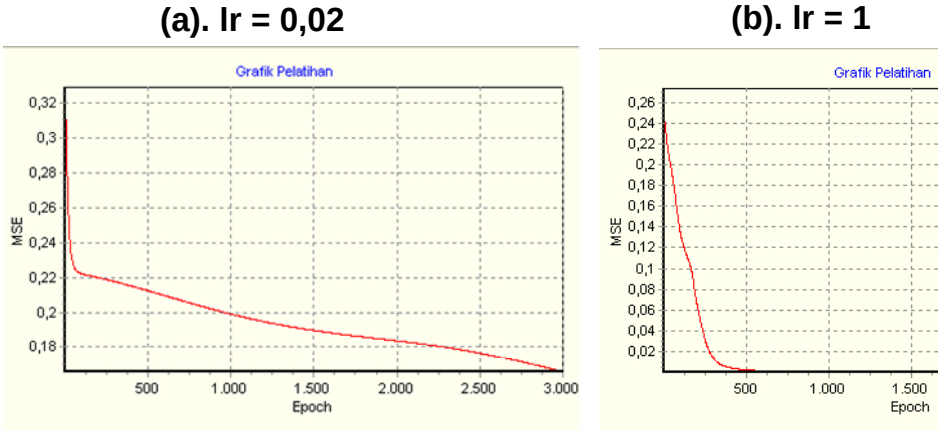
## 4. Eksperimen

Pada penelitian ini, ada dua kelompok eksperimen yang akan dilakukan: (1) Eksperimen yang dilakukan dengan data simulasi dan terdiri dari 5 sub-kelompok eksperimen. Eksperimen-eksperimen pada kelompok ini bertujuan untuk mencari yang terbaik dari nilai *learning rate*, momentum, fungsi aktivasi, jumlah neuron pada lapis tersembunyi dan jumlah lapis tersembunyi. (2) Eksperimen yang dilakukan dengan data nyata, yaitu data aplikasi kredit di sebuah bank. Eksperimen ini bertujuan untuk mengetahui apakah sistem JST dapat difungsikan untuk mengklasifikasi data nyata. Karena keterbatasan ruang, pada makalah ini hanya diberikan hasil eksperimen dan bahasan utama sedangkan bahasan yang lengkap dapat dilihat di [2].

### 4.1. *Learning rate*

Konfigurasi JST yang digunakan adalah: 1 lapis tersembunyi, jumlah neuron pada lapis tersembunyi = 4, momentum = 0, jumlah siklus pelatihan maksimum = 3000, epsilon = 0,00001, fungsi aktivasi pada lapis tersembunyi dan lapis keluaran adalah sigmoid dan bobot-bobot diinisialisasi dengan nilai acak. Pada eksperimen ini, JST dilatih dengan nilai *learning rate* (*lr*) 0,02, 0,125,

0,25 dan 1. Pada setiap pelatihan, grafil MSE terhadap siklus pelatihan (*epoch*) diamati. Sebagai contoh, dua grafik hasil eksperimen diberikan pada Gambar 5.

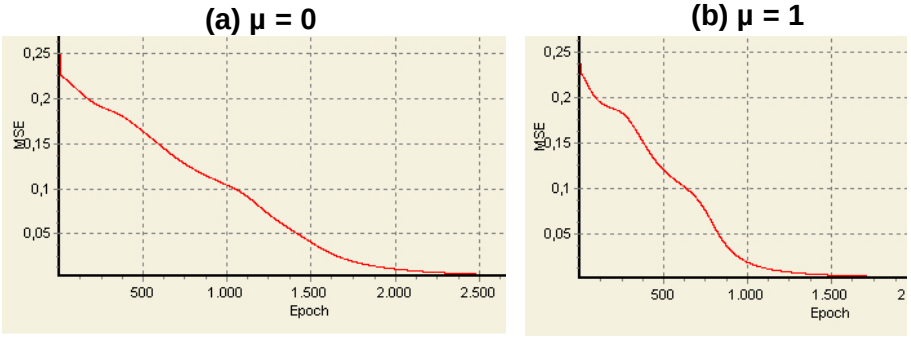



Gambar 5. Grafik MSE terhadap siklus pelatihan (*epoch*) untuk 2 nilai *learning rate* (lr).

Dari hasil eksperimen didapati bahwa makin besar *lr* maka MSE makin cepat turun atau makin cepat JST belajar. Hasil ini sesuai dengan yang diharapkan, karena *lr* memang merupakan laju dari JST dalam belajar. Hanya saja, sebagai kosekuensinya, tingkat ketelitian JST dalam perhitungan penyesuaian bobot menjadi berkurang.

**4.2. Momentum**

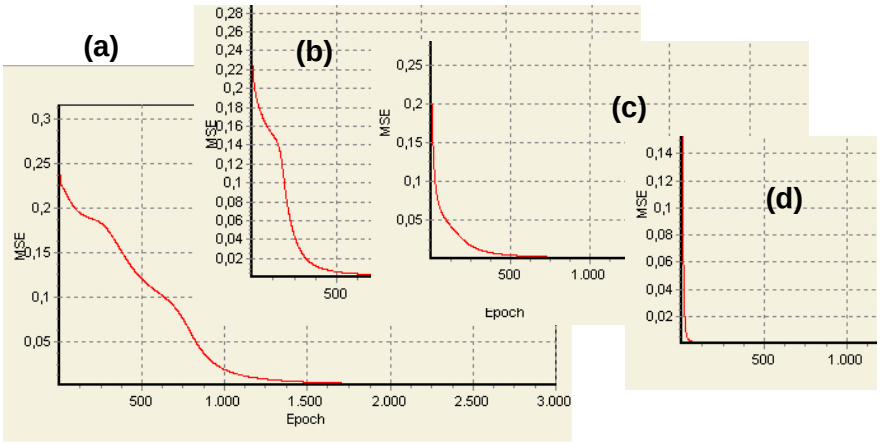
Pada eksperimen ini, konfigurasi JST yang digunakan sama dengan eksperimen *lr*, hanya saja nilai *lr* dibuat konstan (0,125) dan nilai momentum, μ, diubah-ubah: 0; 0,2; 0,5 dan 1. Dari hasil eksperimen didapati bahwa makin besar momentum, makin cepat proses pelatihan JST. Juga didapati bahwa perubahan nilai momentum tidak berbanding lurus dengan perubahan MSE. Sebagai contoh hasil eksperimen, pada Gambar 6 diberikan grafik pelatihan (MSE vs siklus) untuk μ= 0 dan μ = 1.




Gambar 6. Grafik MSE terhadap siklus pelatihan (epoch) untuk: (a) momentum = 0, (b) momentum = 1.

### 4.3. Fungsi Aktivasi

Konfigurasi JST pada eksperimen ini sama dengan yang digunakan pada eksperimen momentum, dengan momentum = 1. Pasangan fungsi yang diobservasi di lapis tersembunyi dan keluaran pada eksperimen ini adalah kombinasi dua (keduanya boleh sama) dari tiga fungsi aktivasi sigmoid, arc tangen dan radial basis, contohnya: sigmoid – sigmoid, sigmoid – arc-tangen , arc-tangen – sigmoid, sigmoid - radial basis, radial basis – arc tangen, radial basis – radial basis, dll. Sebagai contoh hasil eksperimen, pada Gambar 7 diberikan grafik MSE vs siklus dari 4 pasangan fungsi aktivasi.

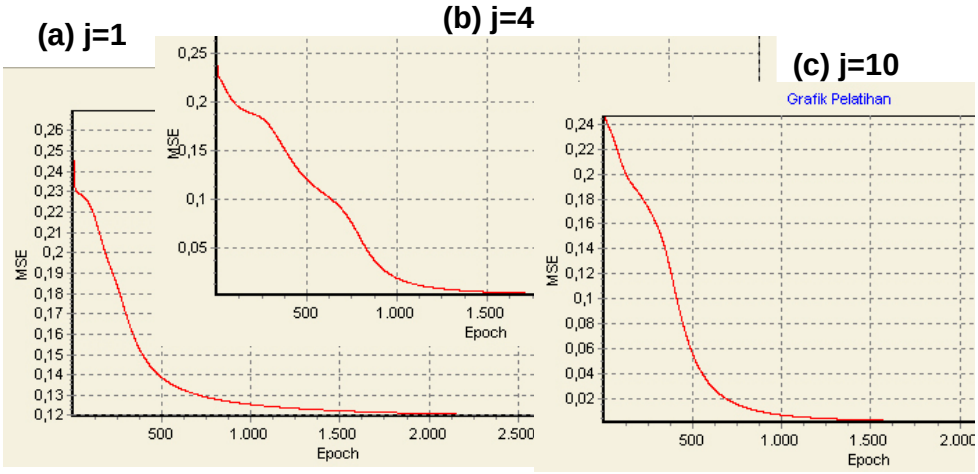



Gambar 7. Grafik MSE terhadap siklus pelatihan (*epoch*) untuk 4 kombinasi fungsi aktivasi di lapis tersembunyi dan lapis keluaran. (a) sigmoid - sigmoid (b) arc-tangen – sigmoid (c). radial basis – sigmoid (d) radial basis – radial basis

Hasil keseluruhan eksperimen menunjukkan bahwa fungsi aktivasi berperan sangat penting dalam proses pelatihan JST. Penggunaan fungsi aktivasi radial basis ternyata mempercepat waktu pelatihan (Gambar 7.c dan 7.d) dan pelatihan tercepat terjadi ketika pasangan fungsi aktivasi adalah radial basis - radial basis (Gambar 7.d).

### 4.4. Jumlah Neuron pada Lapis Tersembunyi

Konfigurasi JST yang digunakan pada eksperimen ini mirip dengan pada eksperimen learning rate. Hanya saja, sekarang jumlah neuron (unit) pada lapis tersembunyi diubah-ubah menjadi 1, 4 dan 10. Grafik MSE vs siklus pelatihan untuk JST dengan ketiga konfigurasi ini ditunjukkan pada Gambar 8.a, 8.b dan 8.c, dengan waktu pelatihan (hingga mencapai siklus ke-3000 dan belum konvergen) 21,5; 22,1 dan 22,9 detik.




Gambar 8. Grafik MSE terhadap siklus pelatihan (epoch) untuk jumlah unit (neuron) di lapis tersembunyi: (a) 1, (b) 4, (c) 10.

Dari hasil eksperimen dapat disimpulkan bahwa semakin banyak jumlah neuron maka semakin cepat MSE turun atau semakin cepat JST belajar dilihat dari banyaknya siklus pelatihan. Namun, ternyata semakin banyak neuron di lapis tersembunyi, waktu eksekusi untuk menuju kekonvergenan tidak semakin mengecil.

### 4.5. Jumlah Lapis Tersembunyi

Pada eksperimen ini, konfigurasi JST yang digunakan mirip dengan eksperimen pada 5.1 dengan jumlah lapis tersembunyi 1 atau 2 dan dengan total jumlah neuron tetap. Ada 5 eksperimen yang dilakukan, yaitu dengan: 1 lapis tersembunyi (lt) dengan 4 neuron, 2 lt dengan 2 neuron di lt-1 dan 2 neuron di lt-2, 1 lt dengan 6 neuron, 2 lt dengan 4 neuron di lt-1 dan 2 neuron di lt-2 dan 2 lt dengan 2 neuron di lt-1 dan 4 neuron di lt-2.

Dari hasil eksperimen (lihat [2]), ternyata tidak dapat ditarik kesimpulan apakah JST dengan 1 lapis tersembunyi lebih baik daripada dengan 2 lapis (dengan jumlah neuron tertentu atau konstan) atau sebaliknya.

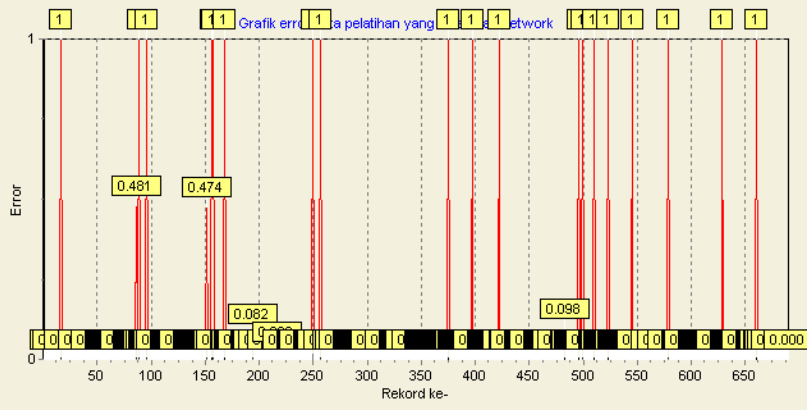
### 4.6. Eksperimen dengan Data Aplikasi Kredit Bank

Pada eksperimen ini, sistem JST akan diuji apakah sudah dapat digunakan untuk mengklasifikasi data nyata. Data nyata untuk pelatihan yang digunakan di eksperimen ini adalah *Australian Credit Database* dari http://www.liacc.up.pt/ML/statlog/datasets/australian/australian.doc.html. Data ini memiliki 690 rekord dan jumlah atribut adalah 14 (bertipe kontinyu dan diskret) ditambah 1 atribut kelas (0 menyatakan kredit disetujui dan 1 ditolak atau sebaliknya). Semua data dijadikan data pelatihan dan 25 % dari data tersebut diambil sebagai data validasi.

Konfigurasi JST yang digunakan adalah: 2 lapis tersembunyi, jumlah neuron di lapis 1 = 20, jumlah neuron lapis 2 = 10, l*earning rate* = 0.125, momentum = 1, jumlah siklus pelatihan maksimum = 3000, epsilon = 0,00001, fungsi aktivasi lapis 1 = sigmoid, fungsi aktivasi lapis 2 2 = arc tangen, fungsi aktivasi lapis keluaran = sigmoid. (Pada

kasus ini, penggunaan fungsi radial basis ternyata tidak mempercepat proses pelatihan JST ini.) Bobot-bobot diinisialisasi dengan nilai-nilai yang dapat mempercepat proses pelatihan.

Dengan konfigurasi di atas, pelatihan JST makan waktu tiga setengah menit. Selesai dilatih, JST lalu divalidasi, kemudian data pelatihan dipresentasikan lagi ke sistem JST untuk dihitung kesalahannya. Nilai kesalahan untuk tiap rekord (dalam rentang [0,1]) diberikan pada Gambar 9, dimana pada gambar ini terlihat bahwa sebagian besar rekord diklasifikasi tanpa kesalahan (dengan kesalahan nol) dan sebagian kecil rekord diklasifikasi dengan salah. Dari perhitungan jumlah rekord yang salah diklasifikasi dibagi dengan total rekord didapat prosentase kesalahan klasifikasi = 7,074 %.




Gambar 9. Grafik nilai kesalahan untuk semua rekord pada data pelatihan.

## 5. Kesimpulan dan Penelitian Lanjutan

JST propagasi balik yang dilatih dengan menggunakan momentum dapat difungsikan sebagai teknik klasifikasi data. Salah satu masalah utama pada JST adalah lamanya proses pelatihan (pembentukan model jaringan), karena itu pemilihan konfigurasi jaringan (jumlah lapis tersembunyi, neuron, nilai momentum, *learning-rate*, fungsi aktivasi) yang tepat diperlukan untuk mempercepat proses pelatihan. Hanya saja, konfigurasi ini dapat berbeda dari satu set data pelatihan yang lain, sehingga diperlukan eksperimen untuk mencarinya.

Eksperimen-eksperimen lanjutan untuk melatih sistem JST dengan berbagai tipe data simulasi dan data nyata, juga untuk membandingkan dengan teknik-teknik klasifikasi data yang lain (misalnya: pohon keputusan, *bayesian belief networks*, dll.) masih diperlukan untuk mengobservasi performansi dari JST ini. Kekurangan atau kelemahan dari JST yang mungkin didapati kemudian dapat diperbaiki dengan mempelajari dan mengimplementasikan algoritma-algoritma pengembangan JST propagasi balik hasil penelitian yang paling mutakhir.

Sistem JST yang dibangun ini masih bersifat *standalone*, padahal banyak data yang perlu diklasifikasi tersimpan di server

basisdata di jaringan. Pengintegrasian JST di server basisdata (sebagai *stored procedures*) yang dapat diakses dari *clients* akan menambah fungsionalitas dari basisdata dan membuat basisdata lebih pintar, sehingga lebih mendatangkan banyak manfaat bagi para pengguna basisdata.

## 6. Referensi

[1] Fausett, Laurent, "*Fundamentals of Neural Networks: Architectures, Algorithms, and Applications"*, Prentice-Hall, Inc., New Jersey, 1994.

[2] Dhaneswara, Giri, "*Implementasi Jaringan Saraf Tiruan Tipe Multilayer Feed-Forward Menggunakan Algoritma Backpropagation dengan Momentum untuk Klasifikasi Dat*a", Skripsi, Jur. Ilmu Komputer, Univ. Katolik Parahyangan, Juli 2004.

[3] Freeman, James A., dan David M. Skapura, "*Neural Networks: Algorithms, Applications, and Programming Techniques"*. Addison-Wesley Publishing Company, 1991.

[4] Borland Software Cp., Borland Delphi v.7 On-line Manual, USA, 2002.