PEMILIHAN ALGORITMA PELATIHAN DENGAN TEKNIK OPTIMASI NUMERIS DALAM PENENTUAN "MAHASISWA TERBAIK" REGULER TEKNIK INFORMATIKA

Lia Praba Kusuma Putri
liapraba@gmail.com
Program Studi Teknik Informatika
Fakultas Teknik, Matematika dan IPA
Universitas Indraprasta PGRI

Abstract. *Usually when making decision for "The Best Student College" in every departement at University, we use in manualy. Sometimes, we miss some student that honour to be "The Best Student College" because of withdrawal or has a nepotism or a lot of number of student. The condtiton or problem can be solve if we built a smart system. Many example of smart system was being applied recently in our life. Neural network is one of them. Neural network has many training algorithm to built a smart system. In this paper, we will discuss about how to choose training algorithm that has the fastest epoch for this case. The right algorithm can be used to built a smart system to make decision which one is "The Best Student College" in Tehnic of Informatics. The end of discussion, we compare every algorithm so we can have the best and fastest algorithm to built a smart system.*

Keywords : Neural Network, numerical optimization, training algorithm,making decision

PENDAHULUAN

Latar Belakang

Pada awal abad ke-21 "*soft computing*" memegang peranan sangat penting, terutama dalam permodelan, identifikasi sistem dan pengendalian sistem mulai dari yang sederhana sampai dengan sistem yang kompleks. Pendekatan sistem dilakukan melalui "*artificial intelegence*" untuk mengembangkan "*soft computing*" agar dapat diaplikasikan dalam berbagai disiplin ilmu. Salah satunya adalah smart system yang merupakan suatu langkah memperbesar kemampuan manusia dalam berpikir dan belajar. Agar suatu sistem dapat mengemulasikan kemampuan otak manusia dalam hal berpikir, berasosiasi, membedakan, dan menginterpretasikan apa yang dialami, sistem harus memahami bagaimana model otak manusia dalam menarik kesimpulan. Smart system dipandang sebagai satu bagian kumpulan suatu sistem peralatan yang mempunyai beberapa tugas. Sistem mempunyai kemampuan menyimpulkan, mengimplementasikan, atau merekomendasikan suatu keputusan berdasrkan kesimpulan yang mengacu pada fakta yang terjadi. Teknologi utama yang mempengaruhi implementasi smart system adalah sistem pakar (expert system), jaringan syaraf tiruan/JST (neural network), dan struktur data.

Sistem pakar juga memasukkan kecerdasan tiruan ke dalam sistem dengan menempatkan pengetahuan sebelumnya ke dalam modul. Modul dapat bereaksi menggunakan stimulan eksternal dengan suatu cara seperti yang diinginkan oleh seorang pakar. Samahalnya, JST adalah suatu sistem yang menginterpretasikan hasil proses komputasi berdasarkan klasifikasi atau karakterisasi yang diintregasikan secara menyeluruh ke dalam suatu pola. Di sisi lain, struktur data berkontribusi pada kecerdasan tiruan dengan cara mengorganisasikan, menyimpan, dan memanggil kembali data yang berhubungan dengan fungsi-fungsi kecerdasan. (Pandjaitan, 2007:5)

Dalam beberapa dekade, ilmu pengetahuan dan teknologi mempunyai tujuan mengembangkan mesin cerdas dengan sejumlah elemen sederhana. Selama tahun 1940, para peneliti yang ingin menduplikasikan otak manusia, telah mengembangkan model neuron biologis dan sistem interaksinya dalam perangkat keras sederhana. Pada tahun 1950 sampai dengan 1960, sekelompok peneliti mengkombinasikan sifat biologis dan psikologis untuk menghasilkan JST pertama kali. (Pandjaitan, 2007:8)

Fungsi-fungsi jaringan yang dikehendaki JST mempunyai 3 elemen dasar, yaitu topologi, proses belajar, pemanggilan kembali. Berdasarkan ketiga elemen dasar, maka arsitektur yang tepat dapat ditentukan. Sebuah pengenalan pola yang paling penting adalah pada saat pembangunan jaringan serta arsitektur jaringan. Ketika seseorang salah menganalisa arsitektur jaringan suatu kasus akan berakibat error besar pada saat mengenali suatu pola. untuk dilakukan untuk menentukan algoritma pelatihan (training) yang paling cepat untuk membangun JST. Kecepatan algoritma tersebut akan membantu kecepatan dan ketepatan sistem cerdas yang dibangun. Sehingga pada saat proses penentuan keputusan tidak berlangsung lama.

Penelitian ini menggunakan JST dengan sifat psikologisnya dalam mengenali sebuah pengenalan pola/karakter suatu kasus. Dalam hal ini kasus yang dianalisa adalah penentuan mahasiswa berprestasi sebagai acuan untuk penghargaan/pengajuan beasiswa. Kendala yang terjadi pada saat penentuan mahasiswa berprestasi adalah jika seseorang yang menentukan mahasiswa tersebut *withdrawal* atau terdapat unsur nepotisme atau terlalu banyak jumlah mahasiswa (program studi Tehnik Informatika), maka akan mempengaruhi penentuan mahasiswa berprestasi. Padahal, akibat kesalahan pengambilan keputusan akan mempengaruhi performa dari kampus/program studi yang bersangkutan.

Berdasarkan permasalahan tersebut, maka penelitian ini diberi judul "Pemilihan Algoritma Pelatihan dengan Tehnik Optimasi Numeris dalam Penentuan "Mahasiswa Terbaik" Reguler Teknik Informatika".

Perumusan Masalah

Perumusan masalah pada penelitian ini adalah :

1. Bagaimanakah arsitektur jaringan dari sistem cerdas penentuan "Mahasiswa Terbaik" Reguler Teknik Informatika ?
2. Bagaimanakah algoritma pelatihan yang terbaik untuk sistem cerdas tersebut ?

Tujuan Penelitian

Tujuan dari penelitian ini adalah :

1. Memilih algoritma pelatihan, khususnya algoritma-algoritma conjugate gradient yang lebih cepat meraih epoch.
2. Menentukan algoritma pelatihan yang akan digunakan untuk pembuatan aplikasi sistem cerdas penentuan mahasiswa terbaik.

TINJAUAN PUSTAKA

JST dapat digunakan untuk beberapa aplikasi, antara lain klasifikasi pola, pengklasteran/kategorisasi, aproksimasi fungsi, prediksi, optimasi, asosiatif memori, dan kendali. Unsur-unsur yang penting pada JST adalah model saraf, proses belajar, dan arsitektur jaringan.

1. Model Saraf

JST berkembang dari model matematika saraf biologi yang dilandasi asumsi-asumsi sebagai berikut (Fausett, 1994) :

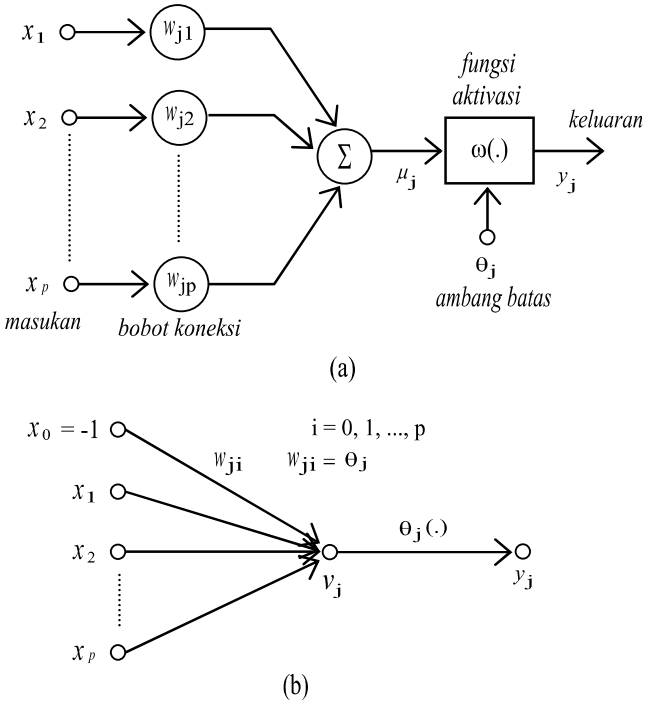
a. Pengolahan informasi terjadi *path* banyak elemen tunggal yang disebut dengan unit (saraf).

b. Sinyal dilewatkan antar unit melalui suatu untai penghubung.

c. Setiap untai penghubung mempunyai bobot hubung.

d. Setiap unit menggunakan suatu fungsi aktivasi untuk menentukan sinyal keluaran.

Berdasarkan asumsi-asumsi tersebut, model saraf digambarkan seperti Gambar 1.



Gambar 1. (a) Model saraf tiruan (b) Dalam bentuk aliran sinyal

Dari Gambar 1a diketahui bahwa keluaran dari penjumlahan linier adalah:

$$u_j = \sum_{i=1}^{p} w_{ji}\ x_i \tag{1}$$

dimana $x_1, x_2, \ldots, x_p$, adalah sinyal-sinyal masukan dan $w_{j1}, w_{j2}, \ldots$, adalah bobot hubung. Tingkat aktivitas adalah :

$$v_j = u_j - \theta_j \tag{2}$$

dimana θ_j adalah ambang batas saraf.
Karena θ_j merupakan parameter luar, maka (1) disubstitusikan ke dalam (2) menjadi:

$$v_j = \sum_{i=1}^{p} w_{ji} x_i \tag{3}$$

dimana $x_0 = -1$ dan $w_{ji} = \theta_j$
Sehingga sinyal keluaran unit j adalah :

$$y_j = \varphi_j (v_j) \tag{4}$$

dimana $\varphi(.)$ adalah fungsi aktivasi.
Fungsi aktivasi $\varphi(.)$ menentukan hasil keluaran unit sesuai dengan tingkat aktivasi pada masukannya.

Ada tiga jenis fungsi aktivasi yaitu :

a. Fungsi *Threshold*

Fungsi aktivasi ini mempunyai persamaan :

$$\mu[x]=\begin{cases}+1, v>0\\ 0\ , v=0\\ -1, v<0\end{cases} \tag{5}$$

dan kurvanya seperti terlihat pada Gambar 2.2a.

b. Fungsi *Piecewise-Linear*

Fungsi *piecewise-linear* ini dapat dilihat pada Gambar 2. 2b dan mempunyai persamaan:

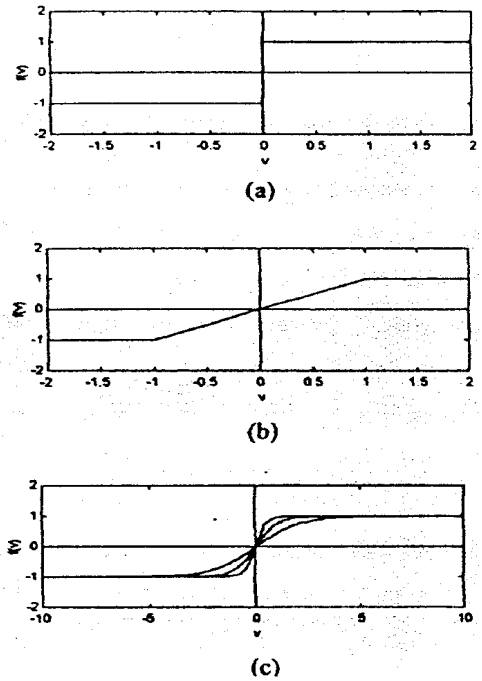
$$\mu[x]=\begin{cases}+1 \quad , v\geq 0\\ 0\ , -1<v<1\\ -1 \quad , v\leq -1\end{cases} \tag{6}$$

c. Fungsi Sigmoid

Fungsi sigmoid adalah fungsi aktivasi yang paling sering digunakan pada JST. Salah satu contoh fungsi sigmoid adalah fungsi logistik yang mempunyai persamaan:

$$\mu\varphi[v]=\frac{1-\exp(-av)}{1+\exp(-av)} \tag{7}$$

dimana *a* adalah parameter lekukan *(slope).* Kurva fungsi tersebut dapat dilihat pada Gambar 2.2c untuk *a* = 1, 2, 4.



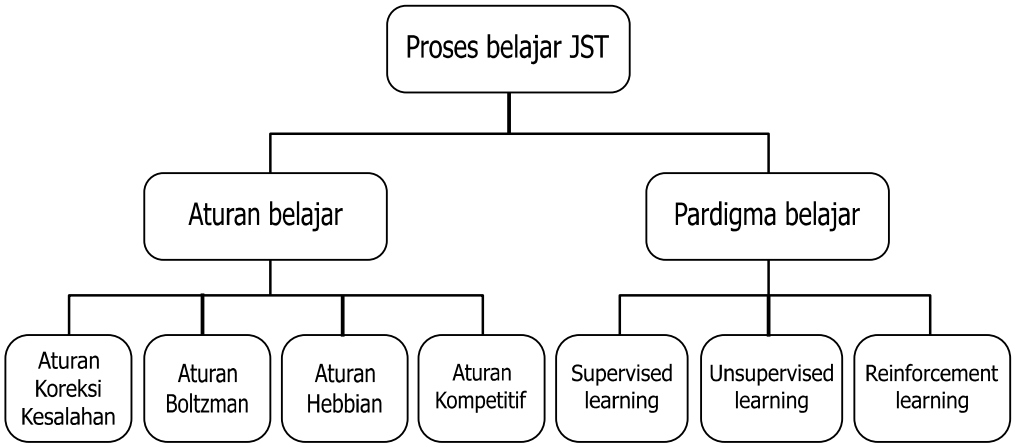
Gambar 2.2. (a) Fungsi *threshold* (b) Fungsi *piecewise-linear* (c) Fungsi sigmoid

2. Proses Belajar JST

Kemampuan belajar merupakan sifat utama yang penting dari JST. Definisi belajar dalam konteks JST adalah suatu proses dimana parameter bebas JST berubah melalui proses stimulasi berkesinambungan oleh lingkungan dimana jaringan ditanamkan. Jenis belajar ditentukan oleh bagaimana cara perubahan parameter bebas tersebut (Haykin, 1994). Ada dua hal yang penting pada proses belajar tersebut Pertama, apakah lingkungan dimodelkan terlebih dahulu atau tidak untuk dijadikan target pada proses belajar. Hal ini disebut dengan paradigma belajar. Yang kedua adalah bagaimana cara mengubah parameter (bobot hubung) tersebut, dikenal dengan aturan atau algoritma belajar.

Paradigma belajar dapat diklasifikasikan atas *supervised learning, reinforcement learning,* dan *unsupervised learning.* Pada paradigma belajar *supervised learning,* nilai bobot hubung diubah berdasarkan selisih antara target dan nilai aktual. Paradigma belajar *unsupervised learning* tidak memerlukan target, jaringan hanya diberi urutan masukan saja. Jaringan memeriksa struktur dasar data masukan atau korelasi antar pola data masukan dan mengorganisasikannya ke dalam katagori-katagori berdasarkan korelasinya. Paradigma belajar *reinforcement learning* merupakan kasus khusus dari belajar *supervised learning.* Pada paradigma belajar ini, target digantikan dengan suatu kriteria atau kritikan untuk mengevaluasi apakah nilai keluaran aktual sudah dapat diterima atau belum.

Ada empat jenis dasar aturan belajar, yaitu aturan Koreksi Kesalahan, aturan Boltzman, aturan Hebbian, dan aturan Kompetitif. Paradigma dan aturan belajar tersebut dapat digambarkan dalam taksonomi berikut :



Gambar 3. Taksonomi proses belajar JST

3. Arsitektur JST

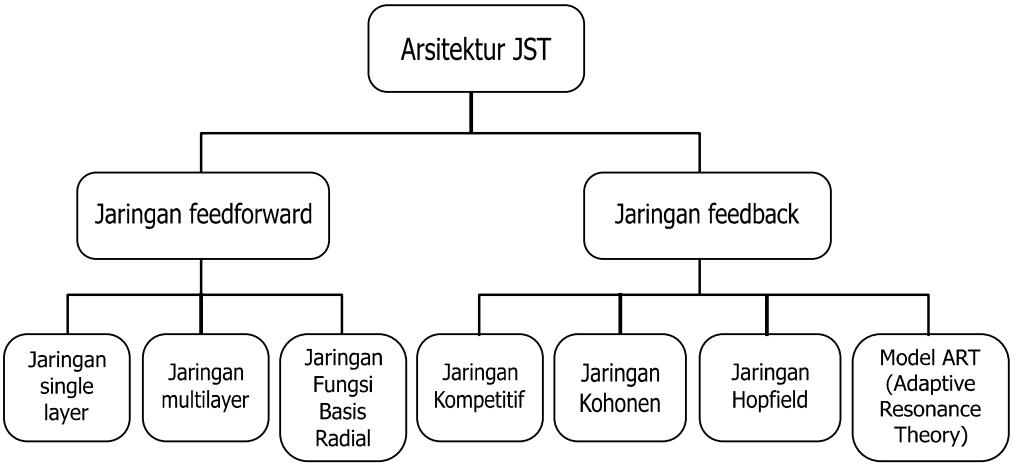
JST dapat dipandang sebagai garis-garis arah aliran sinyal yang menghubungkan unit masukan dengan unit keluaran. Arsitektur JST didasarkan atas konfigurasi hubungan antar unit tersebut. Berkaitan dengan arah aliran sinyal, secara garis besar ada dua jenis arsitektur yaitu: jaringan *feedforward* dan jaringan *feedback.*

Jaringan *feedforward* adalah jaringan yang arah sinyalnya dalam arah maju saja dan tidak ada ikal. Secara umum, jaringan *feedforward* bersifat statik dimana jaringan hanya menghasilkan satu set keluaran untuk setiap urutan nilai masukan. Dengan demikian jaringan *feedforward* bersifat *memoryless,* dalam

pengertian tanggapan terhadap suatu masukan tidak tergantung dan keluaran sebelumnya.

Jaringan *feedback* adalah jaringan yang arah sinyalnya dalam arah maju dan mundur yang berasal dan hubungan umpanbalik. Pada jaringan ini, paling tidak memiliki satu ikal. Dengan lintasan umpanbalik, hasil keluaran saraf dipengaruhi oleh urutan nilai masukan dan keluaran itu sendiri. Jadi arsitektur jaringan *feedback* bersifat dinamis. Arsitektur jaringan *feedback* ini dimaksudkan untuk memperoleh keluaran yang stabil.

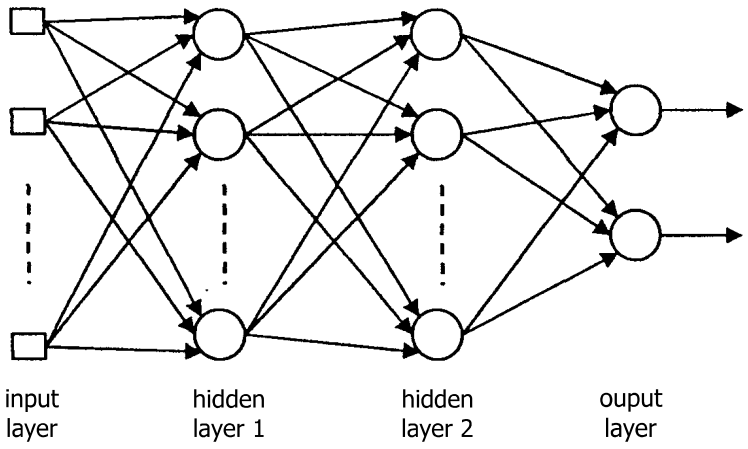
Arsitektur yang berbeda akan menghasilkan perilaku jaringan yang berbeda pula dan masing-masing arsitektur membutuhkan algoritma belajar yang sesuai. Arsitektur JST dapat ditaksonomikan seperti terlihat pada Gambar 2.4



Gambar 2.4. Taksonomi arsitektur JST

Jaringan Backpropagation

Salah satu arsitektur jaringan *feedforward* adalah jaringan *multilayer.* Secara umum, jaringan *feedforward L layer (multilayer)* terdiri dari satu set unit sensor yang merupakan *input layers,* satu atau lebih *(L*-1*) hidden layer,* dan satu *output layer,* seperti terlihat pada Gambar 5.



Gambar 5. Arsitektur jaringan *multilayer L* = 3

Proses belajar jaringan *multilayer* menggunakan paradigma belajar *supervised learning* dan algoritma belajar *backpropagation* yang didasari atas aturan koreksi kesalahan. Jaringan *multilayer* dengan algoritma belajar *backpropagation* ini disebut dengan jaringan *Backpropagation*.

Proses *backpropagation* terdiri dan dua lintasan sinyal, yaitu: lintasan maju dan lintasan balik. Pada lintasan maju, vektor masukan *x(n)* dimasukkan pada set unit sensor *(input layer),* kemudian dipropagasikan melalui *layer* demi *layer* dalam jaringan, sehingga didapat vektor sinyal keluaran aktual *y*(*n*) pada *output layer.* Selama lintasan maju, bobot hubung antar unit bernilai tetap. Sinyal keluaran aktual *y*(*n*) dikurangkan dengan target *d*(*n*), sehingga dihasilkan sinyal kesalahan *e*(*n*). Kemudian sinyal kesalahan *e*(*n*) dipropagasikan balik melalui jaringan. Selama lintasan balik ini bobot hubung disesuaikan dengan aturan koreksi kesalahan.

Algoritma Backpropagation

Algoritma baku *backropagation* adalah sebagai berikut:

1) Inisialisasi bobot hubung dengan bilangan acak kecil berdistribusi *uniform*.

2) Tentukan pola belajar [*x*(*n*) , *d*(*n*)] untuk iterasi ke-*n*.

3) Propagasikan sinyal dalam arah maju,

(a) Tingkatkan aktivitas unit *j* path *layer* 1 adalah:

$$v_j^{(j)}(n) = \sum_{i=0}^{p} w_{ji}^{(t)}(n) y_i^{(t-1)}(n)$$

Untuk $i = 0$: $y_0^{(t-1)}(n) = -1$; $w^1_{j0}(n) = \theta_j^{(t)}(n)$

Untuk $i = 1$: $y_i^{(t)}(n) = x_i(n)$

(b) Sinyal keluaran unit *j* pada *layer* 1 adalah :

$y_j^{(t)}(n) = \varphi_j\{v_j^{(t)}(n)\}$

Untuk $i = L$: $y_j^{(L)}(n) = o_j(n)$

(c) Sinyal kesalahan adalah :

$e_j(n) = d_j(n) - o_j(n)$

4) Propagasikan sinyal dalam arah balik :

(a) Gradien lokal pada *output layer* (i = L):

$\delta_j^{(L)}(n) = e_j(n)\,\varphi'_j\{v_j^{(L)}(n)\}$

(b) Gradien lokal pada *hidden layer* ($1 \le l \le L$):

$$\delta_j^{(t)}(n) = \varphi'_j\left\{v_j^{(L)}(n)\right\}\sum_{k=C}\delta_k^{(l+1)}(n)\delta_{kj}^{(l+1)}(n)$$

5) Sesuaikan bobot hubung dengan:

$\Delta w_{ji}^{(t)}(n) = \eta\delta_j^{(t)}(n)\, y_i^{(l-1)}(n)$

6) Ulangi langkah 2 sampai

$\xi_{av} = \frac{1}{N}\sum_{i=0}^{N}\xi(n)$ sekecil mungkin,

dimana :

$$\xi(n) = \frac{1}{2}\sum_{j=k} e_j^2(n)$$

Dari algoritma tersebut terlihat bahwa pelatihan jaringan dengan algoritma *backpropagation* melibatkan tiga tingkatan, yaitu umpan maju pola belajar masukan, kalkulasi dan propagasi balik sinyal kesalahan yang bersesuaian, dan penyesuaian bobot hubung. Setelah pelatihan, penggunaan jaringan hanya melibatkan perhitungan arah maju saja

Satu siklus seluruh pola belajar (satu set) yang diselesaikan pada pelatihan disebut dengan satu *epoch.* Lamanya proses belajar ditentukan oleh jumlah *epoch* yang dibutuhkan sampai didapat nilai rata-rata kuadrat kesalahan sekecil mungkin.

Untuk mempercepat proses pelatihan, algoritma pelatihan diperbaiki dengan menggunakan teknik optimasi numeris. Pada MATLAB, algoritma pelatihan untuk kedua metode disediakan dalam batch mode. Algoritma yang dibahas dan

digunakan dalam penelitian ini adalah algoritma Conjugate Gradient. Pada algoritma tersebut, pengaturan bobot disesuaikan dengan arah konjugasinya.

Misal bobot baru (bobot input, bias input, bobot lapisan dan bias lapisan) tergabung dalam sebuah vektor yang bernama W. Nilai vektor W pada epoh ke-k diperoleh dari :

$$W_k = W_{k-1} + a * dW_k$$

Dimana dW adalah arah yang akan dicari. Parameter a dicari untuk meminimumkan kinerja selama arah pencarian. Fungsi line search (search Fcn) digunakan untuk menempatkan titik minimum (a) tersebut. Ada beberapa line search yang dapat digunakan, antara lain :

- Golden Section Search (srchgol)
  Golden section search merupakan pencarian linear sederhana tanpa harus menghitung gradien dari garis tersebut. Misalkan fungsi kinerja untuk suatu iterasi adalah fx. Proses pencarian nilai minimum dimulai dari menempatkan suatu interval (delta), antara a (= 0) dan b (= delta). Kemudian dicari fungsi kinerja untuk a (fa) dan b (fb). Apabila fb < fa, maka pelebaran interval dilakukan lagi sebesar delta = 2*delta. Iterasi tersebut dilakukan hingga b ≥ bmax, atau fb ≥ fa . Tetapi, jika fb ≥ fa, berarti nilai minimum terdapat pada interval [a b] tersebut. Letakkan 2 nilai pada interval [a b], misalkan c dan d. Hitung fungsi kinerja c sebagai fc, dan d sebagai fd. Kedua nilai ini akan menentukan daerah mana yang dapat dibuang, dan suatu titik yang terletak di dalam interval baru. Iterasikan proses tersebut hingga lebar interval kurang dari suatu toleransi minimum.
- Brent's Search (**srchbre**)
  Brent's search merupakan pencarian linear yang menggunakan kombinasi antara golden section search dan interpolasi kuadratik. Proses pencarian pada awalnya dilakukan dengan cara yang sama dengan golden section search. Kemudian interpolasi kuadratik digunakan untuk mencocokkan nilai fungsi dari setiap titik yang diperoleh, lalu dilanjutkan dengan mencari nilai minimumnya.
- Hybrid Bisection-Cubic Search (**srchhyb**)
  Hybrid bisection-cubic merupakan pencarian linear yang menggunakan kombinasi antara metode biseksi dan interpolasi kubik. Pencarian dengan metode biseksi diawali dengan menentukan suatu interval. Sebuah titik diletakkan di dalam interval tersebut, kemudian dihitung turunannya. Kemudian interval dibagi menjadi 2. Dari nilai turunan titik tadi, bisa diketahui bagian interval yang dapat dibuang. Interpolasi kubik digunakan untuk mencocokkan nilai fungsi dan turunannya pada kedua titik batas interval.
- Charalambous' Search (**srchcha**)
  Charalambous' search merupakan metode pencarian yang menggunakan kombinasi antara interpolasi kubik dan suatu tipe sectioning. Metode ini dipergunakan sebagai default dari algoritma pelatihan dengan conjugate gradient.

Arah pencarian pertama (dW) awal, masih berupa negatif gradient dari kinerja. Arah pencarian selanjutnya dihitung dari gradien baru dan arah pencarian sebelumnya, dengan rumus:

$$dW_k = -gW_k + dW_{k-1} * Z$$

Dengan gW adalah vektor yang berisi (φ1 , β1 , φ2 , β2). Parameter Z dapat dihitung dengan beberapa cara, yaitu :

1. Fletcher-Reeves Update (**traincgf**)
   Pada metode ini, nilai Z ditentukan dengan rumus :
   $$Z = \frac{gW_k' * gW_k}{gW_{k-1}' * gW_{k-1}}$$
2. Polak-Ribie're (**traincgp**)
   Pada metode ini, nilai Z ditentukan dengan rumus :

$$Z = \frac{(gW_k - gW_{k-1})'*gW_k}{gW_{k-1}'*gW_{k-1}}$$

3. Powell-Beale Restarts (**traincgb**)
   Pada semua algoritma dengan cunjugate gradient, arah pencarian akan direset ke negatif gradient secara periodis. Reset ini umumnya dilakukan pada saat jumlah iterasi sama dengan jumlah parameter jaringan (bobot input, bobot lapisan, dan bobot bias). Pada metode ini, reset arah gradien ini dilakukan pada saat :

$$\left|g_{k-1}'*g_k\right| \geq 0.2\left\|g_k\right\|^2$$

4. Scaled Conjugate Gradient (**trainscg**)
   Hampir semua algoritma yang menggunakan *conjugate gradient* akan melakukan proses line search secara terus-menerus selama iterasi. Untuk jumlah data yang besar dan iterasi yang besar pula proses ini akan memakan waktu yang cukup lama. Algoritma *scaled conjugate* gradient mencoba untuk memperbaiki hal tersebut.

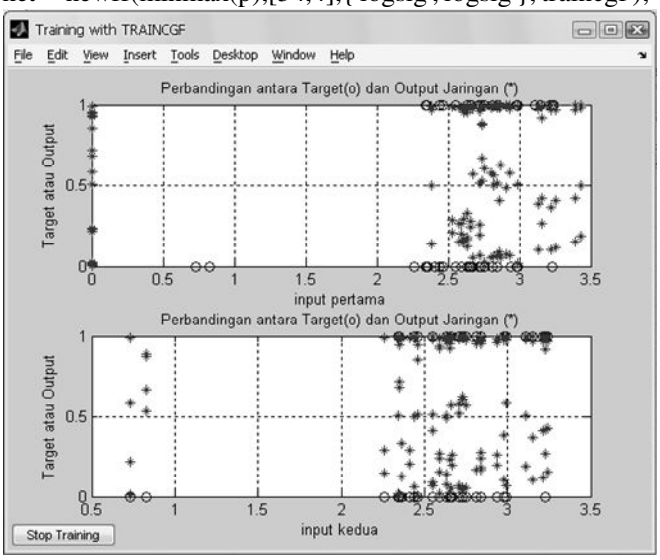
METODOLOGI PENELITIAN

Metode penelitian yang digunakan melalui pendekatan ***sistem*** dan penelitian studi literatur. Data yang digunakan sebagai sampel adalah mahasiswa reguler semester 4 kelas F program studi Teknik Informatika Universitas Indraprasta PGRI sebanyak 32 mahasiswa. Dengan jumlah Indeks Prestasi semester 1, 2, dan 3 serta jumlah sks yang ditempuh selama 3 semester.

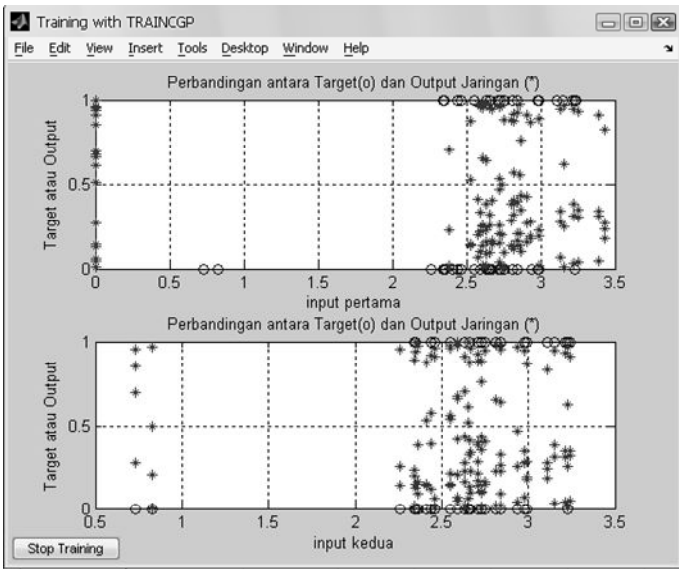
HASIL DAN PEMBAHASAN

MATLAB 7.0 digunakan sebagai alat untuk mendeteksi algoritma yang cepat meraih epoch dengan tehnik optimasi numeris. Arsitektur jaringan yang dibangun adalah 32 data sebagai input dan 4 buah kriteria sebagai output.
Berikut adalah algoritma pelatihan serta hasilnya :

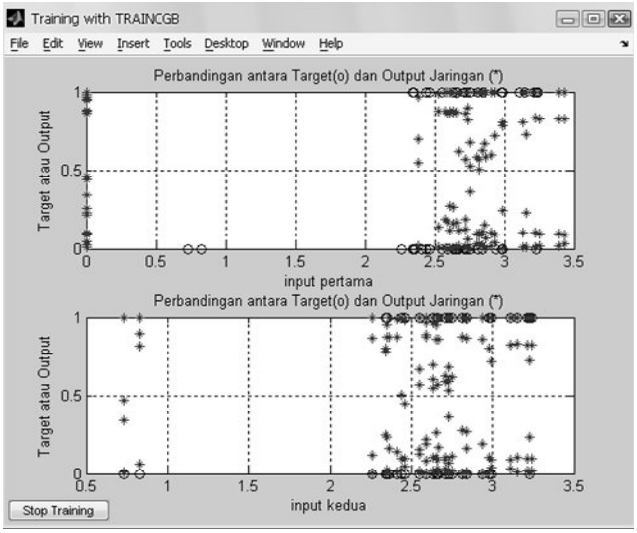
1. Fletcher-Reeves Update (**traincgf**)
   Jaringan pelatihan dibangun dengan fungsi aktivasi log sigmoid :
   net = newff(minmax(p),[34,4],{'logsig','logsig'},'traincgf');



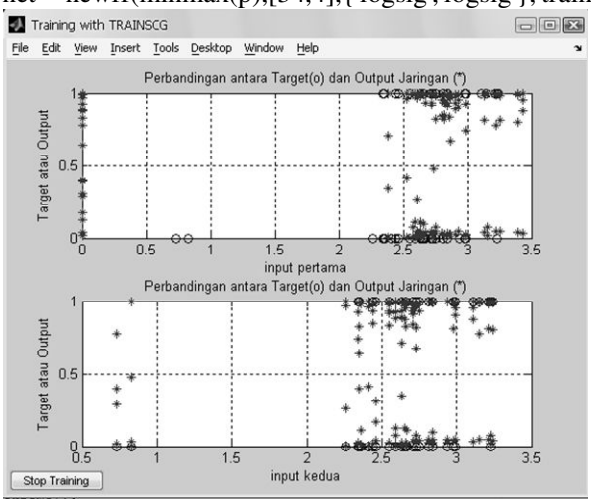
2. Polak-Ribie're (**traincgp**)
   Jaringan pelatihan dibangun dengan fungsi aktivasi log sigmoid :
   net = newff(minmax(p),[34,4],{'logsig','logsig'},'traincgp');



3. Powell-Beale Restarts (**traincgb**)
   Jaringan pelatihan dibangun dengan fungsi aktivasi log sigmoid :
   net = newff(minmax(p),[34,4],{'logsig','logsig'},'traincgb');



4. Scaled Conjugate Gradient (**trainscg**)
   Jaringan pelatihan dibangun dengan fungsi aktivasi log sigmoid :
   net = newff(minmax(p),[34,4],{'logsig','logsig'},'trainscg');



Dilihat juga pada tabel berikut untuk hasil algoritma pelatihannya :

Algoritma Pelatihan	Epoch ke- (Maks 1000)	MSE (Min 0,001)	Gradien (1e-0,006)
TRAINCGF	931	0.000977109	0.00587487
TRAINCGP	546	0.000979846	0.0173201
TRAINCGB	364	0.0009839	0.0146861
TRAINSCG	454	0.000998122	0.00614872

PENUTUP

Kesimpulan

Berdasarkan gambar dan tabel pada hasil, semua algoritma berhasil mencapai *performa goal*-nya. Algoritma yang mencapai epoch lebih cepat adalah *Powell-Beale Restarts* dengan mean squared error yang masih dapat ditolerir. Sehingga, algoritma pelatihan *Powell-Beale Restarts* (traincgb) dapat digunakan untuk rancangan sistem cerdas berikutnya.

Saran

Hasil dari penelitian ini dapat dilanjutkan untuk perancangan pembuatan sistem cerdas "Prediksi Mahasiswa Terbaik Program Studi Tehnik Informatika. Algoritma pelatihan tersebut digunakan sebagai acuan untuk pengujian data yang akan diinput selanjutnya.

DAFTAR PUSTAKA

Away, Gunaidi A. 2006. **The Shortcut of MATLAB Programming**. Bandung : Penerbit Informatika.

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

Kusumadewi, Sri. 2004. **Membangun Jaringan Syaraf Tiruan (Menggunakan MATLAB & Excel Link)**. Yogyakarta : Graha Ilmu.

Pandjaitan, Lanny W. 2007. **Dasar-dasar Komputasi Cerdas**. Yogyakarta : ANDI.

Puspitaningrum, Dyah. 2006. **Pengantar Jaringan Syaraf Tiruan**. Yogyakarta: ANDI.

Santosa, Budi. 2007. **Data Mining: Teknik Pemanfaatan Data untuk Keperluan Bisnis**. Yogyakarta:Graha Ilmu.

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

Suteja, B.R. **Penerapan Jaringan Saraf Tiruan Propagasi Balik Studi Kasus Pengenalan Jenis Kopi**. Jurnal Teknik Informatika, Vol.3, No.1, Juni 2007:49-62. Fakultas Teknologi Informasi, Jurusan S1 Teknik Informatika, Universitas Kristen Maranatha.

Suyanto. 2007. **Artificial Intelligence**. Bandung: Penerbit Informatika.