JURNAL ILMIAH TEKNIK SIPIL

ARTIFICIAL NEURAL NETWORK DALAM HIDROLOGI, SUATU PENGANTAR

Oleh:
Putu Doddy Heka Ardana

ABSTRAK

Jaringan syaraf tiruan (*Artificial neural network*) menjadi salah satu pilihan ketika rumusan persoalan-persoalan yang dihadapi, khususnya dalam hidrologi, tidak bisa diselesaikan secara analitis. Dengan mengasumsikan suatu kotak hitam (*black box*) yang kita tidak tahu isinya, *artificial neural network* akan menemukan pola hubungan antara *input* dan *output* melaui fase pelatihan (*training*). *Artificial neural network* termasuk ke dalam kategori *supervised learning*.

Penelitian dengan menggunakan *artificial neural network* dalam dalam hidrologi telah banyak dilakukan. Dengan menggunakan *black box model* tersebut, maka dalam penerapannya tidak membutuhkan pengetahuan yang kompleks antar elemen-elemen, misalnya elem-elemen dalam suatu sistem DAS yang juga tidak secara eksplisit mempresentasikan hubungan antar elemen dalam DAS dan proses interaksi curah hujan-limpasan. Sehingga perubahan antar elemen di dalam suatu DAS tidak perlu diketahui sepanjang hujan dan limpasan diukur dengan akurat dari waktu ke waktu, oleh karena itu pemodelan dapat dilakukan secara lebih sederhana dengan hanya memiliki *input* hujan dan *output* limpasan di dalam pengembangan model tersebut.

Kata kunci: *artificial neural network, black box model, input, output*

1. PENDAHULUAN

Perencanaan, pengelolaan dan pengembangan sumber daya air selalu memerlukan analisa terhadap parameter hidrologi seperti curah hujan dan aliran sungai. Untuk keperluan analisa hidrologi diperlukan data hidrologi yang panjang, tetapi sering dijumpai data yang tersedia tidak lengkap atau bahkan tidak ada sama sekali. Sesuai dengan karakteristik fenomena hidrologi suatu daerah pengaliran sungai, aliran sungai berubah-ubah tidak beraturan, oleh karena itu sulit untuk meramalkan besarnya debit yang melintasi penampang sungai secara pasti pada suatu saat tertentu. Untuk mendekati fenomena tersebut maka perlu dikembangkan suatu analisa sistem hidrologi dengan menggunakan model yang merupakan penyederhanaan kenyataan alam yang sebenarnya (Hadihardaja, Sutikno, 2005).

Berbagai macam model DAS telah dikembangkan seiring dengan berkembangnya dunia digital (komputer). Baik itu model empiris (*black box model*), model konseptual (*physical process based*), model kontinyu (*continoust events*), *lumped* model, model distribusi dan model *single* (Setiawan dan Rudiyanto, 2004). Model-model tersebut dibentuk oleh satu set persamaan matematis yang mencerminkan perilaku aliran di DAS, sehingga parameter-parameter yang terkandung dalam persamaan tersebut mempunyai arti fisik. Tahap kalibrasi parameter merupakan tahap yang terpenting agar menghasilkan perpanjangan data yang handal (Adidarma, dkk., 2004).

Salah satu alternatif model yang dapat dikembangkan untuk pemodelan hujan limpasan secara berkesinambungan adalah dengan menggunakan model Jaringan Syaraf Tiruan (*Artificial Neural Network*). Model ini menggunakan satu set persamaan matematis linier dan non linier yang tidak memperhitungkan sama sekali proses fisiknya, yang penting *output* yang dihasilkan mendekati yang sebenarnya (Adidarma, dkk., 2004). *Artificial Neural Network* (ANN) yang merupakan salah satu bentuk kecerdasan buatan yang mempunyai kemampuan untuk belajar dari data dan tidak membutuhkan waktu yang lama dalam pembuatan model. ANN mampu mengidentifikasi struktur model dan juga efektif serta mampu menghubungkan *input output* simulasi dan model peramalan tanpa membutuhkan struktur internal DAS (Setiawan dan Rudiyanto, 2004).

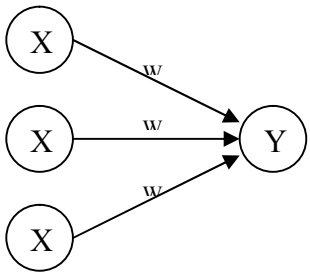
2. TEORI

2.1 *Artificial Neural Network* (Jaringan Syaraf Tiruan)

Artificial Neural Network (ANN) atau jaringan syaraf tiruan (JST) adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan syaraf biologi. Jaringan syaraf tiruan dibentuk sebagai generalisasi model matematika dari pemahaman manusia (*human cognition*).

Sistem jaringan syaraf tiruan disebut juga *brain metaphor*, *computional neuroscience* atau *parallel distributed processing* serta *conection*. Jaringan syaraf tiruan tersusun dari sejumlah besar elemen yang melakukan kegiatan yang analog dengan fungsi-fungsi biologis *neuron* yang paling elementer. Elemen-elemen ini terorganisai sebagaimana layaknya anatomi otak, walaupun tidak persis. Jaringan syaraf tiruan dapat belajar dari pengalaman, melakukan generalisasi atas contoh-contoh yang diperolehnya dan mengabstraksi karakteristik esensial input bahkan untuk data yang tidak relevan. Jaringan syaraf tiruan juga dikenal sebagai kotak hitam (*black box technology*) karena tidak dapat menerangkan bagaimana suatu hasil didapatkan. Hal inilah yang membuat jaringan syaraf tiruan mampu digunakan untuk menyelesaikan persoalan yang tidak terstruktur dan sulit didefinisikan (Hermawan, A., 2006).

Sebagai contoh, perhatikan *neuron* Y pada gambar berikut:



Gambar 1. *Neuron* pada JST

Y menerima input dari neuron X_1, X_2 dan X_3 dengan bobot hubungan masing-masing adalah w_1, w_2 dan w_3. Ketiga *impuls neuron* yang ada kemudian dijumlahkan dengan menggunakan persamaan:

$$net = X_1.w_1 + X_2.w_2 + X_3.w_3 \quad (2.1)$$

Besarnya *impuls* yang diterima oleh Y mengikuti fungsi aktivasi $y = f(net)$. Apabila nilai fungsi aktivasi cukup kuat, maka sinyal akan diteruskan. Nilai fungsi aktivasi (keluaran model jaringan) juga dapat dipakai sebagai dasar untuk merubah bobot.

2.2 Arsitektur Jaringan Syaraf Tiruan (*Artificial Neural Network Architecture*)

Pembagian arsitektur jaringan syaraf tiruan dapat dilihat dari kerangka kerja dan skema interkoneksi. Kerangka kerja jaringan syaraf tiruan bisa dilihat dari jumlah lapisan (*layer*) dan jumlah node pada setiap lapisan (LiMin Fu, 1994).

Lapisan-lapisan penyusun jaringan syaraf tiruan dapat dibagi menjadi tiga, yaitu:

1. Lapisan *input* (*input layer*)

   Node-node di dalam lapisan *input* disebut unit-unit *input*. Unit-unit *input* menerima *input* dari dunia luar. *Input* yang dimasukkan merupakan penggambaran dari suatu masalah.
2. Lapisan tersembunyi (*hidden layer*)

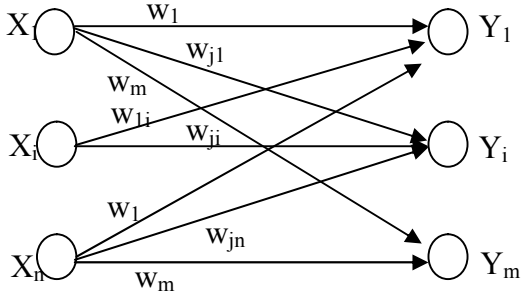
   Node-node di dalam lapisan tersembunyi disebut unit-unit tersembunyi. *Output* dari lapisan ini tidak secara langsung dapat diamati.
3. Lapisan *output* (*output layer*)

   Node-node pada lapisan *output* disebut unit-unit *output*. Keluaran atau *output* dari lapisan ini merupakan *output* jaringan syaraf tiruan terhadap suatu masalah.

Beberapa arsitektur jaringan yang yang sering dipakai dalam jaringan syaraf tiruan antara lain:

a. Jaringan Layar Tunggal (*Single Layer Network*)

Dalam jaringan ini, sekumpulan *input neuron* dihubungkan langsung dengan sekumpulan *output*-nya. Dalam beberapa model (misal model *Perceptron*), hanya sebuah unit *neuron output*.

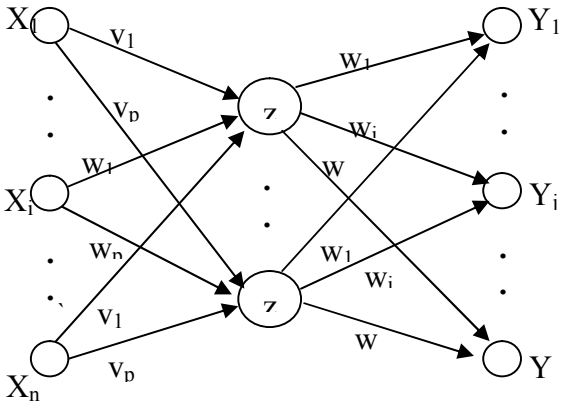


Gambar 2.2 Jaringan layar tunggal (*single layer network*)

Gambar 2.2 menunjukkan arsitektur jaringan dengan n unit input (X_1, X_2, ..., X_n) dan m buah unit output (Y_1, Y_2, ..., Y_m). Dalam jaringan ini **semua** unit *input* dihubungkan dengan **semua** unit *output*, meskipun dengan bobot yang berbeda-beda. Tidak ada unit *input* lainnya demikian pula dengan unit *output*-nya. Besaran w_{ji} menyatakan bobot hubungan antara unit ke-i dalam *input* dengan unit ke-j dalam *output*. Bobot-bobot ini saling berdiri sendiri. Selama proses pelatihan, bobot-bobot tersebut akan dimodifikasi untuk meningkatkan keakuratan hasil. Model semacam ini tepat digunakan untuk pengenalan pola karena kesederhanaannya.

b. Jaringan Layar Jamak (*Multi Layer Network*)

Jaringan layar jamak merupakan perluasan dari layar tunggal. Dalam jaringan ini, selain unit *input* dan *output*, ada unit-unit lain (sering disebut dengan layar tersembunyi/*hidden layer*). Dimungkinkan pula ada beberapa layar tersembunyi. Sama seperti pada unit *input* dan *output*, unit-unit dalam satu layar tidak saling berhubungan.



Gambar 2.3 Jaringan layar jamak (*multi layer network*)

2.3 Fungsi Aktivasi

Dalam jaringan syaraf tiruan, fungsi aktivasi merupakan bagian terpenting dalam tahapan perhitungan keluaran dari suatu algoritma (*neuron*). Argumen fungsi aktivasi adalah net masukan (kombinasi linier masukan dan bobotnya). Fungsi aktivasi (*activation functions*) berguna untuk memeras fungsi *output* supaya berada dalam nilai ambang (*threshold*) yaitu nilai yang berada diantara dua garis asimtot 0 sampai 1 atau -1 sampai 1. Batas tersebut sangat bermanfaat dalam menjaga agar supaya *output* dari proses elemen-elemen senantiasa berada dalam keadaan dinamis yang handal (Adidarma, dkk., 2004). Jika $net = \Sigma x_i w_i$, maka fungsi aktivasinya adalah $f(net) = f(\Sigma x_i w_i)$.

Menurut Fausett, L., 1994, beberapa fungsi aktivasi yang digunakan dalam jaringan syaraf tiruan adalah:

1. Fungsi identitas (linier)

   $f(x) = x$ untuk semua x (2.2)

   dengan penurunan fungsi:

   $$f'(x) = 1$$

   Fungsi identitas memiliki nilai *output* yang sama dengan nilai *input*-nya. Fungsi identitas sering dipergunakan apabila diinginkan suatu keluaran (*output*) jaringan berupa sembarang bilangan riil (bukan hanya pada range [0,1] atau [1,-1].

2. Fungsi dengan batas ambang (*with threshold* θ)

   $$f(x) = \begin{cases} 1 & jika\ x \geq 0 \\ 0 & jika\ x < 0 \end{cases} \quad (2.3)$$

   Untuk beberapa kasus, fungsi *threshold* yang dibuat tidak berharga 0 atau 1, tapi berharga -1 atau 1 (sering disebut *threshold bipolar*), sehingga:

   $$f(x) = \begin{cases} 1 & jika\ x \geq 0 \\ -1 & jika\ x < 0 \end{cases} \quad (2.4)$$

3. Fungsi sigmoid

   Fungsi aktivasi sigmoid atau fungsi logistik (*logistic functions*) digunakan untuk jaringan syaraf yang dilatih dengan menggunakan metode *backpropagation*. Fungsi sigmoid memiliki nilai pada rentang 0 (nol) sampai 1 (satu). Fungsi sigmoid dirumuskan sebagai:

   $$y = f(x) = \frac{1}{1 + e^{-\sigma x}} \quad (2.5)$$

   Fungsi sigmoid sering dipergunakan karena nilai fungsinya yang terletak 0 sampai 1 dan dapat diturunkan dengan mudah, yaitu:

   $f'(x) = \sigma f(x)\left[1 - f(x)\right]$ dengan σ = konstanta

4. Fungsi sigmoid bipolar

   Range nilai untuk fungsi bipolar sigmoid terletak dari nilai -1 sampai 1. Fungsi sigmoid bipolar dirumuskan sebagai:

$$g(x) = 2f(x) - 1 = \frac{2}{1 + e^{-\sigma x}} - 1$$

$$= \frac{1 - e^{-\sigma x}}{1 + e^{-\sigma x}} \qquad (2.6)$$

   dengan penurunan fungsi:

$$g'(x) = \frac{\sigma}{2}[1 + g(x)][1 - g(x)]$$ dengan σ = konstanta

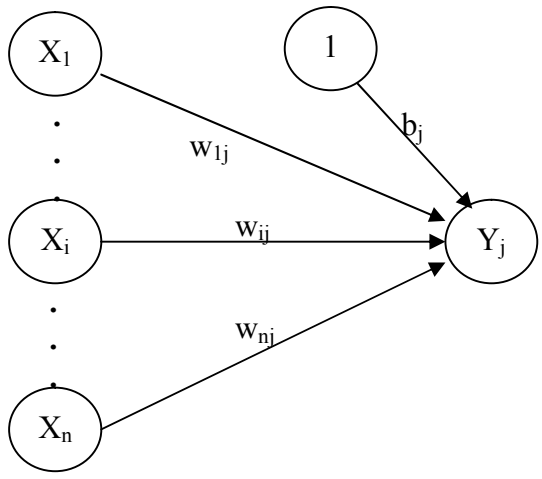
2.4 Bias dan *Threshold*

Kadang-kadang dalam jaringan ditambahkan sebuah nilai masukan yang nilainya selalu = 1. Unit yang sedemikian itu disebut dengan Bias. Bias dapat dipandang sebagai sebuah *input* yang nilainya = 1. Bias berfungsi untuk mengubah nilai *threshold* menjadi = 0 (bukan = θ). Jika melibatkan bias, maka keluaran unit penjumlah adalah (Jong Jek Siang, 2005):

$$net = b + \sum_i x_i w_i \qquad (2.7)$$

Sehingga fungsi aktivasi *threshold* menjadi:

$$f(x) = \begin{cases} 1 & jika\ net \geq 0 \\ -1 & jika\ net < 0 \end{cases} \qquad (2.8)$$



Gambar 2.4 Struktur jaringan syaraf tiruan dengan bias

2.5 Proses Pembelajaran/Pelatihan (*Training*)

Keluaran yang dihasilkan oleh jaringan syaraf tiruan untuk suatu pola masukan tertentu tergantung dari nilai (bobot) sambungan antar *neuron* dalam jaringan syaraf tiruan tersebut. Dalam jaringan syaraf tiruan, untuk aplikasi pengenalan pola diperlukan suatu pelatihan awal agar bisa digunakan di dalam menerima *input* dari luar. Sebuah jaringan syaraf tiruan dapat menyelesaikan persoalan yang rumit apabila digunakan nilai bobot yang tepat antar neuron-neuron pada lapisan yang berbeda. Nilai bobot yang tepat didapatkan melalui proses pembelajaran (*training*).

Proses pembelajaran adalah suatu proses untuk mengubah bobot antar neuron sehingga sebuah jaringan dapat menyelesaikan sebuah persoalan. Ada dua jenis proses pembelajaran, yaitu (Fendi dan Suwandi, 2000):

a. Pelatihan terbimbing (*Supervised Learning*)

   Jenis pembelajaran ini menggunakan sejumlah pasangan data *input-output* yang dipergunakan sebagai contoh, dimana data yang dipergunakan sebagai contoh sebaiknya menggunakan data yang sudah diketahui kebenarannya. *Output* dari jaringan lalu dibandingkan dengan data *output* yang diharapkan (*output* contoh) untuk mendapatkan selisih antara *output* perkiraan dengan *output* sebenarnya. Selisih inilah yang dipergunakan untuk mengubah bobot jaringan.

b. Pelatihan tak terbimbing (*Unsupervised Learning*)

   Pada proses ini tidak terdapat contoh *output* yang diharapkan dari sebuah *input*. Informasi yang tersedia hanyalah hubungan antar data *input*. Sebuah jaringan syaraf tiruan diharapkan mampu untuk menentukan kategori dari hubungan antara data *input* dan menghasilkan keluaran berdasarkan atas kategori-kategori tersebut.

2.6 Perhitungan Kesalahan (*Error*)

Perhitungan kesalahan merupakan pengukuran bagaimana jaringan dapat belajar dengan baik sehingga jika dibandingkan dengan pola yang baru akan dengan mudah dikenali. Kesalahan pada keluaran jaringan merupakan selisih antara keluaran sebenarnya (*current*

output) dan keluaran yang diinginkan (*desired output*). Selisih yang dihasilkan antara keduanya biasanya ditentukan dengan cara dihitung menggunakan suatu persamaan.

a. MSE (*Mean Square Error*)

Menurut Lekkas, D.F., et.al, persamaan yang dipergunakan untuk menghitung nilai MSE adalah sebagai berikut:

$$MSE_{(t)} = \frac{1}{2}\sum_{j=1}^{n}(y_j(t) - d_j(t))^2 \qquad (2.9)$$

Dimana : $y_j(t)$ = penjumlahan *output* hasil prediksi

$d_j(t)$ = *output* aktual

b. SSE (*Sum Square Error*)

Menurut A. Joorabchi, et.al, persamaan yang dipergunakan untuk menghitung nilai SSE adalah sebagai berikut:

$$SSE_{(t)} = \sum_{k=i}^{m}(\overline{y_{(t)}} - y_{(t)})^2 \qquad (2.10)$$

Dimana : $\overline{y}_{(t)}$ = nilai hasil prediksi

$y_{(t)}$ = nilai aktual

c. RMSE (*Root Mean Square Error*)

$$RMSE(t) = \sqrt{\frac{\sum_{i=1}^{N}(y_i - x_i)^2}{N}} \qquad (2.11)$$

Dimana : x_i = nilai observasi/aktual

y_i = nilai hasil prediksi

N = jumlah pola

d. KAR (Kesalahan Absolut Rata-Rata)

Menurut Adidarma, dkk., 2004, ukuran kedekatan atau kecocokan antara nilai hasil perhitungan model dengan data hasil pengamatan, dilakukan uji kecocokan dengan menggunakan fungsi objektif atau fungsi kesalahan yang merupakan persamaan dari perhitungan dan pengamatan. Persamaan untuk menghitung nilai KAR adalah sebagai berikut:

$$KAR = \frac{1}{n}\sum\frac{Abs(Q_{comp} - Q_{obs})}{Q_{obs}} \qquad (2.12)$$

Dimana : Q_{comp} = nilai hasil perhitungan

Q_{obs} = nilai hasil pengamatan

N = banyak data

2.7 Metode Perambatan Galat Mundur (*Backpropagation Methode*)

Terdapat lebih dari 20 model jaringan syaraf tiruan. Masing-masing model menggunakan arsitektur, fungsi aktivasi dan perhitungan yang berbeda-beda dalam prosesnya. Salah satunya adalah model perambatan galat mundur (*Backpropagation model). Backpropagation* merupakan algoritma pembelajaran yang terawasi (*supervised*) dan biasanya digunakan oleh *perceptron* dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan *neuron-neuron* yang ada pada lapisan tersembunyinya. Algoritma *backpropagation* menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error* ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, *neuron-neuron* diaktifkan dengan menggunakan fungsi aktivasi yang dapat dideferensialkan (Kusumadewi, S., 2004). Fungsi aktivasi yang digunakan dalam metode *backpropagation* antara lain:

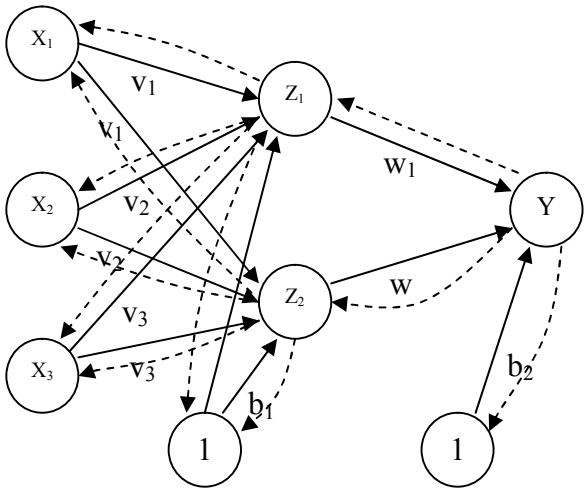
a. Fungsi identitas: $f(x) = x$

b. Fungsi sigmoid: $y = f(x) = \frac{1}{1 + e^{-\alpha x}}$

c. Fungsi sigmoid bipolar: $g(x) = \frac{1-e^{-\alpha x}}{1+e^{-\alpha x}}$

Arsitektur jaringan *backpropagation* seperti terlihat pada Gambar 2.8. Pada gambar 2.8, jaringan terdiri atas 3 unit (*neuron*) pada lapisan input, yaitu x_1, x_2, dan x_3; 1 lapisan tersembunyi dengan 2 neuron, yaitu z_1 dan z_2; serta 1 unit lapisan *output*, yaitu y. Bobot yang menghubungkan x_1, x_2, dan x_3 dengan *neuron* pertama pada lapisan tersembunyi adalah v_{11}, v_{21}, dan v_{31} (v_{ij}

adalah bobot yang menghubungkan *neuron input* ke-*i* ke *neuron* ke-*j* pada lapisan tersembunyi). Sedangkan $b1_1$ dan $b1_2$ adalah bobot bias yang menuju ke *neuron* pertama dan kedua pada lapisan tersembunyi. Bobot yang menghubungkan z_1 dan z_2 dengan *neuron* pada lapisan *output* adalah w_1 dan w_2. Bobot bias b_2 menghubungkan lapisan tersembunyi dengan lapisan *output*. Fungsi aktivasi yang dipergunakan, antara lapisan *input* dan lapisan tersembunyi, dan antara lapisan tersembunyi dengan lapisan *output* adalah fungsi aktivasi logsig (tidak diperlihatkan pada gambar).



Gambar 2.8 Arsitektur Jaringan *Backpropagation*

2.8 Optimalitas Arsitektur *Backpropagation*

Masalah utama yang dihadapi dalam *Backpropagation* adalah lamanya iterasi yang harus dilakukan. *Backpropagation* tidak dapat memberi kepastian tentang berapa *epoch*/siklus yang harus dilalui untuk mencapai kondisi yang diinginkan. Oleh karena itu orang berusaha meneliti bagaimana parameter-parameter jaringan dibuat sehingga menghasilkan jumlah iterasi yang relatif lebih sedikit (Jong Jek Siang, 2005).

a. Inisialisasi Bobot Awal Secara Random

Pemilihan bobot awal sangat mempengaruhi jaringan syaraf dalam mencapai nilai minimum global terhadap nilai *error*, serta cepat tidaknya proses pelatihan menuju kekonvergenan. Apabila nilai bobot awal terlalu besar, maka *input* ke setiap lapisan tersembunyi atau lapisan *output* akan jatuh pada daerah dimana **turunan** fungsi sigmoidnya akan sangat kecil. Sebaiknya, apabila nilai bobot awal terlalu kecil, maka ***input*** ke setiap lapisan tersembunyi atau lapisan *output* akan sangat kecil, yang akan menyebabkan proses pelatihan akan berjalan sangat lambat. Biasanya bobot awal diinisialisasikan secara random dengan nilai antara -0,5 sampai 0,5, atau -1 sampai 1, ataupun interval yang lainnya (Kusumadewi, S., 2004).

b. Inisialisasi Bobot Awal Dengan Metode *Nguyen-Widrow*

Metode *Nguyen-Widrow* akan menginisialisasi bobot-bobot lapisan dengan nilai antara -0,5 sampai 0,5. Sedangkan bobot-bobot dari lapisan input ke lapisan tersembunyi dirancang sedemikian rupa sehingga dapat meningkatkan kemampuan lapisan tersembunyi dalam melakukan proses pembelajaran. Metode Nguyen-Widrow secara sederhana dapat diimplementasikan dengan prosedur sebagai berikut (Kusumadewi, S., 2004):

0. Tetapkan:
   - n = jumlah *neuron* (unit) pada lapisan *input*
   - p = jumlah *neuron* (unit) pada lapisan tersembunyi
   - β = faktor penskalaan
1. Kerjakan untuk setiap unit pada lapisan tersembunyi (j = 1,2,...,p)
   a. Inisialisasi bobot-bobot dari lapisan *input* ke lapisan tersembunyi:
   b. Hitung:

$$\left\| v_j \right\| = \sqrt{v^2{}_{j1} + v^2{}_{j2} + ... + v^2{}_{jn}} \quad (2.13)$$

   c. Inisialisasi ulang bobot-bobot:

$$vij = \frac{\beta v_{ij}}{\left\| v_j \right\|} \quad (2.14)$$

   d. Set bias:

   $b1_j$ = bilangan random antara $-\beta$ sampai β

2.8 Pembelajaran Metode Backpropagation

Proses pembelajaran dalam *artificial neural networks* bertujuan untuk mengubah faktor bobot sehingga diperoleh bobot hubungan yang diinginkan.

Algoritma *backpropagation* menggunakan *error output* untuk mengubah bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error*, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, *neuron-neuron* diaktifkan dengan menggunakan fungsi aktivasi sigmoid. Pada awal proses pembelajaran, bobot hubungan diambil secara random. Algoritma pembelajaran, memodifikasi bobot tersebut dalam setiap *epoch* sampai pembelajaran selesai. Bila proses telah mencapai nilai maksimum *epoch* yang direncanakan atau telah mencapai nilai *error* yang direncanakan, maka proses berhenti. Koleksi bobot hubungan yang tersimpan digunakan sebagai bobot hubungan pada proses pengujian (Hadihardaja, Sutikno, 2005).

Proses pembelajaran *backpropagation* dimulai dengan menentukan target *error*, maksimum *epoch*, *learning rate* (α) dan memasukkan bobot awal secara acak dalam model. Algoritma pembelajaran metode *backpropagation* adalah sebagai berikut:

1. Inisialisasi bobot awal (ambil bobot awal dengan nilai random yang kecil)
2. Lakukan tahap perambatan maju (*forward propagation*) untuk mendapatkan *error*:
   a. Tiap unit *input* (X_i, i = 1,2,3,….,n) menerima sinyal xi dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada di atasnya (lapisan tersembunyi)
   b. Tiap-tiap unit tersembunyi (Z_j, j = 1,2,3,…., p) menjumlahkan sinyal-sinyal input terbobot:

$$z\_in_j = b_{1j} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.15)$$

   Gunakan fungsi aktivasi untuk menghitung sinyal *output*nya: $z_j = f(z\_in_j)$ dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit-unit *output*)
   c. Tiap-tiap unit *output* (Y_k, k = 1,2,3,..,m), menjumlahkan sinyal-sinyal *input* terbobot.

$$y\_in_k = b_{2k} + \sum_{i=1}^{p} z_j w_{jk} \qquad (2.16)$$

   Gunakan fungsi aktivasi untuk menghitung sinyal *output*nya: $y = f(y\_in)$, dan kirimkan sinyal tersebut ke semua unit lapisan atasnya (unit-unit *output*)
3. Lakukan tahap perambatan mundur (*backward propagation*)
   a. Tiap-tiap unit *output* (Y_k, k = 1,2,3,…,m) menerima target pola yang berhubungan dengan pola *input* pembelajaran, hitung informasi *error*nya:

$$\delta_k = (t_k - y_k)\, f'(y\_in_k) \qquad (2.17)$$

   Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai w_{jk}):

$$\Delta w_{jk} = \alpha \delta_k . z_j \qquad (2.18)$$

   Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k}):

$$\Delta w_{0k} = \alpha \delta_k \qquad (2.19)$$

   Kirimkan δ_k ini ke unit-unit yang ada di lapisan bawahnya.
   b. Tiap unit tersembunyi (Z_j, j = 1,2,3,…., p) menjumlahkan *delta input*nya (dari unit-unit yang berada di lapisan atasnya):

$$\delta\_in_j = \sum_{j=1}^{p} \delta_k . w_{jk} \qquad (2.20)$$

   Kalikan nilai ini dengan turunan dari fungsi aktivasinya untuk menghitung informasi error:

$$\delta_j = \delta\_in_j\, f'(z\_in_j) \qquad (2.21)$$

   Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai vij)

$$\Delta vi_j = \alpha \delta_j x_i \qquad (2.22)$$

   Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{oj}):

$$\Delta v_{oj} = \alpha \delta_j \qquad (2.23)$$

   c. Tiap unit output (Yk, k = 1,2,3,….,m) memperbaiki bias dan bobotnya (j = 1,2,3,….,p):

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \qquad (2.24)$$

   Tiap-tiap unit tersembunyi (Zj, j = 1,2,3,…,p) memperbaiki bias dan bobotnya (I = 1,2,3,…,n):

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \qquad (2.25)$$

Bobot-bobot baru ini digunakan sebagai bobot awal pada proses *epoch* berikutnya, proses ini dilakukan sampai target *error* tercapai atau sampai maksimum *epoch*.

4. Hitung MSE (*mean square error*)

Untuk pembelajaran artificial neural network dengan metode backpropagation dibuat dengan menggunakan program computer dengan berbagai bahasa pemrograman seperti *Borland Delphi 5, Visual Basic, MATLAB* ataupun dengan menggunakan program *Excel Link.*

3. APLIKASI *ARTIFICIAL NEURAL NETWORK* DALAM HIDROLOGI

Penelitian dengan menggunakan artificial neural network dalam hidrologi telah banyak dilakukan, salah satunya adalah di dalam memodelkan curah hujan-limpasan. Salah satu penelitian yang menggunakan artificial neural network untuk meninjau curah hujan-limpasan dilakukan oleh Hadihardaja, I.K., Sutikno, S., 2005.

Menurut Hadihardaja, I.K., Sutikno, S., 2005, hubungan curah hujan limpasan telah dikembangkan secara terus menerus dengan menerapkan *black box model* yakni *artificial neural network*. Dengan menggunakan *black box model* tersebut, maka dalam penerapannya tidak membutuhkan pengetahuan yang kompleks antar elemen-elemen dalam suatu sistem DAS yang juga tidak secara eksplisit mempresentasikan hubungan antar elemen dalam DAS dan proses interaksi curah hujan-limpasan. Sehingga perubahan antar elemen di dalam suatu DAS tidak perlu diketahui sepanjang hujan dan limpasan diukur dengan akurat dari waktu ke waktu, oleh karena itu pemodelan dapat dilakukan secara lebih sederhana dengan hanya memiliki *input* hujan dan *output* limpasan di dalam pengembangan model tersebut.

Penelitian yang dilakukan oleh Hadihardaja, I.K., Sutikno, S., menggunakan data curah hujan tengah bulanan dan debit limpasan DAS Way Sekampung_Pujorahayu, selama kurun waktu 19 tahun. Data curah hujan merupakan *input* sedangkan debit merupakan variabel *output*. Dari hasil penelitian tersbut diperoleh koefisien korelasi tertinggi sebesar 0,813 atau 81,3% sehingga dapat disimpulkan bahwa secara umum penggunaan *black box model* dengan metode *artificial neural network* dapat diterapkan dalam pemodelan curah hujan-limpasan.

DAFTAR PUSTAKA

Adidarma, W.K., Hadihardaja, I.K., Legowo, S., 2004, "*Perbandingan Pemodelan Hujan-Limpasan Antara Artificial Neural Network (ANN) dan NRECA*", Jurnal Teknik Sipil ITB, Vol. 11 No. 3.

Atan, S., Fendi, 2000, "*Aplikasi Jaringan Syaraf Tiruan Dalam Bidang Teknik Sipil*", Makalah Seminar Mahasiswa FKMTSI, Palembang.

Laurence, F., 1994, "*Fundamental of Neural Networks*", Prentice Hall, Engelwood Cliffs, New Jersey.

Hadihardaja, I.K., Sutikno, S., 2005, "*Pemodelan Curah Hujan-Limpasan Menggunakan Artificial Neural Network (ANN) dengan Metode Backpropagation*", Jurnal Teknik Sipil ITB, Vol. 12 No. 4.

Hermawan, A., 2006, "*Jaringan Syaraf Tiruan; Teori dan Aplikasi*", Andi Offset, Yogyakarta.

Jong Jek Siang, 2005, "*Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan MATLAB*", Andi Offset, Yogyakarta.

Joorabchi, A., H. Zhang, M. Blumenstein, 2007, "*Application of artificial neural networks in flow discharge prediction for the Fitzroy River, Australia*", Journal of Coastal Research, Australia.

Kusumadewi, S., 2004, "*Membangun Jaringan Syaraf Tiruan Menggunakan MATLAB dan Excel Link*", Graha Ilmu, Yogyakarta.

Lekkas, D.F., Onof, C., Lee, M.J., Baltas, E.A., 2005, "*Application Of Artificial Neural Networks For Flood Forecasting*", Global Nest, Greece.

LiMin Fu, 1994, "*Neural Networks In Computer Intelligence*", McGraw-Hill Inc., Singapore

Puspitaningrum, D., 2006, "*Pengantar Jaringan Syaraf Tiruan*", Andi Offset, Yogyakarta.

Setiawan, B.I., Rudiyanto, 2004, "*Aplikasi Neural Networks Untuk Prediksi Aliran Sungai*", Prosiding Semiloka Teknologi Simulasi dan Komputasi serta Aplikasi 2004 – BPPT, Jakarta.