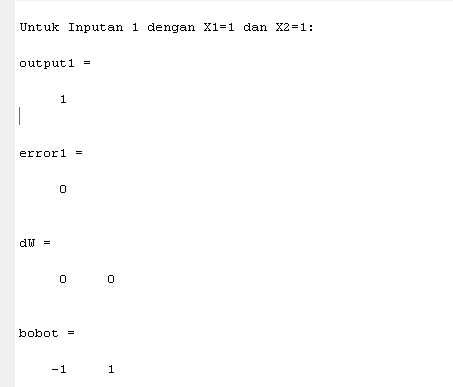


DItampilkan inputan apa saja yang nantinya dapat digunakan oleh variabel X1 dan X2 dalam bentuk matriks 2x4 yang disimpan dalam variabel p

Kita buat fungsi perceptron dengan menggunakan fungsi, dapat dilakukan di awal program atau setelah deklarasi inputan dan target:

|  |
| --- |
| net = newp([0 1;0 1],1); |



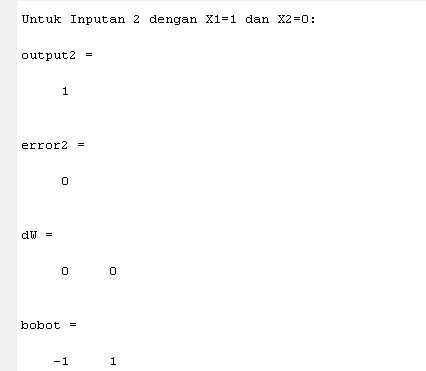
Dilakukan perubahan bias dan bobot satu per satu inputan dengan fungsi

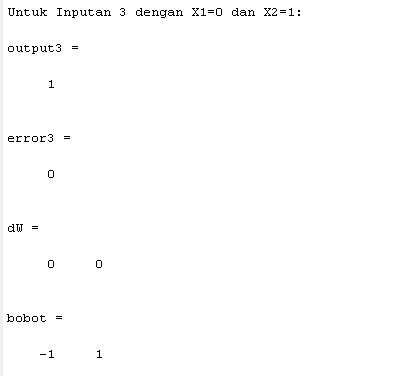
|  |
| --- |
| output1 = sim(net,p1)  error1 = t1 - output1  % Nilai dW didapat dari e\*p  dW = learnp(bobot,p1,[],[],[],[],error1,[],[],[],[],[]) |

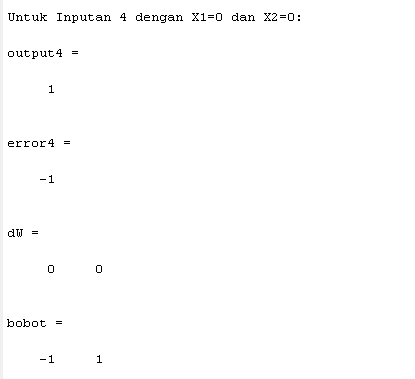
learnp akan menyimpan perubahan bobot ke dalam variabel dW, kemudian untuk mendapatkan bobot setelah perubahan maka digunakan fungsi selanjutnya

|  |
| --- |
| bobot = bobot + dW |

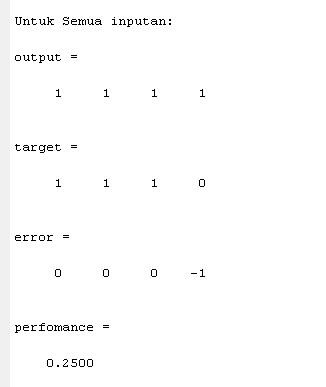
Lakukan hingga semua inputan , hingga diketahui bobot akhir







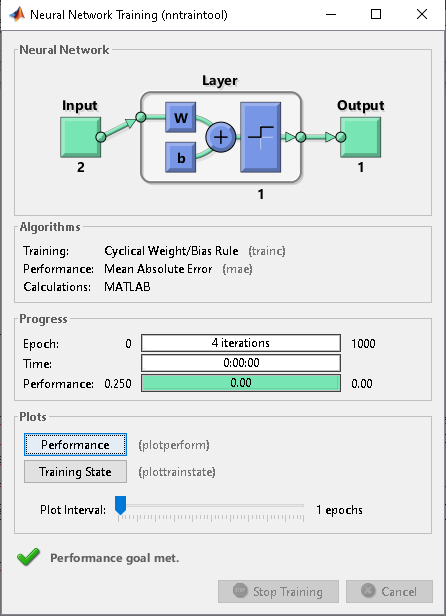
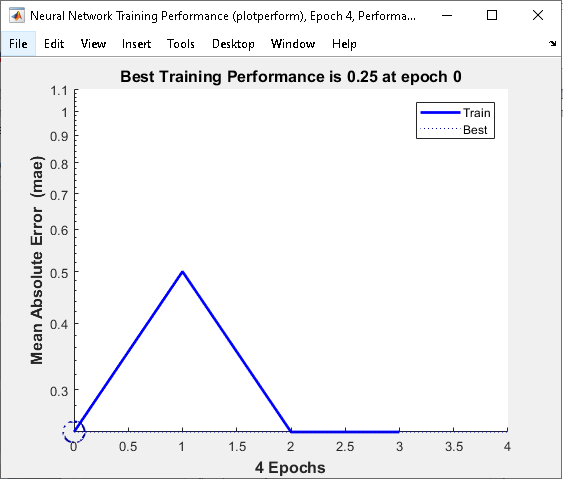
Proses ini dilakukan berulang hingga error sudah = 0



Berikut merupakan hasil output dari hasil JST, ternyata target dan output yang dihasilkan belum sesuai. Nilai output = [1 1 1 1 ], padahal hasil dari pola “and” yang benar adalah [1 1 1 0]. Dari hasil error diketahui bahwa pada inputan ke 4, hasil tidak sesuai sehingga menghasilkan error, dari situ diketahui pula performa yang dihasilkan adalah 0.25. Agar nilai error mencapai 0 maka diperlukan pelatihan kembali pada epoch-epoch (perubahan bobot) selanjutnya, pelatihan dapat dilakukan secara manual dengan menggunkan fungsi sebelumnya, yaitu learnp namun akan memakan banyak waktu oleh karena itu digunakan fungsi matlab, yaitu seperti di bawah:

|  |
| --- |
| net = train(net,p,t); |

Yang menghasilkan



Dari sini diketahui bahwa diperlukan 4 epoch untuk menyelesaikan kasus “and” hingga nilai error bernilai 0.

Kemudian dapat kita lihat bobot dan bias optimal menggunakan fungsi

|  |
| --- |
| disp(net.IW{1,1})  disp(net.b{1}) |

Sehingga terlihat seperti di bawah

