BAB II

KAJIAN PUSTAKA

A. JARINGAN SYARAF TIRUAN

1. PENGERTIAN JARINGAN SYARAF TIRUAN

Jaringan Syaraf Tiruan (JST) adalah prosesor tersebar paralel (*parallel distributed processor)* yang sangat besar yang memiliki kecenderungan untuk menyimpan pengetahuan yang bersifat pengalaman dan membuatnya siap untuk digunakan. JST menyerupai otak manusia dalam dua hal, yaitu: Pengetahuan diperoleh jaringan melalui proses belajar; Kekuatan hubungan antar sel syaraf (neuron) yang dikenal sebagai bobot - bobot *sinaptik* digunakan untuk menyimpan pengetahuan (Suyanto, 2014).

Menurut Shanmuganathan & Samarasinghe (2016), Jaringan syaraf tiruan (*artificial neural network*) adalah model komputasi yang terinspirasi secara biologis, jaringan syaraf tiruan terdiri dari beberapa elemen pengolahan (*neuron*) dan ada hubungan antara neuron. Jaringan syaraf tiruan dapat digambarkan sebagai model matematis dan komputasi untuk fungsi aproksimasi *non-linear*, klasifikasi data *cluster* dan *regresi non-parametrik* atau sebuah simulasi dari korelasi model syaraf biologi.

Model jaringan syaraf ditunjukkan dengan kemampuan dalam emulasi, analisis, prediksi, dan asosiasi. Kemapuan yang dimiliki

jaringan syaraf tiruan dapat digunakan untuk belajar dan menghasilkan aturan atau operasi dari beberapa contoh atau *input* yang dimasukkan dan membuat prediksi tentang kemungkinan *output* yang akan muncul atau menyimpan karakteristik dari *input* yang disimpan kepada jaringan syaraf tiruan.

Setiap pengolahan elemen membuat perhitungan berdasarkan pada jumlah masukan (*input*). Sebuah kelompok pengolahan elemen disebut *layer* atau lapisan dalam jaringan. Lapisan pertama adalah *input* dan yang terakhir adalah *output*. Lapisan diantara lapisan *input* dan *output* disebut dengan lapisan tersembunyi (*hidden layer*). Salah satu organisasi yang dikenal dan sering digunakan dalam paradigma jaringan syaraf tiruan adalah perambatan galat mundur / *backpropagation* (Hermawan, 2006).

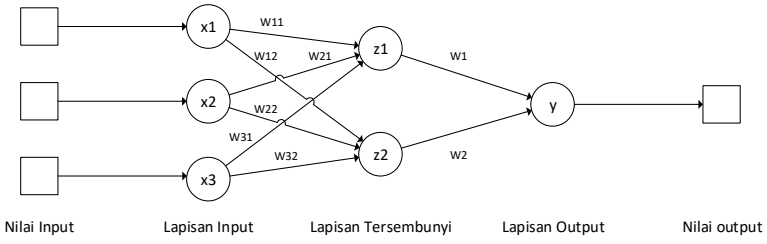
Penggunaan neuron dalam lapisan tersembunyi yang terlalu sedikit dalam lapisan tersembunyi akan menyebabkan *underfitting*. *Underfitting* terjadi apabila neuron yang berada dalam lapisan tersembunyi terlalu sedikit dalam mendeteksi sinyal dalam sebuah himpunan data yang kompleks/rumit. Begitupun sebaliknya, ketika neuron dalam lapisan tersembunyi terlalu banyak maka dapat mengakibatkan beberapa masalah. Pertama, neuron yang terlalu banyak dalam lapisan tersembunyi dapat mengakibatkan *overfitting. Overfitting* terjadi ketika jaringan syaraf memiliki kapasitas pemrosesan informasi terlalu banyak, sedangkan keterbatasan jumlah informasi yang ada pada

kumpulan pelatihan (*training set*) tidak cukup untuk melatih semua neuron dalam lapisan tersembunyi. Masalah kedua terjadi ketika data pelatihan tidak memadai. Sebagian besar neuron dalam lapisan tersembunyi dapat meningkatkan waktu yang diperlukan untuk melatih jaringan.

2. Arsitektur Jaringan Syaraf Tiruan

Jaringan syaraf tiruan dirancang dengan menggunakan suatu aturan yang bersifat menyeluruh (*general rule*) dimana seluruh model jaringan memiliki konsep dasar yang sama. Arsitektur sebuah jaringan akan menentukan keberhasilan target yang akan dicapai karena tidak semua permasalahan dapat diselesaikan dengan arsitektur yang sama. Beberapa arsitektur jaringan syaraf tiruan diantaranya jaringan dengan lapisan tunggal (*single layer net*), jaringan dengan banyak lapisan (*multilayer net*), dan jaringan dengan lapisan kompetitif (*competitif layer net*). Pada penelitian ini arsitektur jaringan yang digunakan adalah *multilayer net* atau jaringan dengan banyak lapisan.

Jaringan dengan banyak lapisan memiliki satu atau lebih lapisan yang terletak diantara lapisan *input* dan lapisan *output* seperti terlihat pada Gambar 1.



Gambar 1. Jaringan Syaraf Tiruan dengan Banyak Lapisan

Umumnya terdapat lapisan bobot – bobot yang terletak antara 2 lapisan yang bersebelahan. Jaringan dengan banyak lapisan dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan tunggal, dengan pembelajaran yang lebih rumit.

B. ALGORITMA PEMBELAJARAN *BACKPROPOGATION*

Pada *backpropogation* terdapat 12 algoritma pelatihan yang digunakan diantaranya yaitu algoritma *Fletcher-Reeves Update (traincgf), Polak-Ribiĕre (traincgp), Powell-Beale Restarts (traincgb), Scaled Conjugate Gradient (trainscg), Gradient Descent (traingd), Gradient Descent dengan Adaptive Learning Rate (traingda), Gradient Descent dengan Momentum (traingdm), Gradient Descent dengan Momentum dan Adaptive Learning Rate (traingdx), Resilent Backpropagation (trainrp), BFGS (trainbfg), One Step Secant (trainoss),* dan *Levenberg-Marquardt (trainlm).* Diantara 12 algoritma pelatihan tentunya perlu dipilih algoritma yang paling optimal sehingga diperoleh hasil yang terbaik, dalam penelitian tersebut dilakukan pengujian terhadap algoritma pelatihan yang terdapat dalam jaringan backpropogation sejumlah 12 algoritma dan algoritma pelatihan ditinjau dari *error* yang dihasilkan (Mustafidah & Suwarsito, 2015b).

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 dideferensiasikan seperti sigmoid, persamaan dan perhitungan – perhitungan pada persamaan (1), (2), (3).

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

dengan : $f'^{(x)} = \sigma f(x)[1 - f(x)]$

atau tangen sigmoid :

$$y = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

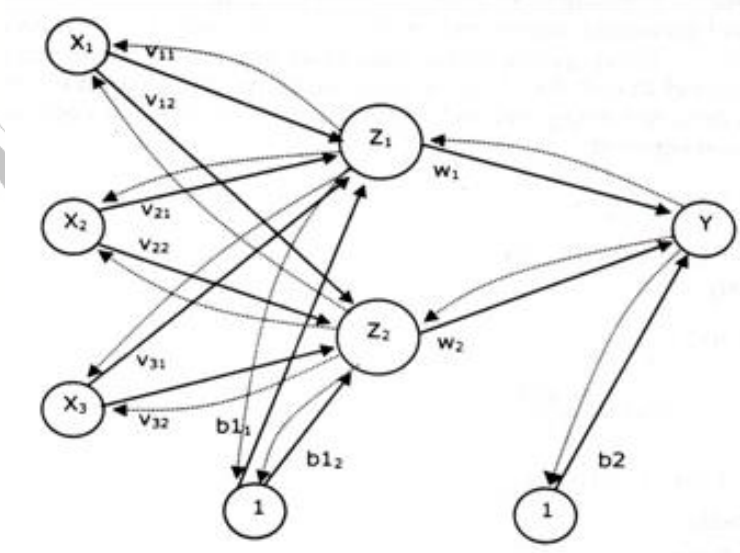
atau $y = f(x) = \frac{1- e^{-2x}}{1+ e^{-2x}}$ ..... (2)

dengan $f'(x) = [1 + f(x)][1 - f(x)]$

atau purelin : $y = f(x) = x$

$$f'(x) = 1 \text{ .....} \quad (3)$$

Arsitektur jaringan *backpropagation* seperti terlihat pada Gambar 2.



Gambar 2. Arsitektur Backpropagation

Pada gambar 2, jaringan terdiri dari 3 unit atau 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 pada 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}: bobot yang menghubungkan neuron *input* ke-i ke neuron ke-j pada lapisan tersembunyi). Perlu diingat bahwa, untuk pemakaian *toolbox* nnet pada Matlab, bobot v_{ij} memiliki pengertian yang sebaliknya (v_{ij}: bobot yang menghubungkan neuron ke-j pada suatu lapisan ke neuron ke-i pada lapisan sesudahnya). Misalnya: v_{12} adalah bobot yang menghubungkan neuron ke-2 pada lapisan *input*, ke neuron ke-1 pada lapisan tersembunyi). Pada $b1_1$ 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 b2 menghubungkan lapisan tersembunyi dengan lapisan *output*.

Berikut adalah algoritma *backpropagation* :

- Menginisialisasi bobot (ambil bobot awal dengan nilai random yang cukup kecil).
- Mengerjakan langkah–langkah berikut selama kondisi berhenti bernilai FALSE.

  Langkah–langkahnya seperti berikut :

  a. Untuk tiap pasangan elemen yang akan dilakukan:

     Feedforward (persamaan 4 – 7) :

1. Tiap–tiap unit input (Xi, i=1,2,3,…,n) menerima sinyal x_i dan meneruskan sinyal tersebut kesemua unit pada lapisan yang ada diatasnya (lapisan tersembunyi).
2. Tiap–tiap unit tersembunyi (Zj, j=1,2,3,…,p) menjumlahkan sinyal–sinyal input terbobot:

   $z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$ …….. (4)

   Kemudian gunakan fungsi aktivasi untuk menghitung sinyal outputnya:

   $z_j = f(z_{in_j})$…… (5)

   Dan mengirimkan sinyal tersebut kesemua unit di lapisan atasnya (unit–unit output).
3. Tiap–tiap unit output (Yk, k=1,2,3,…,m) menjumlahkan sinyal–sinyal input terbobot.

   $y_{in_k} = w_{ok} + \sum_{i=1}^{p} z_i w_{jk}$ ……...(6)

   Gunakan fungsi aktivasi untuk menghitung sinyal outputnya:

   $y_k = f(y_{in_k})$ …….. (7)

   Dan kirimkan sinyal tersebut ke semua unit di lapisan atasnya (unit–unit output).

Backpropagation (persamaan 8 – 16) :

4. Tiap–tiap unit output (Yk, k=1,2,3,…,m) menerima target pola yang berhubungan dengan pola input pembelajaran, menghitung informasi *error* nya:

   $\delta_k = (t_k - t_y) f'(y_{in_k})$ ……..… (8)

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

$\Delta w_{jk} = \alpha\, \delta_k\, z_j$ ……..…. (9)

Selain itu hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai awal w_{0k}):

$\Delta w_{0k} = \alpha\, \delta_k$ ……..……. (10)

Mengirim δ_k ini ke unit–unit yang ada pada lapisan bawahnya.

5. Tiap–tiap unit tersembunyi (Zj, j=1,2,3,…,p) menjumlahkan delta inputnya (dari unit–unit yang berada pada lapisan atasnya):

$\delta_{in_j} = \sum_{k=1}^{m} \delta_k\, w_{jk}$ ……..…… (11)

Mengalikan nilai tersebut dengan turunan dari fungsi aktivasinya untuk menghitung informasi *error*:

$\delta_j = \delta\_in_j f'(z_{\_in_j})$ …….. (12)

Kemudian menghitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}):

$\Delta v_{jk} = \alpha\, \delta_j\, x_i$ ……..…….(13)

Selain itu hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j}):

$\Delta v_{0j} = \alpha\, \delta_j$ ……..…..(14)

6. Tiap–tiap unit output (Yk, k=1,2,3,…,p) memperbaiki bias dan bobotnya (j=0,1,2,…,p):

$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$ ……....(15)

Tiap–tiap unit tersembunyi (Zj, j=1,2,3,…,p) memperbaiki bias dan bobotnya (i=0,1,2,3,…,n):

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad \text{.....(16)}$$

C. HIDDEN LAYER

Jaringan syaraf memiliki dua lapisan yaitu lapisan *input* dan keluaran *output*. Diantara dua lapisan ini terdapat lapisan lain yang disebut lapisan tersembunyi. Lapisan tersembunyi ini disisipkan di antara lapisan *input* dan *output*. Lapisan yang tersembunyi ini mengambil struktur yang lebih kompleks. Tujuan dari lapisan tersembunyi adalah memungkinkan jaringan syaraf menghasilkan *output* yang diharapkan dari *input* yang diberikan. Lapisan tersembunyi seperti sebuah "kotak hitam." ini didefinisikan dalam istilah jumlah neuron untuk lapisan tersembunyi dan lapisan *output*. Jaringan syaraf menghasilkan output yang benar yang dilakukan oleh lapisan tersembunyi. Salah satu tantangannya adalah untuk menghindari pembuatan struktur tersembunyi yang kompleks atau terlalu sederhana. Struktur tersembunyi yang terlalu rumit akan memakan waktu terlalu lama untuk melatih. Apabila strukturnya terlalu sederhana maka tidak akan banyak ditemukan masalahnya. Titik awal yang baik adalah lapisan tersembunyi tunggal dengan jumlah neuron yang sama dengan dua kali lapisan *input*. Berdasarkan kinerja jaringan ini, jumlah lapisan tersembunyi dari jumlah neuron akan meningkat atau menurun. Berdasarkan penelitian dari Heaton (2008), untuk menentukan neuron yang digunakan dalam lapisan tersembunyi berdasarkan :

1. Jumlah neuron tersembunyi sebaiknya berada diantara neuron lapisan *input* dan *output*.
2. Jumlah neuron yang tersembunyi sebaiknya 2/3 ukuran lapisan *input* ditambah neuron lapisan keluaran.
3. Jumlah neuron yang tersembunyi sebaiknya kurang dari dua kali neuron dari lapisan *input*.

D. PARAMETER JARINGAN

Beberapa parameter algoritma seperti yang disampaikan oleh Kusumadewi (2004) adalah:

1. Epoh

   Epoh menunjukkan banyaknya iterasi/langkah yang diperlukan oleh jaringan dalam melaksanakan pelatihan hingga dipenuhi kriteria tertentu untuk berhenti.

2. Kinerja tujuan (target *error*)

   Kinerja tujuan merupakan target nilai fungsi kinerja (*error*).

3. *Learning rate* (lr)

   Learning rate adalah laju pembelajaran. Nilai lr = 0 s/d 1.

4. Jumlah epoh yang akan ditunjukkan kemajuannya

   Parameter ini menunjukkan berapa jumlah epoh berselang yang akan ditunjukkan kemajuannya.

5. Jumlah neuron dalam lapisan tersembunyi

Penggunaan neuron dalam lapisan tersembunyi yang terlalu sedikit dalam lapisan tersembunyi akan menghasilkan sesuatu yang disebut *underfitting*.

E. STUDI PENDAHULUAN

1. Algoritma pelatihan *traincgf, traincgp, traincgb,* dan *trainscg* telah diuji keoptimalannya dalam penyelesaian kasus prediksi prestasi belajar mahasiswa dan dihasilkan kesimpulan bahwa dari keempat algoritma tersebut dengan *alpha* 5% tidak berbeda keoptimalannya secara signifikan (Mustafidah et al., 2013).
2. Tingkat ketelitian pengenalan pola data dari beberapa algoritma yaitu *traingda, traingdx, trainrp, trainbfg, trainoss,* dan *trainlm* telah diuji oleh Wibowo et al. (2013) dan menghasilkan informasi bahwa dengan tingkat kepercayaan 95% diperoleh hasil bahwa algoritma *trainlm* merupakan algoritma yang paling teliti dengan rata-rata *error* 0,0063.
3. Algoritma pelatihan 12 dalam jaringan syaraf telah diuji, yaitu *traingd*, *traingdm*, *traingdx*, *trainrp*, *traingda, traincgf*, *traincgp*, *traincgb*, *trainscg, trainbfg, trainoss,* dan *trainlm* dalam kesesuaian mengenali pola data. Parameter jaringan termasuk maksimum epoh = 1000, tingkat pembelajaran = 0,05, dan kesalahan target = 0,001. Hasil tes diperoleh bahwa pada interval kepercayaan 95% dari *trainlm* adalah algoritma yang paling tepat di mengenali pola data dengan tingkat rata-rata ketepatan 87,5% (Mustafidah et al., 2014).

4. Mustafidah & Suwarsito (2015a) telah melakukan penelitian dengan menguji 12 algoritma pelatihan. Parameter jaringan yang digunakan adalah target *error* = 0,001 (10^{-3}), maksimum epoh = 10000 dengan 5 neuron input dan 1 neuron output. Dari hasil tes 12 algoritma pelatihan, dihasilkan bahwa algoritma *Levenberg-Marquardt* memiliki tingkat *error* terkecil pada nilai α = 5 % dan nilai *error* sebesar 0,0001986858. Dengan demikian maka algoritma *trainlm* dianggap sebagai algoritma pelatihan paling optimal untuk aplikasi dalam menyelesaikan masalah.

F. MSE (*MEAN SQUARED ERROR*)

Perhitungan kesalahan atau *error* 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 ditemukan dengan cara dihitung menggunakan suatu persamaan. Suatu *error* dapat dihitung menggunakan MSE (*mean squared error*). MSE merupakan fungsi kinerja jaringan yang mengukur kinerja berdasarkan rata-rata dari kuadrat *error* (Kusumadewi, 2004).

Persamaan yang digunakan terlihat pada persamaan (17).

$$\text{MSE} = \frac{\sum p \sum j (T_{jp} - X_{jp})^2}{n_p n_o} \quad \text{.....(17)}$$

T_{jp} = nilai keluaran jaringan syaraf

X_{jp} = nilai target yang diinginkan untuk setiap keluaran

n_p = jumlah seluruh pola

n_o = jumlah keluaran

MSE dapat memberikan gambaran terhadap seberapa konsisten model yang dibangun. Dengan meminimalkan nilai MSE, berarti meminimalkan varian model. Model yang memiliki varian kecil mampu memberikan hasil yang relatif lebih konsisten untuk seluruh data *input* dibandingkan dengan model dengan varian besar (MSE besar).

G. MATLAB

MATLAB merupakan perangkat lunak yang digunakan untuk pemrograman, analisis, serta komputasi teknis dan matematis berbasis matriks. MATLAB adalah singkatan dari *Matrix Laboratory* karena mampu menyelesaikan masalah perhitungan dalam bentuk matriks. Bahasa pemrograman yang kini dikembangkan oleh MathWorks Inc. menggabungkan proses pemrograman, komputasi, dan visualisasi melalui lingkungan kerja yang mudah digunakan. MATLAB juga mempunyai keunggulan umum lainnya, seperti analisis dan eksplorasi data, pengembangan algoritma, pemodelan dan simulasi, visualisasi plot dalam bentuk 2D dan 3D, hingga pengembangan aplikasi antar muka grafis (Tjolleng, 2017).

Perangkat lunak ini menawarkan kemudahan dan kesederhanaan dalam menyelesaikan permasalahan yang berhubungan dengan vektor dan matriks. Memperoleh inversi dan menyelesaikan persamaan linier

merupakan contoh permasalahan yang dapat dipecahkan dengan cara yang sangat singkat dan mudah sekali. Untuk menangani persoalan – persoalan spesifik, matlab menyediakan sejumlah *toolbox* yang bisa dipakai untuk mempermudah perhitungan, misalnya *toolbox neural network* yang bisa dipakai untuk fungsi – fungsi yang berkaitan dengan jaringan syaraf tiruan (Irawan, 2012).

H. SPSS

Taniredja & Mustafidah (2011) menyatakan bahwa SPSS singkatan dari *Statistical Package For Social Science* yaitu merupakan paket statistika untuk ilmu – ilmu sosial, akan tetapi SPSS banyak juga digunakan untuk bidang – bidang lain yang memang membutuhkan statistika. Sejak dikeluarkannya SPSS dengan versi *under* DOS sampai sekarang dengan versi *under* Windows, sudah dikembangkan SPSS sampai generasi atau *release* 17 yang paling baru dengan penambahan fasilitas yang makin lengkap seperti grafik pengendali untuk *Quality control* dan penambahan fasilitas untuk *link* S-Plus yaitu *Package* Statistika terbaru yang sangat cocok untuk tujuan ilmiah dan pengembangannya, tidak hanya pengolahan data semata. SPSS berfungsi untuk membantu memproses data – data statistik secara cepat dan tepat, serta menghasilkan berbagai output yang dikehendaki oleh para pengambil keputusan. Statistik dapat diartikan sebagai suatu kegiatan yang bertujuan untuk mengumpulkan data, meringkas atau menyajikan data kemudian menganalisis data dengan

menggunakan metode tertentu, dan menginterpretasikan hasil dari analisa tersebut.

I. ANAVA

Anava atau disebut juga Anova (*Analysis of Variance*) merupakan perluasan dari uji rata–rata k sampel, dengan k > 2.

- H_0: $\mu_1 = \mu_2 = \cdots = \mu_k$ VS H1: minimal 2 *mean* tidak sama.
- $F_{hit} = \frac{MST}{MSE} \sim F_{k-1,N-k}$

Dimana MST = *Mean Square of Treatment*

MSE = *Mean Square of Error*

Pada ANAVA jika H_0 ditolak maka kita masih mempunyai pekerjaan untuk menentukan *mean–mean* mana saja yang berbeda. Untuk itu perlu dilakukan MCA (*Multiple Comparison Analysis*) atau Analisis Pembandingan Ganda (Taniredja & Mustafidah, 2011).