FNN_To_Equation Function

BENALLAL Mohamed Anis

May 25, 2016

Abstract

Here we present a function which transform a FeedForward Neural Network (FNN) object constructed with MATLAB^{TM} into an equation using it's parameters.

1 Source Code

```
function [resu,v,ai,bi,ci,di,Range_Input,Range_Output] = FNN_To_Equation(net)
% using a standard FNN with the Levenberg-Marquardt backpropagation algorithm
% 2 neurons in the input layer & one output neuron and one hidden layer
%Get all parameters needed for the equation
IW = net.IW{1,1} ;
b1 = net.b{1};
b2 = net.b{2};
LW = net.LW{2,1};
InputsParam=net.inputs{1};
PS_Input=InputsParam.processSettings{1,1};
Range_Input=net.inputs{1}.range;
 % | h | g |
 % | l | k |
 % e = 1, f = -1;
OutputParam = net.outputs{2};
 % | o (x_min) | n (x_max) |
 % m= -1 (y_min), p=1 (y_max)
PS_Output = OutputParam.processSettings{1,1};
Range_Output=net.outputs{2}.range;
```

```
v = num2str(b2);
%Construction of the equation
%All equation coefficient are  :
% v ai(i=1..39) bi(i=1..39) ci(i=1..39) di(i=1..39)
% e f g h k l m n o p

resu = ['OUTPUT = (v'] ; ai=''; bi=''; ci=''; di='';

for i=1:size(IW,1)
 resu = [resu,' + a',num2str(i),' * tansig(b',num2str(i),...
       ' * ( (e-f)*(INPUT1-h)/(g-h) + f) + c',num2str(i),...
       ' * ( (e-f)*(INPUT2-l)/(k-l) + f) + d',num2str(i),')'];
 ai = [ai,'a',num2str(i),'=',num2str(LW(i)),'; '];
 bi = [bi,'b',num2str(i),'=',num2str(IW(i,1)),'; '];
 ci = [ci,'c',num2str(i),'=',num2str(IW(i,2)),'; '];
 di = [di,'d',num2str(i),'=',num2str(b1(i)),'; '];
end
resu = [resu,' - m)*(n-o)/(p-m) + o'];

end
```

2 Function description

The output function can written as :

$$\begin{aligned} fCO_2^{eau} &= (v + \sum_{i=1}^{j} a_i * tansig(b_i * ((e-f) * (\mathrm{SST} - h)/(g-h) + f) \\ &+ c_i * ((e-f) * (\mathrm{Chl}_a - l)/(k-l) + f) + d_i) \\ &- m) * (n-o)/(p-m) + o \end{aligned} \tag{1}$$

3 Execution outputs example

Here we present an example with 10 hidden neurons in the hidden layer, the code works for a FNN with two input neurons and one output neuron but can be easily adapted to other cases, the only things is that you should have only one hidden layer.

- resu :
  $OUTPUT = (v + a1 * tansig(b1 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c1 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d1) + a2 * tansig(b2 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c2 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d2) + a3 * tansig(b3 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c3 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d3) + a4 * tansig(b4 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c4 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d4) + a5 * tansig(b5 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c5 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d5) + a6 * tansig(b6 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c6 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d6) + a7 * tansig(b7 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c7 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d7) + a8 * tansig(b8 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c8 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d8) + a9 * tansig(b9 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c9 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d9) + a10 * tansig(b10 * ((e - f) * (INPUT1 - h)/(g - h) + f) + c10 * ((e - f) * (INPUT2 - l)/(k - l) + f) + d10) - m) * (n - o)/(p - m) + o$

- a_i :
  $a1 = -0.13128; a2 = 0.12554; a3 = -0.44828; a4 = -11.2318; a5 = 0.08937; a6 = -0.99928; a7 = 0.71418; a8 = 0.12992; a9 = -11.9895; a10 = 0.80048;$

- b_i :
  $b1 = -7.0944; b2 = 63.8881; b3 = -8.0632; b4 = 5.2339; b5 = -0.76284; b6 = 1.2431; b7 = -11.8165; b8 = -52.0118; b9 = -4.97; b10 = 4.5507;$

- c_i :
  $c1 = 28.939; c2 = -60.8016; c3 = -3.2353; c4 = -1.6614; c5 = 58.532; c6 = 0.21424; c7 = -11.709; c8 = -54.0334; c9 = 1.9745; c10 = 7.1484;$

- d_i :
  $d1 = 25.721; d2 = -35.8273; d3 = 6.4796; d4 = 2.3841; d5 = 24.0724; d6 = 0.0852; d7 = -19.5887; d8 = -70.628; d9 = -2.006; d10 = 9.0735;$

- v :
  $v = 0.10312$

- Range_Input :
  $Range_{Input} =$

$|-1,17800000000000|18,1660000000000|$
$|0,0432600006461144|1,47134995460510|$

- Range_Output :
  $Range_{Output} = |285,180000000000|401,270000000000|$