ISSN 2085-4552

Penerapan Jaringan Saraf Tiruan Metode Backpropagation Menggunakan VB 6

Sari Indah Anatta Setiawan
SofTech, Tangerang, Indonesia
cu.softech@gmail.com

Diterima 30 November 2011
Disetujui 14 Desember 2011

***Abstrak*—Jaringan saraf tiruan merupakan salah satu metode soft computing yang banyak digunakan dewasa ini. Berbagai macam metode pembelajaran juga telah dikembangkan oleh para peneliti untuk jaringan saraf tiruan dengan berbagai maksud dan tujuan. Pada penelitian ini, penulis mencoba untuk menerapkan salah satu metode pembelajaran jaringan saraf tiruan yang terawasi, yakni backpropagation, dengan memanfaatkan perangkat lunak Visual Basic 6.**

***Kata kunci*—jaringan saraf tiruan, backpropagation, VB 6**

I. PENDAHULUAN

Jaringan saraf tiruan (*neural network*) telah banyak dikembangkan oleh para peneliti dewasa ini. Beragam metode pembelajaran untuk jaringan saraf tiruan juga terus dikembangkan, seperti *delta learning rule*, *kohonen self-organizing map*, dan *back-propagation*.

Backpropagation adalah salah satu metode pembelajaran terawasi yang banyak digunakan oleh para peneliti dalam membangun suatu sistem. Metode ini umumnya digunakan pada jaringan *multi-layer* dengan tujuan untuk meminimalkan *error* pada *output* yang dihasilkan oleh jaringan selama pelatihan. Penulis mencoba untuk menerapkan jaringan saraf tiruan metode *backpropagation* ini dengan memanfaatkan Visual Basic 6 sebagai lingkungan pengembangan programnya.

II. JARINGAN SARAF TIRUAN

Jaringan saraf tiruan (JST) (*artificial neural network* (ANN)/ *simulated neural network* (SNN)/ *neural network* (NN)) adalah jaringan yang terdiri atas sekelompok unit pemroses kecil yang dimodelkan berdasarkan jaringan saraf manusia.

Adapun alasan-alasan yang mendasari banyak peneliti mempelajari jaringan saraf tiruan adalah sebagai berikut [1].

- Kemampuan untuk menyelesaikan data yang kompleks atau yang tidak tepat.
- Kemampuan untuk menemukan pola (*pattern*) dan *trend* yang terlalu kompleks untuk dikenali oleh manusia atau teknik komputasi lainnya.

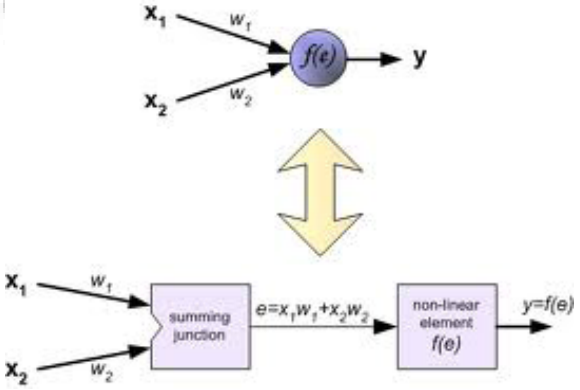
Terdapat beberapa kelebihan dari penerapan jaringan saraf tiruan, antara lain:

- Dapat menyelesaikan tugas yang tidak dapat dikerjakan oleh program linear.
- Saat sebuah elemen dari JST gagal, jaringan dapat terus berjalan tanpa masalah.
- Sebuah JST dapat belajar (dilatih) dan karenanya tidak perlu di-program ulang.
- Dapat diterapkan pada sembarang aplikasi.
- Dapat diterapkan tanpa banyak kendala.

Namun, selain beberapa kelebihan yang telah disebutkan sebelumnya, penerapan jaringan saraf tiruan juga memiliki beberapa kekurangan, seperti berikut:

- JST memerlukan pelatihan (*training*) agar dapat beroperasi.
- Memerlukan waktu proses yang lama untuk jaringan yang besar.

Unsur-unsur dari suatu jaringan saraf tiruan dapat digambarkan seperti gambar di bawah.



Gambar 1. Model jaringan saraf tiruan [2]

- Unit/ Sel/ Node

Tiga komponen utama: unit-unit *input*, unit-unit *hidden*, unit-unit *output*.

- Bobot

Informasi yang digunakan oleh jaringan untuk menyelesaikan persoalan. Setiap unit *input* dan unit *hidden* memiliki bobotnya masing-masing.

- Bias

Suatu bobot penghubung dari unit khusus dengan nilai aktivasi konstan tidak nol. Istilah 'bias' biasanya digunakan untuk merujuk pada 'unit bias' dengan nilai konstan 1. Sementara istilah '*threshold*' biasanya digunakan untuk merujuk pada 'unit *threshold*' dengan nilai konstan -1.

- Fungsi aktivasi

Fungsi aktivasi *linear* dan fungsi aktivasi *nonlinear*. Jaringan *multi-layer* umumnya menggunakan fungsi aktivasi *nonlinear*.

- Fungsi Sigmoid Biner

$$f(x) = \frac{1}{1+\exp(-\sigma x)}$$

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

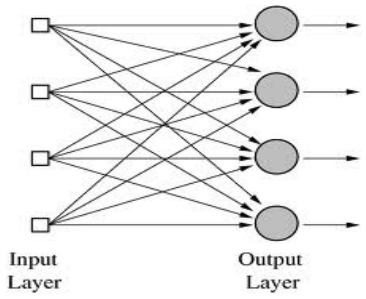
dimana σ adalah parameter kecuraman fungsi.

Jika tidak diketahui dalam soal, σ=1.

Fungsi *Sigmoid Biner* berbentuk kurva S. Fungsi ini umum digunakan dalam jaringan yang menggunakan metode pelatihan *Backpropagation* karena bentuk fungsi dan turunan fungsinya sederhana, sehingga mudah dihitung. *Range* dari fungsi ini berada antara 0 dan 1.

Secara umum, arsitektur jaringan saraf tiruan dibedakan menjadi dua, yakni jaringan saraf *single-layer* dan jaringan saraf *multi-layer*.

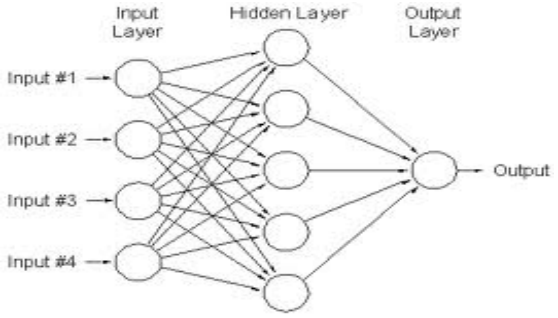
Jaringan saraf single-layer



Gambar 2. Arsitektur JST *single-layer* [3]

- *Neuron-neuron* dikelompokkan menjadi dua bagian, yakni unit-unit *input* dan unit-unit *output*.
- Unit-unit *input* menerima masukan dari luar, unit-unit *output* mengeluarkan respon dari jaringan sesuai dengan masukannya.

Jaringan saraf multi-layer



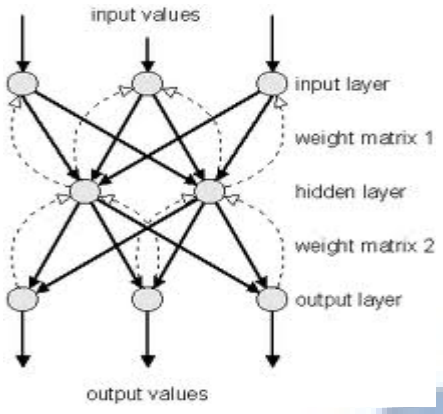
Gambar 3. Arsitektur JST *multi-layer* [4]

- *Neuron-neuron* dikelompokkan menjadi tiga bagian, yakni unit-unit *input*, unit-unit *hidden*, dan unit-unit *output*.
- Jumlah unit-unit *hidden* tergantung pada kebutuhan. Semakin kompleks jaringan, unit-unit *hidden* yang dibutuhkan semakin banyak, demikian pula jumlah *layer*-nya.

Dalam proses pelatihan jaringan saraf tiruan, dikenal proses belajar terawasi dan proses belajar tak terawasi.

Proses belajar terawasi (supervised learning)

- Seolah ada 'guru' yang mengawasi jaringan.
- Cara pelatihan dengan memberikan data-data pelatihan (*training data*) yang terdiri atas pasangan *input-output* yang diharapkan.
- Data-data ini biasanya diperoleh dari pengalaman atau pengetahuan seseorang dalam menyelesaikan suatu persoalan.
- Setelah jaringan dilatih menggunakan data *training* yang ada, jaringan akan mengingat suatu pola (*pattern*).
- Jika diberikan suatu *input* baru, jaringan dapat mengeluarkan *output* seperti yang diharapkan berdasarkan pola yang sudah ada.
- Beberapa metode dengan proses belajar terawasi, antara lain: *Delta Rule*, *Generalized Delta Rule*, *Backpropagation*.



Gambar 4. Arsitektur JST *backpropagation* [5]

Proses belajar tak terawasi (unsupervised learning)

- Tidak ada 'guru' yang mengawasi jaringan.
- Jaringan hanya diberi data *input* (*input vectors*), namun tidak ada *target* (*output vectors*).
- Jaringan akan memodifikasi bobot, sehingga untuk *input* yang hampir sama, *output* yang dihasilkan sama (*cluster units*).
- Beberapa metode dengan proses belajar tak terawasi, antara lain: *Kohonen Self-organizing Maps*, *Counterpropagation*.

III. BACKPROPAGATION

Seperti yang telah dijelaskan pada subbab sebelumnya, *backpropagation* merupakan salah satu metode jaringan saraf tiruan dengan proses belajar terawasi. Dalam penelitian ini, penulis mencoba untuk menerapkan metode *backpropagation* dengan menggunakan perangkat lunak VB 6. Adapun algoritma pelatihan *backpropagation* seperti yang dijelaskan di bawah.

Langkah ke-0 : Inisialisasi bobot dan bias.

Bobot dan bias dapat di-inisialisasi dengan sembarang angka (acak) dan biasanya terletak antara 0 dan 1 atau -1.

Langkah ke-1 : Jika kondisi STOP belum terpenuhi, lakukan langkah 2 – 9.

Langkah ke-2 : Untuk setiap data *training*, lakukan langkah 3 – 8.

Umpan Maju (Feedforward)

Langkah ke-3 : Setiap unit *input* ($X_i, i = 1, \dots, n$) menerima sinyal *input* x_i dan menyebarkan sinyal tersebut ke seluruh unit pada *hidden units*.

Catatan: sinyal *input* x_i yang digunakan adalah input data *training* yang sudah diskalakan. Pertama, dicari nilai terendah dan tertinggi dari *input* data *training*. Kemudian, skala yang digunakan tergantung pada fungsi aktivasinya. Jika yang digunakan adalah fungsi *Sigmoid Biner* yang mempunyai nilai terendah 0 dan tertinggi 1, maka nilai *input* terendah juga dianggap 0 dan tertinggi dianggap 1. Bila fungsi aktivasi yang digunakan adalah *Sigmoid Bipolar*, maka *range* nilainya juga bervariasi mulai dari -1 sampai dengan 1.

Langkah ke-4 : Pada setiap *hidden unit* ($Z_j, j = 1, \dots, p$), jumlahkan sinyal-sinyal *input* yang sudah berbobot (termasuk biasnya)

$$z\_in_j = v_{0j} + \sum_{}^{n} x_i v_{ij}$$

Lalu menghitung sinyal *output* dari *hidden unit* bersangkutan dengan menggunakan fungsi aktivasi yang telah ditentukan

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

Sinyal *output* ini selanjutnya dikirim ke seluruh unit pada unit *output*.

Langkah ke-5 : Pada setiap unit *output* ($Y_k, k = 1, \dots, m$), jumlahkan sinyal-sinyal *input* yang sudah berbobot (termasuk biasnya)

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

Lalu menghitung sinyal *output* dari unit *output* bersangkutan dengan menggunakan fungsi aktivasi yang telah ditentukan

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

Sinyal *output* ini selanjutnya dikirim ke seluruh unit pada *output*.

Umpan Mundur/Propagasi Error (Backpropagation of Error)

Langkah ke-6 : Setiap unit *output* ($Y_k, k = 1, \dots, m$) menerima suatu target *pattern* (*desired output*) yang sesuai dengan *input training pattern* untuk menghitung kesalahan (*error*) antara *target* dengan *output* yang dihasilkan jaringan

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

Faktor δ_k digunakan untuk menghitung koreksi

error (Δw_{jk}) yang nantinya akan dipakai untuk meng-*update* w_{jk}, dimana

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

Selain itu juga dihitung koreksi bias Δw_{0k} yang nantinya akan dipakai untuk meng-*update* w_{0k}, dimana

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

Faktor δ_k kemudian dikirimkan ke *layer* yang berada pada langkah ke-7.

Langkah ke-7 : Setiap *hidden unit* ($Z_j, j = 1, \dots, p$) menerima *input delta* (dari langkah ke-6) yang sudah berbobot

$$\delta\_in_j = \sum^m \delta_k w_{jk}$$

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi *error* δ_j, dimana

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

Faktor δ_j digunakan untuk menghitung koreksi *error* (Δv_{ij}) yang nantinya akan dipakai untuk meng-*update* v_{ij}, dimana

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

Selain itu juga dihitung koreksi bias Δv_{0j} yang nantinya akan dipakai untuk meng-*update* v_{0j}, dimana

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

Update Bobot dan Bias (Adjustment)

Langkah ke-8 : Setiap unit *output* ($Y_k, k = 1, \dots, m$) akan memperbaharui bias dan bobot dari setiap *hidden* unit ($j = 0, \dots, p$)

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

Demikian pula untuk setiap *hidden unit* ($Z_j, j = 1, \dots, p$) akan memperbaharui bias dan bobot dari setiap unit *input* ($i = 0, \dots, n$)

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

Langkah ke-9 : Memeriksa kondisi STOP.

Jika kondisi STOP telah terpenuhi, maka pelatihan jaringan dapat dihentikan.

Untuk memeriksa kondisi STOP, biasanya digunakan kriteria MSE (*Mean Square Error*) berikut

$$MSE = 0.5 \times \{(t_{k1} - y_{k1})^2 + (t_{k2} - y_{k2})^2 + \cdots + (t_{km} - y_{km})^2\}$$

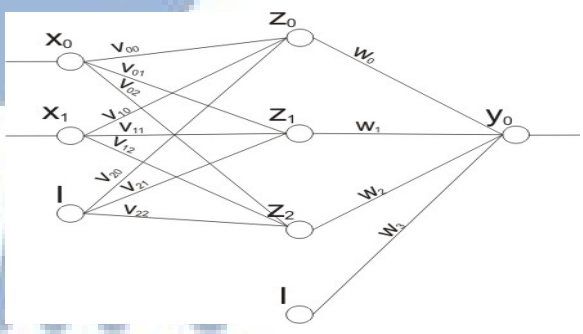
Testing

Setelah pelatihan selesai, jika jaringan diberi *input*, maka jaringan akan dapat menghasilkan *output* seperti yang diharapkan. Caranya adalah dengan menerapkan metode *Backpropagation* di atas, namun hanya pada bagian umpan maju (langkah ke-3 hingga langkah ke-5).

Catatan: Variabel y_k adalah *output* yang masih berada dalam skala menurut *range* fungsi aktivasi. Untuk mendapatkan nilai *output* sesungguhnya, y_k harus diskalakan kembali seperti semula.

IV. PENERAPAN DENGAN VISUAL BASIC 6

Setelah mengetahui algoritma jaringan saraf tiruan metode *backpropagation*, penulis mencoba menerapkannya dengan menggunakan Visual Basic 6. Untuk penelitian ini, akan disimulasikan sistem jaringan saraf tiruan yang memiliki dua piranti masukan (x_0 dan x_1) dan satu piranti keluaran (y). Arsitektur jaringan yang akan dibangun seperti yang terlihat di bawah.



Gambar 5. Arsitektur jaringan yang dibangun

Pada sistem terdapat satu *hidden layer* yang terdiri atas tiga unit Z (yakni z_0, z_1, dan z_2). *Node-node* bias dilambangkan dengan I dan memiliki bobot masing-masing v_{20}, v_{21}, v_{22} untuk bobot bias pada unit tersembunyi Z, serta w_3 untuk bobot bias pada piranti keluaran Y. v_{00}, v_{01}, v_{02}, v_{10}, v_{11}, v_{12} merupakan bobot garis dari piranti masukan X ke unit tersembunyi Z. w_0, w_1, w_2 merupakan bobot garis dari unit tersembunyi Z ke piranti keluaran Y. Fungsi aktivasi yang digunakan adalah fungsi *Sigmoid biner* dengan *range* 0 sampai 1.

Sistem yang dibangun dapat menerima maksimal 20 data *training*, dan dapat men-*generate* maksimal 20 data *testing*. Misal diberikan 20 data *training*, sebagai berikut:

Tabel 1. Data *training*

No	Input		Output
	X_0	X_1	Y
1	0.5	0.6	1
2	0.75	0.5	1
3	0.1	0.4	0
4	0.25	0.75	0
5	0.55	0.77	1
6	0.05	0.25	0
7	0.9	0.2	1
8	0.7	0.6	1
9	0.4	0.25	0
10	0.01	0.9	0

No	Input		Output
	X_0	X_1	Y
11	0.2	0.7	0
12	0.9	0.1	1
13	0.7	0.7	1
14	0.4	0.65	1
15	0.2	1	1
16	1	0	1
17	0	0.95	0
18	0.8	0.6	1
19	0.3	0.7	0
20	0.2	0.1	0

Akan diprediksi *output* (piranti keluaran) dari 10 data *testing* berikut:

Tabel 2. Data *testing*

No	Input		Output
	X_0	X_1	Y
1	0.5	0.6	...
2	0.55	0.77	...
3	0.4	0.25	...
4	0.7	0.7	...
5	0	0.95	...
6	0	1	...
7	0.8	0.88	...
8	0.32	0.12	...
9	0.4	0.05	...
10	0.45	0.1	...

Misal laju pembelajaran (*learning rate*) yang digunakan adalah 0.2 dan toleransi yang diberikan adalah 0.05.

Sementara bobot-bobot awal yang di-inisialisasi diperlihatkan pada tabel berikut.

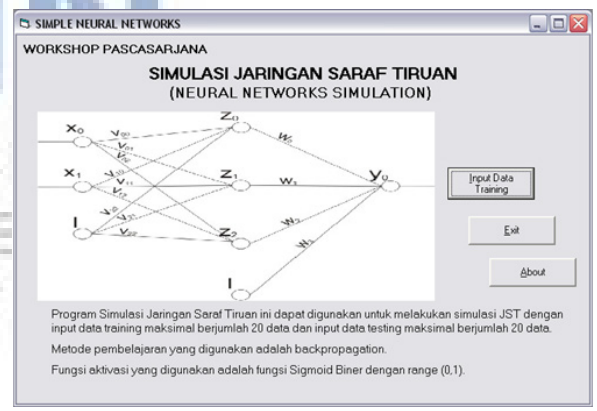
Tabel 3. Bobot-bobot awal

V_{ij}	Z_0	Z_1	Z_2
X_0	0.2	0.3	-0.1
X_1	0.5	0.1	0.3
I	-0.3	-0.1	0.3

W_{jk}	Y
Z_0	0.4
Z_1	-0.4
Z_2	0.1
I	-0.1

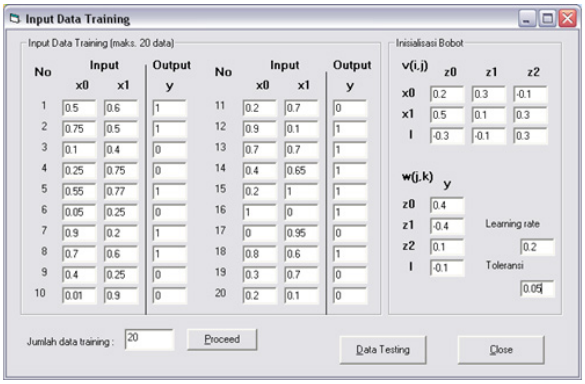
Dari arsitektur jaringan saraf tiruan yang telah ditentukan dan berdasarkan algoritma *back-propagation* yang telah dijelaskan pada subbab sebelumnya, dibangun perangkat lunak sederhana yang dapat menyelesaikan permasalahan yang diberikan dengan menggunakan Visual Basic 6.

Berikut adalah tampilan awal dari perangkat lunak yang berhasil dibangun.



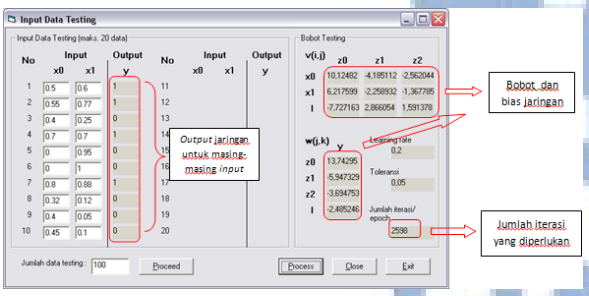
Gambar 6. Tampilan awal program

Pada tampilan awal tersebut, diberikan gambar arsitektur jaringan saraf tiruan yang digunakan beserta beberapa penjelasan singkat mengenai metode pembelajaran dan fungsi aktivasi yang digunakan, serta jumlah maksimal data *training* maupun *testing* yang dapat dimasukkan.



Gambar 7. Tampilan *input* data *training*

Bila program dilanjutkan, maka akan ditampilkan *form input* data *training* seperti yang ditampilkan pada gambar 7 di atas. *User* dapat memasukkan data *training* yang digunakan untuk melatih sistem jaringan saraf tiruan, hingga dapat diperoleh bobot-bobot yang terbaik untuk data *training* yang diberikan.



Gambar 8. Tampilan *input* data *testing*

Gambar 8 memperlihatkan *form input* data *testing* yang dapat digunakan untuk melakukan uji coba terhadap sistem jaringan saraf tiruan yang berhasil dibentuk. Dari hasil *testing*, dimana lima data pertama dari data *testing* diambil dari data *training*, diperoleh tingkat akurasi 100%. Dengan demikian dapat disimpulkan bahwa sistem jaringan saraf tiruan dengan arsitektur yang ditentukan telah berhasil dibangun.

V. PENUTUP

Jaringan saraf tiruan dengan metode pembelajaran *backpropagation* telah berhasil dibangun dengan memanfaatkan Visual Basic 6. Dari hasil uji coba yang dilakukan diperoleh tingkat akurasi sistem yang sangat baik. Namun demikian, arsitektur yang dapat diterapkan terbatas pada arsitektur jaringan *multi-layer* dengan satu *hidden layer* dan tiga *hidden node*. Jumlah *node input* maupun *output* yang digunakan dalam sistem juga telah ditentukan di awal, sehingga ke depannya dapat dibangun sistem yang lebih dinamis, dimana *user* dapat menentukan dengan bebas arsitektur jaringan saraf tiruan yang hendak digunakan.

DAFTAR PUSTAKA

[1] Setiawan, K., Paradigma Sistem Cerdas, edisi 3, Sekolah Tinggi Teknik Surabaya, Surabaya.
[2] http://trapexit.org
[3] http://informaworld.com
[4] http://nnf.sourceforge.net/
[5] http://nnwj.de
[6] http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html