Lecture Series on

Intelligent Control

Lecture 7

Artificial Neural Networks
Load Forecasting

Kwang Y. Lee
Professor of Electrical & Computer Engineering
Baylor University
Waco, TX 76706, USA
Kwang_Y_Lee@baylor.edu

1

Short-Term Load Forecasting Using an Artificial Neural Network

K. Y. Lee, Y. T. Cha, and J. H. Park

IEEE Trans. on Power Systems, Vol. 7. No. 1, pp. 124-132, February 1992

2

Abstract - Artificial Neural Network (ANN) Method is applied to forecast the short-term load for a large power system.

- The load has two distinct patterns: weekday and weekend-day patterns.
- The weekend-day pattern include Saturday, Sunday, and Monday loads.
- A nonlinear load model is proposed and several structures of ANN for short-term load forecasting are tested.
- Inputs to the ANN ***are*** past loads and the output of the ANN is the load forecast for a given day.

3

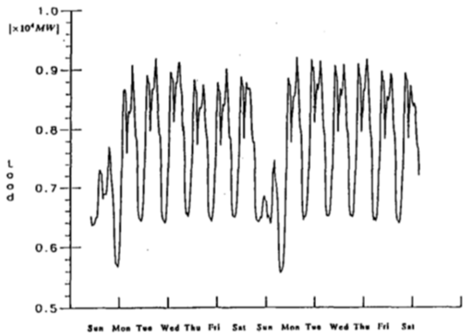


Fig. 1 Hourly load curve over two weeks

4

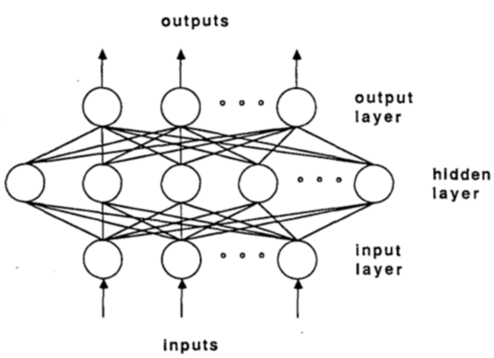
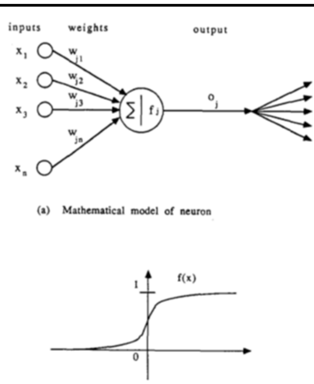


Fig. 2 Schematic of feedforward neural network

5



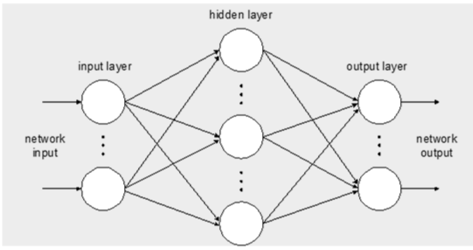
(a) Mathematical model of neuron

(b) Sigmoid function

Fig. 3 Schematic of an artificial neuron

6

Neural Network Terminology

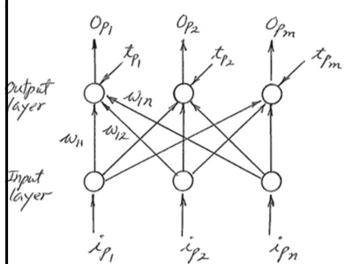


A Feedforward Multi-layer Network: each circle corresponds to a node and each arrow represents a weighted link.

7

Backpropagation

Delta Rule



p : pattern

i_p : input vector for pattern p

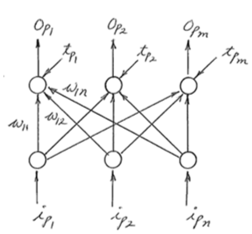
O_p : output vector for pattern p

t_p : target vector for pattern p

w_{ij} : weight from jth input to ith output component.

8

Delta Rule:



For a pattern p, the error is

$$\epsilon_p = \frac{1}{2} \sum_j (t_{p_j} - O_{p_j})^2 , \qquad \epsilon = \sum_p \epsilon_p$$

The gradient is, by chain rule,

$$\frac{\partial \epsilon_p}{\partial w_{ji}} = \frac{\partial \epsilon_p}{\partial O_{p_j}} \frac{\partial O_{p_j}}{\partial w_{ji}}$$

$$\frac{\partial \epsilon_p}{\partial O_{p_j}} = -(t_{p_j} - O_{p_j}) \triangleq -\delta_{p_j}$$

Assuming linear units,

$$O_{p_j} = \sum_i w_{ji} i_{p_i} \Rightarrow \frac{\partial O_{p_j}}{\partial w_{ji}} = i_{p_i}$$

Therefore, the negative gradient is

$$-\frac{\partial \epsilon_p}{\partial w_{ji}} = \delta_{p_j} i_{p_i}$$

$\therefore$ The weight w_{ji} can be updated by

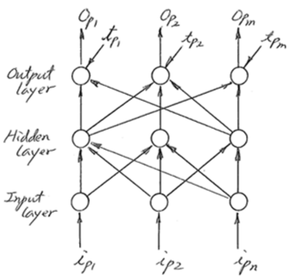
$$\Delta_p w_{ji} = \eta \, \delta_{p_j} \, i_{p_i} , \qquad \eta : \text{learning rate}$$

Since $\frac{\partial \epsilon}{\partial w_{ji}} = \sum_p \frac{\partial \epsilon_p}{\partial w_{ji}}$,

the Delta rule implements a gradient descent on E.

9

Generalized Delta Rule:



Now we add hidden layers (can be multiple).

Assume semilinear activation function:

$$O_{pj} = f_j(net_{pj}) \quad \text{- - - (1)}$$

$$net_{pj} = \sum_i w_{ji} O_{pi} \quad \text{-- (2)}$$

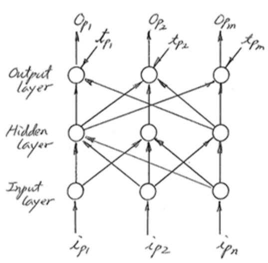
f_j: differentiable, nondecreasing

net_{pj}: weighted sum of inputs to jth neuron for pattern p.

Note: inputs to jth neuron are outputs of ith neuron in the previous layer.

10

Generalized Delta Rule:



$$O_{pj} = f_j(net_{pj}) \quad \text{- - - (1)}$$

$$net_{pj} = \sum_i w_{ji} O_{pi} \quad \text{-- (2)}$$

Define the error function:

$$E_p = \frac{1}{2}(t_{pj} - O_{pj})^2, \qquad E = \sum_p E_p$$

Then the gradient is

$$\frac{\partial E_p}{\partial w_{ji}} = \underbrace{\frac{\partial E_p}{\partial net_{pj}}} \; \underbrace{\frac{\partial net_{pj}}{\partial w_{ji}}}_{O_{pi} \text{ from (2)}}$$

The negative gradient is

$$\Rightarrow \; -\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} \, O_{pi} \quad \cdots (3)$$

11

Generalized Delta Rule:

$$E_p = \frac{1}{2}(t_{pj} - O_{pj})^2, \qquad E = \sum_p E_p$$

Then the gradient is

$$\frac{\partial E_p}{\partial w_{ji}} = \underbrace{\frac{\partial E_p}{\partial net_{pj}}} \; \underbrace{\frac{\partial net_{pj}}{\partial w_{ji}}}_{O_{pi} \text{ from (2)}}$$

The negative gradient is

$$\Rightarrow \; -\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} \, O_{pi} \quad \cdots (3)$$

Define

$$\delta_{pj} \triangleq -\frac{\partial E_p}{\partial net_{pj}}$$

$$= -\frac{\partial E_p}{\partial O_{pj}} \; \underbrace{\frac{\partial O_{pj}}{\partial net_{pj}}}_{f_j'(net_{pj}) \text{ from (1)}} \quad \text{- - - - - - - - - - (4)}$$

$$O_{pj} = f_j(net_{pj}) \quad \text{- - - (1)}$$

$$net_{pj} = \sum_i w_{ji} O_{pi} \quad \text{-- (2)}$$

Now we have two different cases to compute the first term: $\frac{\partial E_p}{\partial O_{pj}}$

12

Generalized Delta Rule:

Case I. Output layer

$$\frac{\partial E_p}{\partial O_{pj}} = -(t_{pj} - O_{pj}) \qquad E_p = \frac{1}{2}(t_{pj} - O_{pj})^2,$$

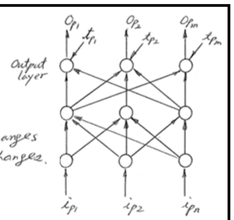
Thus, from (4),

$$\delta_{pj} = (t_{pj} - O_{pj})\, f_j'(net_{pj}) \qquad (5)$$

$$\delta_{pj} \triangleq -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial O_{pj}} \underbrace{\frac{\partial O_{pj}}{\partial net_{pj}}}_{f_j'(net_{pj}) \text{ from (1)}} \quad \cdots\cdots\cdots\cdots (4)$$

13

Generalized Delta Rule:



Case II. Hidden layers

Note that when the output O_{pj} of hidden layer changes the net inputs net_{pk} to the output layer changes.

$$net_{pk} = \sum_i w_{ki} O_{pi}$$

Thus,

$$\frac{\partial E_p}{\partial O_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \underbrace{\frac{\partial net_{pk}}{\partial O_{pj}}}$$

$$= \sum_k \underbrace{\frac{\partial E_p}{\partial net_{pk}}}_{-\delta_{pk}\text{ : already calculated at the output layer.}} w_{kj}$$

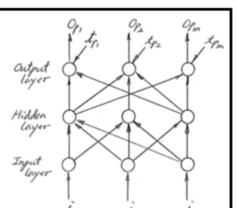
$$= -\sum_k \delta_{pk} w_{kj}$$

Thus, from (4),

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{kj} \quad \text{: error is propagated backward} \quad (6)$$

14

Generalized Delta Rule:



The negative gradient is

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} O_{pi} \quad \cdots (3)$$

Thus, from (4),

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{kj} \quad \text{: error is propagated backward} \quad (6)$$

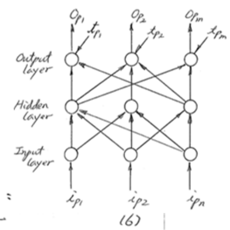
The weight adjustment is, therefore, in the direction of the negative gradient:

$$\Delta_p w_{ji} = \eta\, \delta_{pj} O_{pi} \quad \cdots\cdots\cdots (7)$$

NOTE: δ_{pj} is propagated backward (to neuron j from the next layer).
O_{pi} is propagated forward (from neuron i in the previous layer)

15

Generalized Delta Rule:



Thus, from (4),

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{kj} \quad : \quad (6)$$

The weight adjustment is, therefore, in the direction of the negative gradient:

$$\Delta_p w_{ji} = \eta \, \delta_{pj} O_{pi} \quad \cdots\cdots\cdots \quad (7)$$

Note that the errors in the output layer (5) are propagated backward through output weights (weights between output layer and hidden layer) (6), and multiplied by inputs propagated forward (7) to compute the weight adjustment.

16

Generalized Delta Rule:

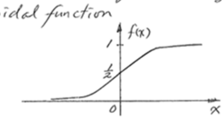
Thus, from (4),

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{kj} \quad : \text{ error is propagated backward} \quad (6)$$

Derivative of the activation function, $f_j'(net_{pj})$, can be computed directly without differentiating numerically.

For example, for a sigmoidal function

$$f(x) = \frac{1}{1+e^{-(x+T)}}$$



it can be shown that

$$f'(x) = f(x)\,[1 - f(x)]$$

Activation function ($T = 0$)

Therefore, from (1),

$$f_j'(net_{pj}) = O_{pj}\,[1 - O_{pj}]$$

$O_{pj} = f_j(net_{pj})$ --- (1)

$net_{pj} = \sum_i w_{ji} O_{pi}$ --- (2)

17

Generalized Delta Rule:

Thus, from (4),

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{kj} \quad : \text{ error is propagated backward} \quad (6)$$

The weight adjustment is, therefore, in the direction of the negative gradient:

$$\Delta_p w_{ji} = \eta \, \delta_{pj} O_{pi} \quad \cdots\cdots\cdots \quad (7)$$

In general, the momentum is added to avoid local minima:

$$\Delta_p w_{ji}^{new} = \eta \, \delta_{pj} O_{pi} + \alpha \Delta_p w_{ji}^{prev}$$

where the second term on the right hand side is the momentum term. This term adds a portion of the most recent weight change when computing the new weight change. The momentum term is supposed to give the neuron momentum in weight space, enabling it to pass through local minima.

18