BAB II

KAJIAN PUSTAKA

A. Jaringan Syaraf Tiruan

1. Definisi Jaringan Syaraf Tiruan

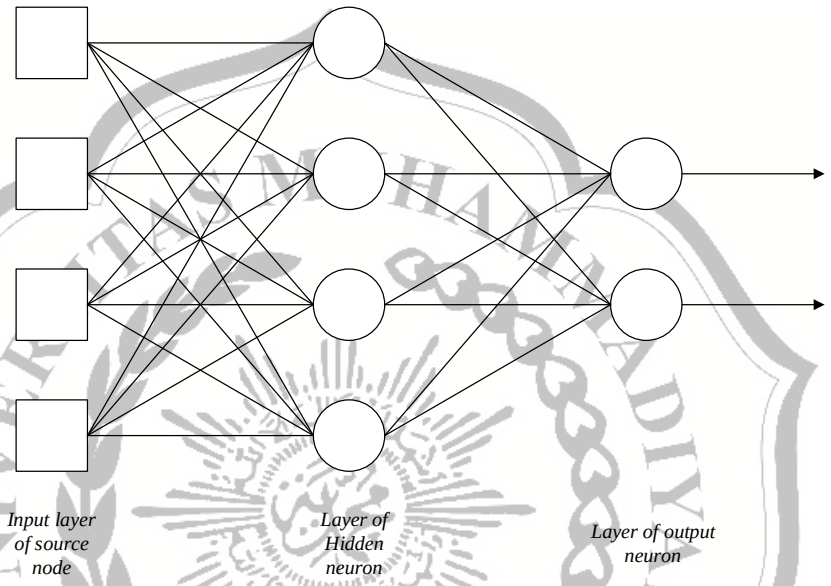
Jaringan syaraf tiruan (JST), yaitu metode *learning* yang bisa digunakan untuk permasalahan yang bernilai diskrit, real, maupun vektor. JST adalah prosesor tersebar paralel (*paralel distributed processor*) yang sangat besar yang memiliki kecenderungan untuk menyimpan pengetahuan yang bersifat pengalaman 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, 2014b).

2. Arsitektur Jaringan Syaraf Tiruan

Menurut Suyanto (2014) Pola di mana neuron-neuron pada JST disusun berhubungan erat dengan algoritma belajar yang digunakan untuk melatih jaringan. Berikut arsitektur jaringan syaraf tiruan dengan banyak lapisan (*multi-layer*).

Kelas ke dua dari *feedforward neural network* adalah jaringan dengan satu atau lebih lapis tersembunyi (*hidden layer)*, dengan *computation nodes* yang berhubungan disebut *hidden neuron* atau *hidden units*. Gambar 1 mengilustrasikan *multilayer feedforward neural network*

untuk kasus satu *hidden layer*. Untuk menyingkat jaringan pada Gambar 1 tersebut, biasanya disebut sebagai jaringan 4-4-2, dalam arti bahwa jaringan tersebut mempunyai 4 *node* masukan, 4 *hidden neurons*, dan 2 *output neurons*.



Gambar 1. *Fully connected feedforward network* dengan satu *hidden layer* dan satu *output layer* (jaringan 4-4-2)

B. Algoritma Pembelajaran *Backpropagation*

Menurut Kusumadewi (2004), Backpropagation merupakan algoritma pembelajaran yang terawasi 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 aktifasi yang dapat

dideferensikan seperti sigmoid seperti pada persamaan 1, persamaan 2 dan persamaan 3.

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

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

atau tansig:

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

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

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

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

$f'(x) = 1$ .....(3)

Arsitektur jaringan *backpropagation* seperti terlihat pada Gambar 2, jaringan terdiri atas 3 unit (*neuron*) pada lapisan *input*, yaitu x_1, x_2 dan x_3, 1 lapisan *output*, yaitu y. Bobot yang meghubungkan x_1, x_2 dan x_3 dengan neuron pertama pada lapisan tersembunyi, adalah v_{11}, v_{12}, v_{13} (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 neuron ke-i pada lapisan sesudahnya). Misal: v_{12} adalah bobot yang menghubungkan neuron ke-2 pada lapisan *input*, ke neuron ke-1 pada lapisan tersembunyi. Kembali ke

Gambar 2, b_1 dan b_1 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 digunakan, antara lapisan *input* dan lapisan tersembunyi, dan antara lapisan tersembunyi dengan lapisan *output* adalah fungsi aktivasi *logsig*.

Gambar 2. Arsitektur Jaringan *Backpropagation*

Berikut adalah algoritma *backpropagation* :

- Inisialisasi bobot (ambil bobot awal dengan nilai random yang cukup kecil).
- Tetapkan: Maksimum *Epoh*, Target *Error*, dan *Learning Rate* (α).
- Inisialisasi: *Epoh* = 0, MSE = 1.
- Kerjakan langkah – langkah berikut selama (*Epoh* < Maksimum Epoh) dan (MSE > Target Error):
  a. *Epoh* = *Epoh* + 1

b. Untuk tiap-tiap pasangan elemen yang akan dilakukan pembelajaran, kerjakan:

Feedforward:

1. Tiap – tiap unit *input* (X_i, i=1,2,3,…,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada di atasnya (lapisan tersembunyi).
2. Tiap – tiap unit tersembunyi (Z_j, j=1,2,3,…,p) menjumlahkan sinyal – sinyal *input* terbobot seperti pada persamaan 4.

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

   Gunakan fungsi aktivasi untuk menghitung sinyal *output*-nya seperti pada persamaan 5.

   $z_j = f(z_{in_j})$……..(5)

   Dan kirimkan 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 seperti pada persamaan 6.

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

   Gunakan fungsi aktivasi untuk menghitung sinyal *output*-nya seperti pada persamaan 7.

   $y_k = f(y_{\_in_k})$ ……..(7)

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

Backpropagation:

4. Tiap – tiap unit *output* (Yk, k=1,2,3,…,m) menerima target pola yang berhubungan dengan pola *input* pembelajaran, menghitung informasi errornya seperti pada persamaan 8, persamaan 9 dan persamaan 10.

   $\delta 2_k = (t_k - t_y) f'(y_{in_k})$

   …….....(8)

   $\varphi 2_{jk} = \delta_k z_j$ …….....(9)

   $\beta 2_k = \delta_k$ …….....

   (10)

   Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai w_{jk}) seperti pada persamaan 11.

   $\Delta w_{jk} = \alpha \varphi 2_{jk}$ ……..... (11)

   Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai awal $b2_k$) seperti pada persamaan 12.

   $\Delta b2_k = \alpha \beta 2_k$ ……..... (12)

   Langkah (d) ini juga dilakukan sebanyak jumlah lapisan tersembunyi, yaitu menghitung informasi error dari suatu lapisan tersembunyi ke lapisan tersembunyi sebelumnya.

5. Tiap – tiap unit tersembunyi (Zj, j=1,2,3,…,p) menjumlahkan delta *input*nya (dari unit – unit yang berada pada lapisan di atasnya) seperti pada persamaan 13.

   $\delta_{-in_j} = \sum_{k=1}^{m} \delta 2_k \, w_{jk}$

   …….....(13)

Kalikan nilai ini dengan turunan dari fungsi aktivasinya untuk menghitung informasi error seperti pada persamaan 14, persamaan 15 dan persamaan 16.

$\delta 1_j = \delta\_in_j f'(z\_in_j)$ …….....(14)

$\varphi 1_{ij} = \delta 1_j x_j$ …….....(15)

$\beta 1_j = \delta 1_j$ …….....(16)

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}) seperti pada persamaan 17.

$\Delta v_{ij} = \alpha \varphi 1_{ij}$

…….....(17)

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai $b1_j$) seperti pada persamaan 18.

$\Delta b1_j = \alpha \beta 1_j$…….....(18)

6. Tiap – tiap unit *output* (Yk, k=1,2,3,…,p) memperbaiki bias dan bobotnya (j=0,1,2,…,p) seperti pada persamaan 19 dan persamaan 20.

$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$ …….....(19)

$b2_k(baru) = b2_k(lama) + \Delta b2_k$ …….....(20)

Tiap – tiap unit tersembunyi (Zj, j=1,2,3,…,p) memperbaiki bias dan bobotnya (i=0,1,2,3,…,n) seperti pada persamaan 21 dan persamaan 22.

$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$ …….....(21)

$b1_j(baru) = b1_j(lama) + \Delta b1_j$ …….....(22)

C. Lapisan Tersembunyi (*Hidden Layer*)

Menurut Heaton (2008) Menentukan jumlah neuron di lapisan tersembunyi adalah bagian yang penting dalam menentukan asitektur neural, walaupun dalam lapisan ini tidak secara langsung beinteraksi dengan lingkungan eksternal akan tetapi, lapisan ini memiliki pengaruh besar pada hasil akhir. Pada penlitian ini menentukan jumlah neuron yang digunakan dalam lapisan tersembunyi berdasarkan :

a. Jumlah neuron tersembunyi harus berada di antara neuron lapisan *input* dan neuron lapisan *output*
b. Jumlah neuron yang tersembunyi harus 2/3 ukuran lapisan *input* ditambah neuron lapisan keluaran
c. Jumlah neuron yang tersembunyi harus kurang dari dua kali neuron dari lapisan *input.*

D. Matlab

Menurut Tjolleng (2017) MATLAB merupakan perangkat lunak yang digunakan untuk pemrograman, analisis, serta komputasi teknik dan matematis berbasis matriks. MATLAB adalah singkatan dari *Matrix Laboratory* karena mampu menyelesaikan masalah perhitungan dalam bentuk matriks. MATLAB versi pertama dirilis pada tahun 1970 oleh Cleve Moler. Pada awalnya, MATLAB didesain untuk menyelesaikan masalah-masalah persamaan aljabar linear. Seiring berjalannya waktu, program ini terus mengalami perkembangan dari segi fungsi dan performa komputasi.

Bahasa pemrograman yang kini dikembangkan oleh MathWorks Inc. menggabungkan proses pemrograman, komputasi, dan visualisasi melalui lingkungan kerja yang mudah digunakan. MATLAB juga memiliki 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. Dalam ruang lingkup perguruan tinggi, MATLAB digunakan sebagai alat pembelajaran pemrograman matematika, teknik, dan sains pada level pengenalan dan lanjutan, sedangkan dalam dunia industri, MATLAB dipilih sebagai alat penelitian, pengembangan, dan analisis produk industri.

MATLAB dapat dioperasikan pada sistem operasi Windows, Linux, maupun macOS. Selain itu, MATLAB juga bisa dihubungkan dengan aplikasi atau bahasa pemrograman eksternal lainnya, seperti C, Java, .NET, dan Microsoft Excel. Dalam MATLAB tersedia pula kotak kakas (*toolbox*) yang dapat digunakan untuk aplikasi-aplikasi khusus, seperti pengolahan sinyal, sistem kontrol, logika *fuzzy*, jaringan saraf tiruan, optimasi, pengolahan citra digital, bioinformatika, simulasi, dan berbagai teknologi lainnya.

Menurut Kusumadewi (2004) Dalam pelatihan *Backpropagation* dapat menggunakan perintah `train` ada beberapa parameter pelatihan yang dapat digunakan diantaranya yaitu:

a. Maksimum *Epoh*

Maksimum *epoh* adalah jumlah *epoh* maksimum yang boleh dilakukan selama proses pelatihan. *Iterasi* akan dientikan apabila nilai *epoh* melebihi maksimum *epoh*.

Intruksi: `net.trainParam.epochs = MaxEpoh`

Nilai default untuk maksimum epoh adalah 10.

b. Kinerja tujuan atau target error

Kinerja tujuan adala target nilai fungsi kinerja. Iterasi akan dientikan apabila nilai fungsi kineja kurang dari atau sama dengan kinerja tujuan.

Intruksi: `net.trainParam.goal = TargetEror`

Nilai default untuk kinerja tujuan adalah 0.

c. *Learning Rate*

Learning rate adalah laju pembelajaran.

Intruksi: `net.trainParam.lr = LearningRate`

Niali default untuk *learning rate* adalah 0.01

d. Rasio untuk Menaikan *Learning Rate*

Rasio ini berguna sebagai faktor penggali untuk menaikan *learning rate* apabila *learning rate* yang ada terlalu rendah untuk mencapai *kekonvergenan*.

Intruksi:

`net.trainParam.lr_inc = IncLearningRate`

nilai default untuk rasio kenaikan *learning rate* adalah 1.05.

e. Rasio untuk Menurunkan *Learning Rate*

Rasio ini berguna sebagai faktor pengali untuk menurunkan *learning rate* apabila *learning rate* yang ada terlalu tinggi dan menurunkan ketidakstabilan

Intruksi:

```
net.trainParam.lr_decc = DecLearningRate
```

nilai default untuk rasio penurunan *learning rate* adalah 0.7.

f. Waktu Maksimum untuk Pelatihan

Menunjukan waktu maksimum yang diijinkan untuk melakukan pelatihan. *Iterasi* akan dihentikan apabila waktu pelatihan melebihi waktu maksimum.

Intruksi : `net.trainParam.time = MaxTime`

Nilai default untuk waktu maksimum adalah tak terbatas (inf).

E. Algoritma Pelatihan

Dalam JST (jaringan syaraf tiruan) terdapat algoritma pelatihan yang bisa digunakan, diantaranya adalah *Fletcher-Reeves Update* (traincgf), *Polak-Ribiere* (traincgp), *Powell-Beale Restarts* (traincgb), *Scaled Conjugate Gradient* (trainscg), *Gradien Descent* (traingd), *Gradient Descent* dengan *Adaptive Learning Rate* (traingda), *Gradient Descent* dengan *momentum* (traingdm), *Gradient Descent* dengan *Momentum* dan *Adaptive Learning Rate* (traingdx), *Levenberg-Marquardt* (trainlm), *One Step Secant (trainoss), BFGS* (trainbfg), *Resilent*

Backpropagation (trainrp), yang termasuk dalam metode *Backpropagation* (Kusumadewi, 2004).

F. ANOVA

ANAVA atau disebut juga ANOVA (*Analysis of Variancce*) merupakan perluasan dari uji rata – rata k sampel, dengan k > 2.

- H_o: $\mu_1 = \mu_2 = \cdots = \mu_k$ vs H_1: 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 Ho ditolak maka masih harus ditentukan *mean – mean* mana saja yang berbeda. Untuk itu perlu dilakukan MCA (*Multiple Comparison Analysis*) atau Analisis Pembandingan Ganda (Taniredja & Mustafidah, 2011).

G. SPSS

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. SPSS sudah dikembangkan 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 (Taniredja & Mustafidah, 2011).

H. Penelitian Terkait

Menurut penelitian Mustafidah & Suwarsito (2015a) yang telah dilakukan sebelumnya pengujian terhadap algoritma pelatihan yang terdapat dalam jaringan *backpropogation* yang sejumlah 12 algoritma. Tes untuk algoritma pelatihan dalam *backpropagation* dilakukan menggunakan parameter jaringan seperti target *error* = 0,001 (10-3), maksimum epoh = 10000, *learning rate* (lr) = 0,01, dengan 5 neuron *input* dan satu neuron *output*. Berdasarkan hasil tes dari 12 algoritma pelatihan, algoritma trainlm memiliki *error* terkecil dengan level α = 5% dan memberikan *error* 0,0001986858. Dengan demikian algoritma *trainlm* dapat dianggap sebagai algoritma pelatihan dalam *backpropagation* untuk aplikasi dalam menyelesaikan masalah.

Menurut penelitian Mustafidah & Suwarsito (2015b) telah dilakukan pengujian 12 algoritma pelatihan. Penelitian ini menghasilkan simpulan bahwa algoritma pelatihan *trainlm* merupakan algoritma pelatihan yang paling optimal pada tingkat laju pembelajaran (learning

rate/lr) = 0,05. Hal ini dibuktikan berdasarkan uji ANOVA dengan tingkat α = 5%, hasil uji menunjukkan tingkat signifikansi sebesar 0,00. Rata-rata *error* yang dihasilkan oleh algoritma *trainlm* ini adalah sebesar 0,0002196.

Menurut penelitian Mustafidah, et al. (2014) algoritma pelatihan 12 dalam jaringan saraf 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 *epoch* = 1000, tingkat pembelajaran = 0,05, dan kesalahan target = 0,001. Hasil pengujian diperoleh bahwa pada interval kepercayaan 95% algoritma trainlm adalah algoritma yang paling tepat dalam mengenali pola data dengan tingkat kelayakan rata-rata 87,5%. Dianjurkan untuk menguji algoritma pelatihan dalam mengenali pola data dalam kasus lain dari kategori kualitas tes yaitu reliabilitas, kekuatan perbedaan uji, dan tingkat kesulitan yang disediakan.