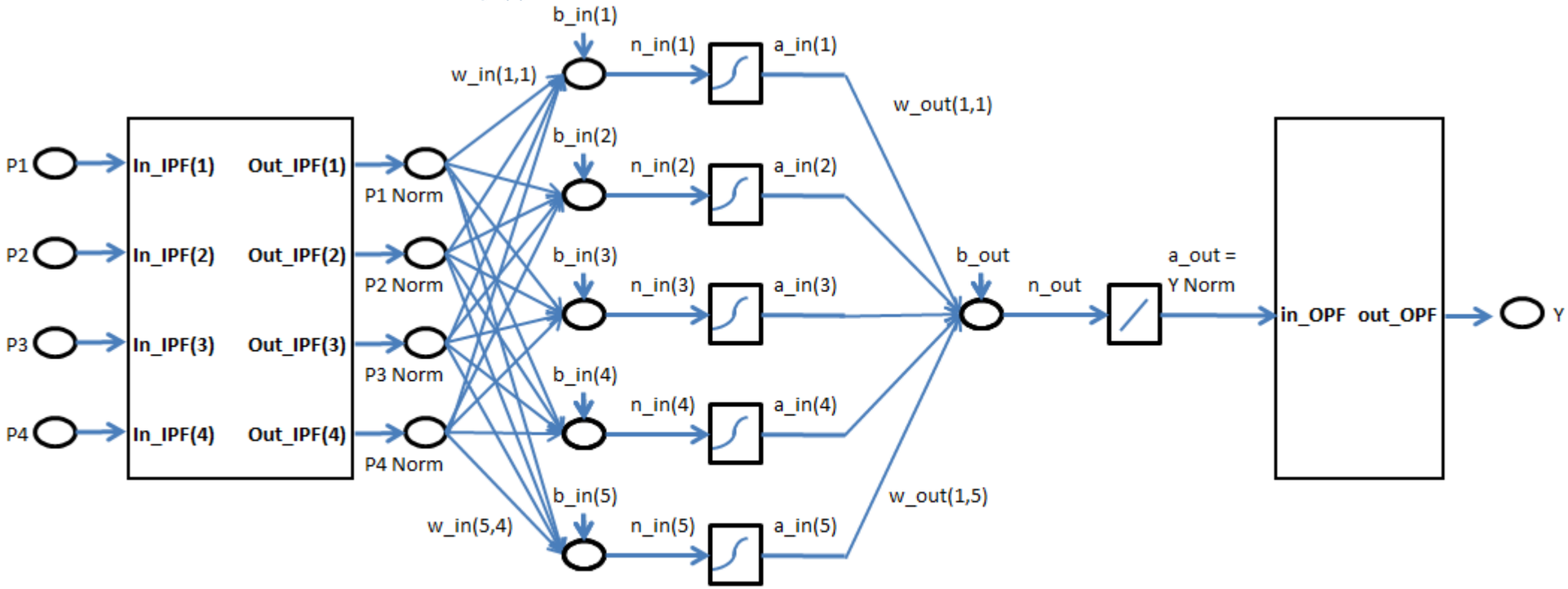
1. NEURAL NETWORK ARCHITECTUR$f(.)$



DATA INPUT

INPUT PROCESSING FUNCTION

OUTPUT PROCESSING FUNCTION

2. Input and Output Processing Functions.

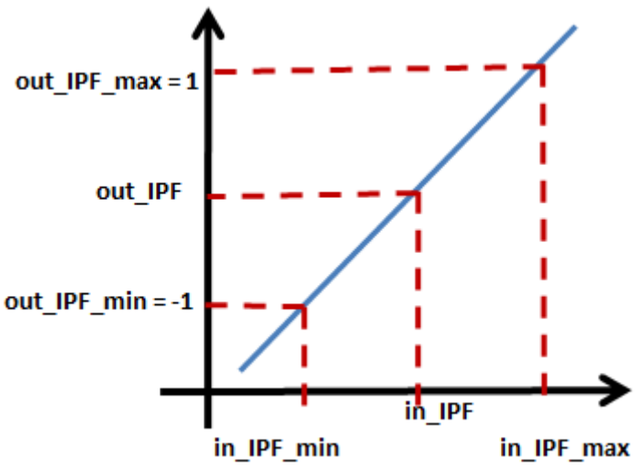
"Input Processing Functions" transforms input data so that all values fall into the interval [-1,1], or normalizes the input values. Training is often faster when values are normalized.

And if "Input Processing Functions" is used to scale the targets, then the output of the network will be trained to produce outputs in the range [-1,1]. To convert these output back into the original units, we use "Output Processing Function".

Example ilustration of "Input and Output Processing Functions" with the function.

INPUT PROCESSING FUNCTIONS :

"Input Processing Functions" transforms input data so that all values fall into the interval [-1,1], or normalizes the input values.

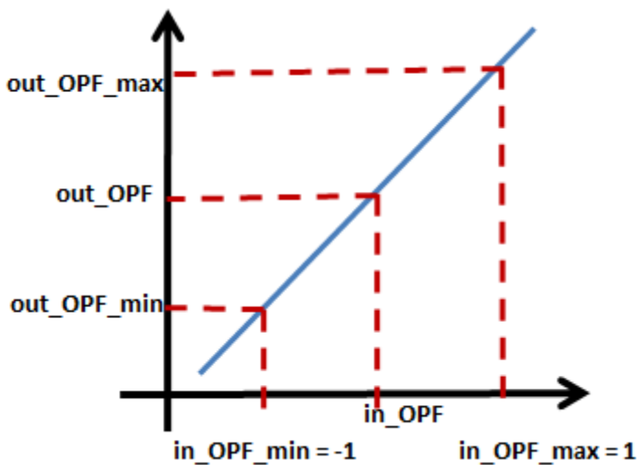


$$PNorm = \frac{(P - in\_IPF\_min) \cdot (out\_IPF\_max - out\_IPF\_min)}{(in\_IPF\_max - in\_IPF\_min)} + out\_IPF\_min$$

$$= \frac{(in\_IPF - in\_IPF\_min) \cdot (2)}{(in\_IPF\_max - in\_IPF\_min)} - 1$$

OUTPUT PROCESSING FUNCTIONS :

If "Input Processing Functions" is used to scale the targets, then the output of the network will be trained to produce outputs in the range [-1,1]. To convert these output back into the original units, we use "Output Processing Function".



$$out\_OPF = \frac{(in\_OPF - in\_OPF\_min) \cdot (out\_OPF\_max - out\_OPF\_min)}{(in\_OPF\_max - in\_OPF\_min)} + out\_OPF\_min$$

$$= \frac{(in_{OPF} + 1) \cdot (out\_OPF\_max - out\_OPF\_min)}{2} + out\_OPF\_min$$

3. EXAMPLE CALCULATION

From the above architecture :

Input Processing Function Parameter : In1_max=7.9 In1_min=4.3 In2_max=4.4 In2_min=2 In3_max=6.9 In3_min=1 In4_max=2.5 In4_min=0.1	w_in(1,1)= -0.743495 w_in(1,2)= 0.449171 w_in(1,3)= -0.585676 w_in(1,4)= -0.309007 w_in(2,1)= 0.437339 w_in(2,2)= 0.21939 w_in(2,3)= -0.869233 w_in(2,4)= 0.774321 w_in(3,1)= -0.556053 w_in(3,2)= 0.387743 w_in(3,3)= 0.791078 w_in(3,4)= 0.212789 w_in(4,1)= 0.546645 w_in(4,2)= -0.274682 w_in(4,3)= -0.857102 w_in(4,4)= 0.181313 w_in(5,1)= -0.43569 w_in(5,2)= -0.170569 w_in(5,3)= 0.301563 w_in(5,4)= 0.971447	b_in(1) = 0.537452 b_in(2) = -0.901691 b_in(3) = -0.55688 b_in(4) = -0.860293 b_in(5) = -0.212749 w_out(1,1) = -0.597222 w_out(1,2) = -0.67192 w_out(1,3) = 0.496553 w_out(1,4) = -0.400212 w_out(1,5) = 0.801744 b_out = -0.187959	Output Processing Function Parameter : Out_max = 3 Out_min = 1

If we choose
P1=5.9
P2=3
P3=5.1
P4=1.8

$$P\ Norm = \frac{(P1 - in1\_min) \cdot (2)}{(in1\_max - in1\_min)} - 1$$

P1 Norm = -0.111111
P2 Norm = -0.166667
P3 Norm = 0.38983
P4 Norm = 0.416667

$$n\_in(i) = w\_in(i,1) \cdot P1Norm + w\_in(i,2) \cdot P2Norm + w\_in(i,3) \cdot P3Norm + w\_in(i,4) \cdot P4Norm + b\_in(i)$$

n_in(1) = 0.188133
n_in(2) = -1.00307
n_in(3) = -0.162672
n_in(4) = -1.13383
n_in(5) = 0.386417

$$a\_in(j) = f(n\_in(j))$$

$$a_{in(i)} = \frac{2}{1 + \exp(-2 \cdot n_{in(i)})} - 1$$

a_in(1) = 0.185945
a_in(2) = -0.76288
a_in(3) = -0.161252
a_in(4) = -0.812326
a_in(5) = 0.368267

$$n\_out(k) = a\_in(1) \cdot w\_out(k,1) + a\_in(2) \cdot w\_out(k,2) + a\_in(3) \cdot w\_out(k,3) + a\_in(4) \cdot w\_out(k,4) + a\_in(5) \cdot w\_out(k,5) + b\_out(k)$$

$$a\_out(k) = f(n\_out(k))$$

n_out = 0.753875

$Y\ Norm = n\_out$ (activation function : linear)

Y Norm = 0.753875

$$Y = \frac{(Y\_Norm + 1) \cdot (out\_max - out\_min)}{(2)} + out\_min$$

Y = 2.75387

4. Learning Equation

Backpropagation Algorithm

1. Initialization: Randomize the weights to small values.
2. Presentation: Apply a pattern to the input and calculate the network output.
3. Error Computation: Compare the output with the desired output and compute the error (difference).
4. Backward Computation: Backpropagate the error through the network and adjust the weights to minimize the error.
5. Iteration: Repeat steps 2-4 until a desired error goal is reached.

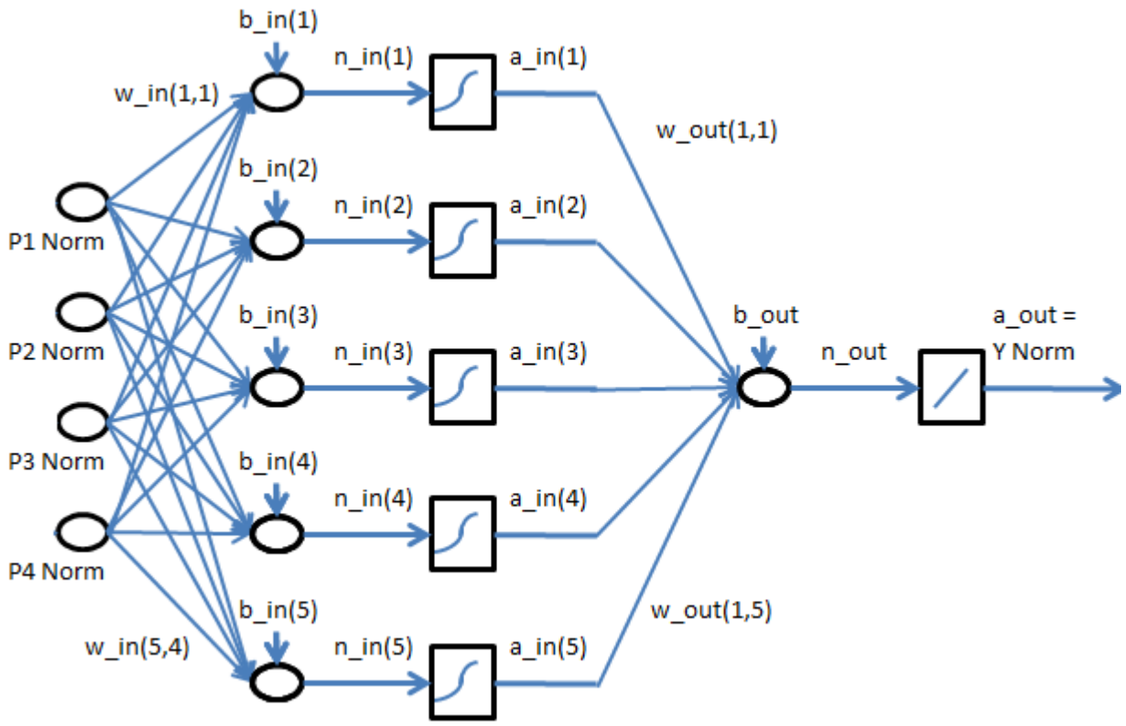
- Total Error Energy : $E = \frac{1}{2}e^2$
- Error Signal at iteration (n) = $e(n) = TNorm(n) - YNorm(n)$
  YNorm : keluaran NN yang masih ternormalisasi (rentang nilai antara -1 s.d. 1)
  Tnorm : target keluaran NN
- Backward Computation :

$$W^{new} = W^{old} + \Delta W$$

$$W^{new} = W^{old} - \eta \frac{dE}{dW}$$

$$W^{new} = W^{old} - \eta \left(\frac{dE}{de}\right) \cdot \left(\frac{de}{dYNorm}\right) \cdot \left(\frac{dYNorm}{dn\_out}\right) \cdot \left(\frac{dn\_out}{dW}\right)$$

$$W^{new} = W^{old} - \eta \cdot e \cdot (-1) \cdot (1) \cdot \left(\frac{dn\_out}{dW}\right)$$



PERSAMAAN NN YANG DIGUNAKAN

$$n\_in(i) = w\_in(i,1) \cdot P1Norm + w\_in(i,2) \cdot P2Norm + w\_in(i,3) \cdot P3Norm + w\_in(i,4) \cdot P4Norm + b\_in(i)$$

$$a_{in(i)} = \frac{2}{1 + \exp(-2 \cdot n_{in(i)})} - 1$$

$$n\_out = a\_in(1) \cdot w\_out(1,1) + a\_in(2) \cdot w\_out(1,2) + a\_in(3) \cdot w\_out(1,3) + a\_in(4) \cdot w\_out(1,4) + a\_in(5) \cdot w\_out(1,5) + b\_out$$

$$Y\ Norm = n\_out$$

MAKA PENURUNAN PERSAMAAN BACKWARD COMPUTATION :

$$b\_out^{new} = b\_out^{old} + \eta \cdot e$$

$$w\_out(1,j)^{new} = w\_out(1,j)^{old} + \eta \cdot e \cdot a\_in(j)$$

$$b\_in(j)^{new} = b\_in(j)^{old} + \eta \cdot e \cdot \left(\frac{dn\_out}{da\_in(j)}\right) \cdot \left(\frac{da\_in(j)}{dn\_in(j)}\right) \cdot \left(\frac{dn\_in(j)}{db\_in(j)}\right)$$

$$= b\_in(j)^{old} + \eta \cdot e \cdot (w\_out(1,j)) \cdot \left(\frac{4 \cdot \exp(-2 \cdot n\_in(j))}{(1 + \exp(-2 \cdot n\_in(j)))^2}\right) \cdot (1)$$

$$w\_in(j,i)^{new} = w\_in(j,i)^{old} + \eta \cdot e \cdot \left(\frac{dn\_out}{da\_in(j)}\right) \cdot \left(\frac{da\_in(j)}{dn\_in(j)}\right) \cdot \left(\frac{dn\_in(j)}{dw\_in(j,i)}\right)$$

$$= w\_in(j,i)^{old} + \eta \cdot e \cdot (w\_out(1,j)) \cdot \left(\frac{4 \cdot \exp\left(-2 \cdot n\_in(j)\right)}{\left(1 + \exp\left(-2 \cdot n\_in(j)\right)\right)^2}\right) \cdot PiNorm$$