**BAB VII**

**JARINGAN SYARAF TIRUAN**

1. **Tujuan**

* Dapat memahami kecerdasan buatan mengenai Jaringan Syaraf Tiruan
* Mengenal berbagai jenis jaringan syaraf tiruan dan dapat mengimplementasikan jaringan syaraf tiruan khususnya *perceptron* menggunakan MATLAB

1. **PENDAHULUAN**

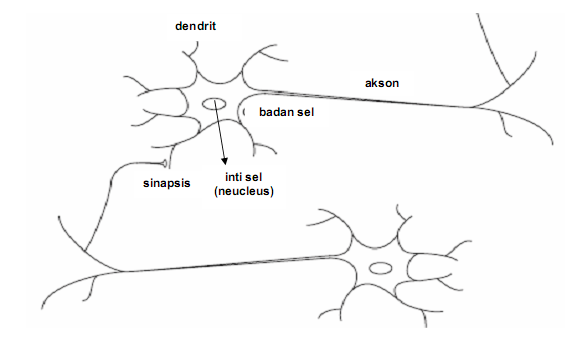
**Pengertian Jaringan Syaraf Tiruan**

Jaringan syaraf tiruan (*artifical neural network*) adalah sistem komputasi yang arsitektur dan operasinya diilhami dari pengetahuan tentang sel syaraf biologis di dalam otak. Jaringan syaraf tiruan merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba menstimulasi proses pembelajaran pada otak manusia tersebut. 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 koleksi model jaringan syaraf biologi.

Model jaringan syaraf ditunjukkan dengan kemampuannya dalam emulasi, analisis, prediksi dan asosiasi. Kemampuan 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 input yang diberikan kepada jaringan syaraf tiruan.

**Konsep Dasar Jaringan Syaraf Tiruan**

Jaringan syaraf tiruan terdiri dari beberapa *neuron* dan ada hubungan antar *neuron*-*neuron* seperti pada otak manusia. *Neuron*/sel syaraf adalah sebuah unit pemroses informasi yang merupakan dasar operasi jaringan syaraf tiruan.



**Gambar 1**. Ilustrasi *Neuron* (Sel Syaraf Manusia)

Jaringan syaraf tiruan terdiri atas beberapa elemen penghitung tak linier yang masing-masing dihubungkan melalui suatu pembobot dan tersusun secara paralel. Pembobot inilah yang nantinya akan berubah (beradaptasi) selama proses pelatihan.

Pelatihan perlu dilakukan pada suatu jaringan syaraf tiruan sebelum digunakan untuk menyelesaikan masalah. Hasil pelatihan jaringan syaraf tiruan adalah dapat diperolehnya tanggapan yang benar (yang diinginkan) terhadap masukan yang diberikan. Jaringan syaraf tiruan dapat memberikan tanggapan yang benar walaupun masukan yang diberikan terkena derau atau berubah oleh suatu keadaan.

**Karakteristik Jaringan Syaraf Tiruan**

Penyelesaian masalah dengan jaringan syaraf tiruan tidak memerlukan pemrograman. Jaringan syaraf tiruan menyelesaikan masalah melalui proses belajar dari contoh-contoh pelatihan yang diberikan. Biasanya pada jaringan syaraf tiruan diberikan sebuah himpunan pola pelatihan yang terdiri dari sekumpulan contoh pola. Proses belajar jaringan syaraf tiruan berasal dari serangkaian contoh-contoh pola yang diberikan.metode pelatihan yang sering dipakai adalah metode belajar terbimbing. Selama proses belajar itu pola masukan disajikan bersama-sama dengan pola keluaran yang diinginkan. Jaringan akan menyesuaikan nilai bobotnya sebagai tanggapan atas pola masukan dan sasaran yang disajikan tersebut.

**Faktor Bobot**

Bobot merupakan suatu nilai yang mendefinisikan tingkat atau kepentingan hubungan antara suatu *node* dengan *node* yang lain. Semakin besar bobot  suatu hubungan menandakan semakin pentingnya hubungan kedua *node* tersebut.

Bobot merupakan suatu hubungan berupa bilangan *real* maupun *integer*, tergantung dari jenis permasalahan dan model yang digunakan. Bobot-bobot tersebut bisa ditentukan untuk berada didalam interval tertentu. Selama proses pelatihan, bobot tersebut dapat menyesuaikan dengan pola-pola input.

Jaringan dengan sendirinya akan memperbaiki diri terus-menerus karena adanya kemampuan untuk belajar. Setiap ada suatu masalah baru, jaringan dapat belajar dari masalah baru tersebut, yaitu dengan mengatur kembali nilai bobot untuk menyesuaikan karakter nilai.

**Fungsi Aktivasi**

Setiap neuron mempunyai keadaan internal yang disebut level aktivasi atau level aktivitas yang merupakan fungsi input yang diterima. Secara tipikal suatu *neuron* mengirimkan aktivitasnya kebeberapa *neuron* lain sebagai sinyal. Yang perlu diperhatikan adalah bahwa *neuron* hanya dapat mengirimkan satu sinyal sesaat, walaupun sinyal tersebut dapat dipancarkan ke beberapa *neuron* yang lain.

***Perceptron***

*Perceptron* merupakan model jaringan syaraf tiruan yang ditemukan oleh Rosenblatt (1962) dan Minsky – Papert (1969). *Perceptron* merupakan model dengan aplikasi dan memiliki sistem pelatihan yang paling baik pada saat itu. Selain itu, *perceptron* juga merupakan model paling sederhana yang dapat diselesaikan dengan MATLAB.

Pada MATLAB, *default* sistem yang digunakan untuk membentuk *perceptron* adalah sebagai berikut :

* Input dan target dapat dibentuk secara bebas (tidak harus *biner*/*bipolar*)
* *Threshold* yang dipakai adalah 0
* Fungsi aktifasi memiliki *output biner* yang dikenal dengan nama *hardlim*. Dengan model *network*
* Bobot dapat diubah berdasarkan *error* yang terbentuk dari selisih antara target dengan *output* jaringan *f*(*net*).
* Perubahan bobot dilakukan hanya jika target tidak sama dengan *output* jaringan.

1. **LANGKAH PRAKTIKUM**

**Pembuatan *Perceptron***

Dalam membuat sebuah *network*, pertama-tama harus dibuat sebuah *perceptron* yang pada MATLAB bisa dilakukan dengan menggunakan perintah newp. Perintah tersebut akan membentuk sebuah perceptron dengan spesifikasi jumlah *input*, jumlah *neuron*, fungsi aktifasi yang secara lengkap memiliki format :

|  |
| --- |
| net= newp (PR,S,TF,LF) |

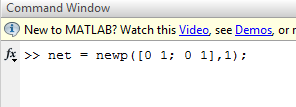
**Listing 1.** Perintah Untuk Membentuk *Perceptron*.

Dengan keterangan :

|  |  |  |
| --- | --- | --- |
| *PR* | : | Merupakan matriks dengan ordo Rx2 yang menyatakan nilai minimum dan maksimum dari tiap unit *input*. (terdapat R buah unit *input*) |
| *S* | : | Jumlah *neuron* target |
| *TF* | : | Fungsi aktifasi (*default* : fungsi *threshold* yang pada MATLAB disebut *hardlim*) |
| *LF* | : | Fungsi pelatihan (*default*: *Learnp*) |

Sebagai contoh:

Untuk membuat *perceptron* yang dapat mengenali pola fungsi logika “*and*” dengan dua (2) variabel x1 dan x2 adalah sebagai berikut :



**Gambar 2.** Membuat Perceptron.

Pada fungsi logika “*and*”, nilai *input* yang bisa digunakan adalah 0 dan 1 serta target yang dihasilkan hanya satu buah target. Sehingga pada baris pertama matriks [0 1] menunjukkan nilai yang bisa dimasukkan untuk x1 dan [0 1] yang kedua setelah tanda (;) merupakan nilai yang bisa dimasukkan untuk x2. Berikutnya nilai S=1 menunjukkan bahwa *perceptron* yang dibentuk tersebut hanya memiliki sebuah target yang pada MATLAB dikenal dengan istilah *neuron*.

**Pembentukan Input dan Target**

Setelah *perceptron* dibangun, menggunakan perintah newp, berikutnya adalah mendefinisikan pola *input* serta target yang dikenali. Masing-masing *input* dan *output* berupa vektor kolom. Hal yang perlu diperhatikan adalah bahwa jumlah dan *range* nilai untuk *input* harus sesuai dengan *perceptron* yang dibuat.

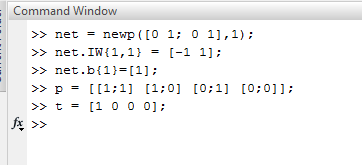
Sebagai Contoh :

Untuk membuat *perceptron* yang dapat mengenali pola fungsi logika “*and*” menggunakan dua (2) variabel x1 dan x2 dengan bobot awal *w* = [-1, 1] dan bias *b* = [1]. Maka yang perlu dilakukan adalah sebagai berikut:

**Tabel 1.** Tabel Pola Input Logika “And”

|  |  |  |
| --- | --- | --- |
| Pola Input | Input | Target |
| P1 |  | 1 |
| P2 |  | 0 |
| P3 |  | 0 |
| P4 |  | 0 |

Sehingga pada MATLAB dapat dituliskan sebagai berikut :

****

**Gambar 3.** Perintah untuk menambahkan bobot awal dan bias pada *perceptron*

**Menghitung *Output***

Setelah pola input dan target diberikan, berikutnya dapat dihitung *output network* nya menggunakan perintah sim pada MATLAB. Format perintah sim adalah sebagai berikut :

|  |
| --- |
| [Y,Pf,Af,E,perf] = sim[net,P,Pi,Ai,T] |

**Listing 2.** Format perintah untuk menghitung hasil *perceptron*

Dengan keterangan parameter untuk masukan :

|  |  |  |
| --- | --- | --- |
| *Net* | : | Nama jaringan *perceptron* yang dibentuk dengan perintah *newp* |
| *P* | : | Vektor input pada jaringan |
| *Pi* | : | Kondisi *delay* awal masukan (*default*: *zeros* atau []) |
| *Ai* | : | Kondisi *delay* layar (*default*: *zeros* atau []) |
| *T* | : | Vektor target jaringan (*default*: *zeros* atau []) |

Sedangkan keterangan untuk parameter hasil :

|  |  |  |
| --- | --- | --- |
| *Y* | : | Output Jaringan |
| *Pf* | : | Kondisi Akhir *Delay Input* |
| *Af* | : | Kondisi Akhir *Delay Layar* |
| *E* | : | *Error* Jaringan (T-Y) |
| *perf* | : | Performansi (unjuk kerja) jaringan |
|  |  |  |

Dari format sim yang bisa digunakan untuk menghitung *output*, *Pi*, *Ai*, *Pf*, dan *Af* hanya digunakan ketika jaringan memiliki *delay*, sehingga jika tidak diketahui nilai *delay* untuk jaringan tersebut dan hanya digunakan untuk menghitung nilai keluran nya saja bisa menggunakan *statemen* sim secara sederhana yaitu :

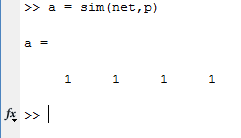
|  |
| --- |
| y = sim(net,p) |

**Listing 3.** *Statement* sederhana untuk menghitung hasil keluaran *perceptron*.

Dari *statemen* tersebut yang perlu menjadi perhatian adalah ketika akan menghitung keluaran jaringan, maka tidak perlu tahu apa *target* yang diperlukan, akan tetapi jika ingin dihitung seberapa besar kesalahan (*error*) yang terjadi pada hasil yang sudah dihitung. Maka harus diketahui targetnya.

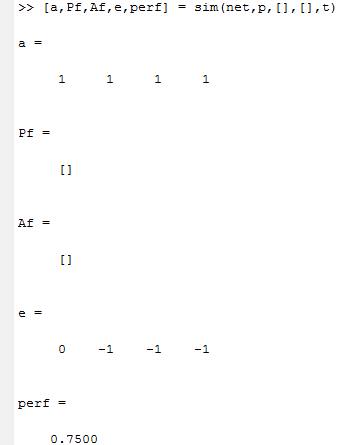
Sebagai contoh:

Untuk menampilkan hasil *output* jaringan pada *perceptron* untuk mengenali pola “*and*” yang menggunakan dua (2) variabel x1 dan x2 dengan bobot awal w = [-1, 1] dan bias b=[1]. Adalah sebagai berikut :



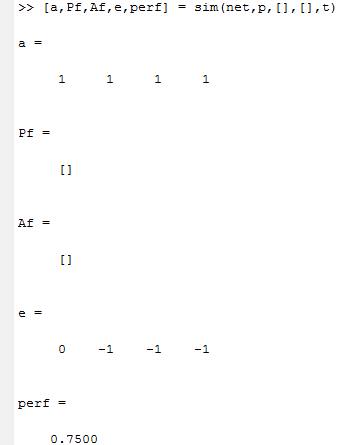
**Gambar 4.** Output perceptron tanpa memperdulikan target.

Pada perintah a=sim(net,p) *output* yang ditampilkan hanya hasil keluaran jaringan *perceptron* berdasarkan bobot dan bias yang sudah dibentuk. Apabila ingin mengetahui besarnya *error* dan performansi dari jaringan tersebut maka parameter untuk *statement* sim akan ditambah menjadi sebagai berikut :



**Gambar 5.** *Statement* sim untuk mengetahui *error* dan performansi jaringan

Sehingga memberikan hasil sebagai berikut :



**Gambar 6.** *Output* jaringan yang dihasilkan *perceptron* untuk logika “*and*”.

Dari pola tersebut hanya pola pertama yang dikenali dengan benar. Hal tersebut dibuktikan dengan nilai *a*[*1*] = 0 yang menandakan *threshold* sudah bernilai benar. Untuk ketiga pola lain masih belum dikenali sehingga performa yang dihasilkan adalah ¾ = 0.75.

**Modifikasi Bobot dan Bias**

Setelah *output* jaringan dihitung, berikutnya adalah mengubah bobot berdasarkan selisih antara *output* jaringan dengan target yang diinginkan. Perintah untuk mengubah bobot pada MATLAB adalah learnp dengan format :

|  |
| --- |
| dW = learnp(W,P,Z,N,A,T,E,gW,gA,D,LP,LS) |

**Listing 4.** statement learnp untuk mengubah bobot

Dengan keterangan :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *W* | : | Matriks Bobot | *E* | : | Vektor Layar Error |
| *P* | : | Vektor Input | *gW* | : | Gradien Input Terhadap Performansi |
| *Z* | : | Vektor Input dengan Bobot | *gA* | : | Gradien Output Terhadap Performansi |
| *N* | : | Vektor Masukan Net | *D* | : | Jarak Neuron |
| *A* | : | Vektor Output | *LP* | : | Parameter Pemahaman |
| *T* | : | Vektor Layar Target | *LS* | : | State Pemahaman |

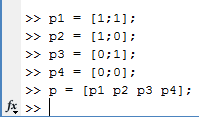
Pada perceptron sederhana, parameter yang perlu diberikan hanyalah *W*, *P*, dan *E* yang didapat dari sim(net,p)-t.

Sebagai contoh :

Untuk menghitung perubahan bobot setelah menghitung *output* pada pola “*and*” yang menggunakan dua (2) variabel x1 dan x2 dengan bobot awal *w* = [-1, 1] dan bias *b* = [1].

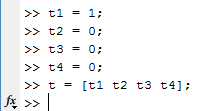
Penyelesaian dari kasus berikut :

1. Definisikan pola input yang akan dihitung



**Gambar 7.** Definisi pola input yang akan dihitung dalam tiap variabel

1. Definisikan target



**Gambar 8.** Definisi Target dalam setiap variabel

1. Buat *perceptronnya*



**Gambar 9.** Pembentukan Perceptron

1. Definisikan bobot awalnya



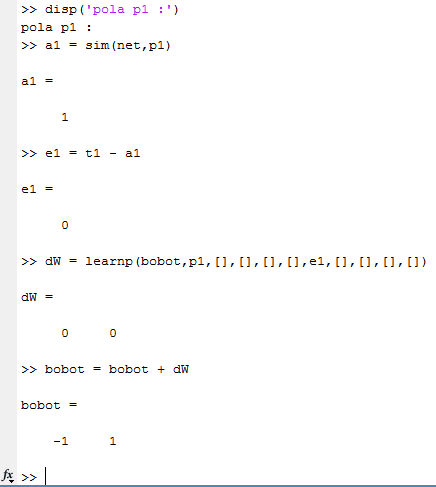
**Gambar 10.** Pendefinisian bobot awal.

1. Definisikan nilai bias awalnya



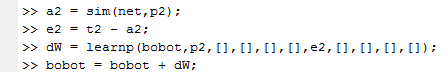
**Gambar 11.** pendefinisan bias awal

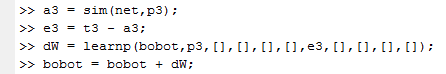
1. Hitung keluaran *perceptron* pada setiap pola *input* dan tampilkan besarnya perubahan bobot.

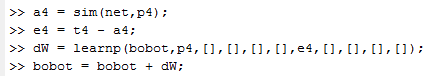


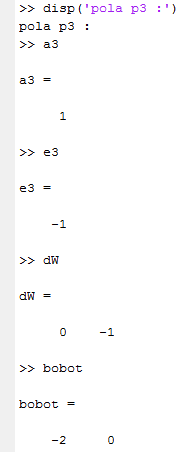
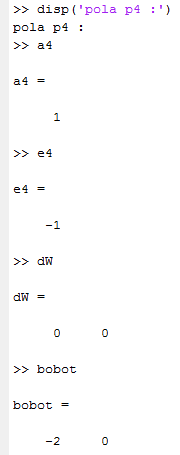
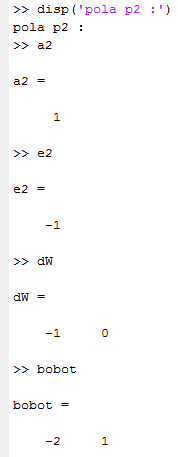
**Gambar 12.** hasil perubahan bobot

Dari perintah learnp yang digunakan akan menyimpan besarnya perubahan bobot pada variable dW. Bobot baru diperoleh dari hasil dW yang ditambahkan dengan vektor bobot. Karena output network = target, maka tidak dilakukan peruibahan bobot yang membuat bobot baru = bobot lama. Akan tetapi apabila proses tersebut dilakukan pada pola p2, p3 dan p4. Hasilnya adalah sebagai berikut :







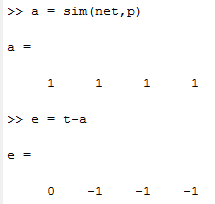


**Gambar 13.** Hasil Pelatihan p2, p3 dan p4.

Pada pola ke 2 dan pola ke 3, e = t-a = -1 sehingga dW = -p. Perubahan ini ditambahkan pada vektor bobot. Dan ketika pola keempat, e = 0 sehingga tidak ada perubahan bobot.

**Pelatihan *Perceptron***

*Perceptron* akan melakukan perubahan bobot secara terus menerus untuk setiap pola yang diberikan hingga mengenali semua pola secara benar. Perubahan bobot untuk setiap pola yang diberikan disebut dengan *epoch*. Pada kasus yang sudah dikerjakan, bobot terakhir setelah *epoch* adalah [-2 0] dengan bias [-1]. Akan tetapi, bobot tersebut belum mampu mengenali semua pola dengan benar. Hal tersebut bisa di cek dengan perintah *sim*.



**Gambar 14.** *output* pengujian dengan bobot setelah *epoch*.

Dari hasil tersebut, pola yang muncul masih [1 1 1 1] sedangkan target yang diinginkan adalah [1 0 0 0] dan terdapat kesalahan e = [0 1 1 1]. Hal tersebut menunjukkan perlu adanya pelatihan lagi untuk *epoch* berikutnya hingga nilai e = 0 yang tentu saja membutuhkan program yang panjang apalagi jika pola pelatihannya banyak.

Sehingga untuk menyingkat proses tersebut, MATLAB menyediakan perintah train dengan format sebagai berikut :

|  |
| --- |
| [net,tr,Y,E,Pf,Af] = train(net,P,T,Pi,Ai,VV,TV) |

**Listing 5.** Perintah untuk training *perceptron*.

Dengan keterangan :

|  |  |  |
| --- | --- | --- |
| *net* | : | Jaringan yang didefinisikan dalam *newp* |
| *P* | : | *Input* Jaringan |
| *T* | : | Target Jaringan |
| *Pi* | : | Kondisi *Delay* awal *input* |
| *Ai* | : | Kondisi *Delay* awal Layar |
| *VV* | : | Struktur validasi vektor |
| *TV* | : | Struktur vektor uji |

Dan perintah train akan menghasilkan :

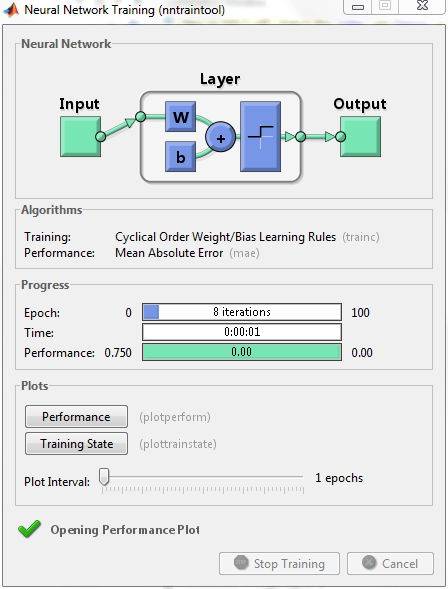
|  |  |  |
| --- | --- | --- |
| *net* | : | Jaringan yang baru |
| *tr* | : | *Record* pelatihan (*epoch* dan performansi) |
| *Y* | : | Keluaran Jaringan |
| *E* | : | *Error* Jaringan |
| *Pf* | : | Kondisi Akhir *Delay* Masukan |
| *Af* | : | Kondisi Akhir *Delay* layar |

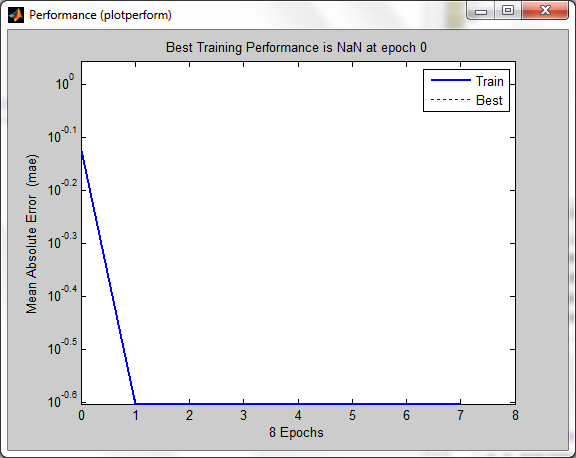
Sehingga untuk menyingkat proses yang dilakukan untuk mengetahui bobot setelah menghitung *output* pada pola “*and*” yang menggunakan dua (2) variabel x1 dan x2 dengan bobot awal *w* = [-1, 1] dan bias *b* = [1]



**Listing 6.** Perintah untuk menjalankan pelatihan *perceptron*.

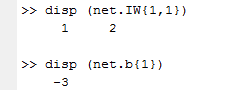
Yang memberikan hasil :





**Gambar 15**. Hasil pelatihan *perceptron* menggunakan *statement* *train*

Dari grafik tersebut menunjukkan bahwa iterasi diselesaikan dalam 8 *epoch* yang sudah memiliki nilai 0. Kemudian berdasarkan keseluruhan hasil tersebut tampilan bobot dan bias dalam keadaan optimal dapat dilihat menggunakan.



**Gambar 16.** Menampilkan nilai bobot dan bias

Dari hasil tersebut, diperoleh bobot w1 = 1 dan w2 = 2 serta bias b=-3.

1. **TUGAS PRAKTIKUM**

Buatlah *perceptron* untuk mengenali pola fungsi logika “*or*” menggunakan dua (2) variabel x1 dan x2 dengan bobot awal *w* = [-1, 1] dan bias *b*=[1]. Serta tampilkan bobot dan bias optimalnya menggunakan MATLAB.