ADALINE

Dr Dayal Kumar Behera

ADALINE

- ADAaptive LInear NEuron.
- Adaptive Linear Neural Element.
- Developed by prof. Bernard Widrow and Ted Hoff of Stanford University in the 1960s.
- It is a supervised learning network.
- It has a single linear unit (One output neuron and the input-output relationship is linear)
- The network is trained using **Delta Rule**. (Also known as LMS (Least Mean Square) Rule or “Widrow-Hoff Rule”)

ADALINE

Structure: Single-layer neural network.

Activation Function: Linear during training, typically a step function during classification.

Output: Continuous-valued during training, binary during classification.

Architecture

Learning

- ADALINE uses Least Mean Square (LMS) algorithm, also known as the Widrow-Hoff rule or Delta rule for training/learning.

LMS Algorithm

- LMS algorithm is a supervised learning algorithm.
- It aims to minimize the mean squared error between the desired output and the actual output(computed output) of the network.

LMS Algorithm

Objective: Minimize the mean squared error (MSE) between the computed output and the desired output(target).

Weight Update Rule (Delta Rule):

$$w_i^{(new)} = w_i^{(old)} + \Delta w_i$$

Where,

$$\Delta w_i = \eta \, (t - y) \, x_i$$

Note# ADALINE employs the **identity activation function** at the output unit during training. This implies that during training $\boldsymbol{y} = \boldsymbol{y_{in}}$

Steps of LMS Algorithm

1. **Initialization**: Initialize weights w_i to small random values.
2. **Input Presentation**: Present an input vector (x) to the network.
3. **Output Calculation**: Compute the actual output (y) using the linear activation function:

$$y = f(y_{in}) = f(\,w\,.\,x\,) = f\,(w^T x) = y_{in}$$

4. **Error Calculation:** Calculate the error (e) as follows:

e = t - y

5. **Weight Update**: Update the weights using the LMS rule:

$$w_i^{(new)} = w_i^{(old)} + \Delta w_i \quad \text{Where,} \quad \Delta w_i = \eta\,(t - y\,)\,x_i$$

6. **Iteration**: Repeat steps 2-5 for each input vector in the training set until the error is minimized.

LMS Algorithm

Weight Update Rule can also be represented as

$$w_i^{(new)} = w_i^{(old)} + \Delta w_i$$

Where,

$$\Delta w_i = \eta \,(t - y)\, x_i$$

$$\boldsymbol{w(n+1)} = \boldsymbol{w(n)} + \Delta w(n)$$

$$\boldsymbol{w(n+1) = w(n) + \eta\, e(n)\, x(n)}$$

How?

LMS Algorithm: Mathematical Derivation

Objective: Minimize the mean squared error (MSE) between the computed output and the desired output(target).

<u>Why Squared Error?</u>

- x^2 is differentiable, while $|x|$ is not differentiable at $x = 0$
- We generally want larger errors to be penalized more than smaller ones.

LMS Algorithm: Mathematical Derivation

- The squared error for a particular binary pattern is

$$E = e^2 = (t - y)^2 \qquad (1)$$

- Error function chosen for easy derivation is

$$E(n) = \frac{1}{2} e^2(n) \qquad (2)$$

- **Objective:** Minimize the error E w.r.t. weight w.
- This is an un-constraint optimization, minimize E(n) w.r.t. w for each n.
- We can apply Gradient Descent Algorithm.

LMS Algorithm: Mathematical Derivation

$$Error\ Function$$

$$E(n) = \frac{1}{2}\ e^2(n)$$

- **Gradient Descent Algorithm:**

  The error can be minimized by adjusting the weight $\boldsymbol{w_{(n)}}$ in the direction of negative gradient.

$$\boldsymbol{w(n+1) = w(n) - \eta\, \nabla_{w(n)}\, E(n)} \qquad \textbf{(3)}$$

LMS Algorithm: Mathematical Derivation

- $$\nabla_w E(n) = \frac{\partial}{\partial w(n)} \left[\frac{1}{2} \; e^2(n)\right]$$

$$\Rightarrow \frac{1}{2} \left[\frac{\partial e^2(n)}{\partial e(n)} . \frac{\partial e(n)}{\partial w(n)}\right]$$

$$\Rightarrow \frac{2}{2} \left[e(n) . \frac{\partial e(n)}{\partial w(n)}\right]$$

$$\Rightarrow \left[e(n) . \frac{\partial [t(n) - y(n)]}{\partial w(n)}\right]$$

t(n) is a constant

$$\Rightarrow - e(n) . \frac{\partial y(n)}{\partial w(n)}$$

$y = \; w^T x$

$$\Rightarrow - e(n) . \frac{\partial \, [w^T(n) . \;\; x(n)]}{\partial w(n)}$$

$$\Rightarrow - e(n) . x(n) \qquad \textbf{(4)}$$

LMS Algorithm: Mathematical Derivation

- **Putting value of $\nabla_w E(n) = -\, e(n)\, .\, x(n)$ in Eq. (3),**

$$w(n+1) = w(n) - \ \eta\, \nabla_{w(n)} E(n)$$

$$\Rightarrow \ w(n+1) = w(n) + \ \eta\, e(n)\, x(n)$$

Perceptron Vs. ADALINE

	Perceptron	ADALINE
Activation Function	Uses a binary step function or thresholding function	Uses a linear activation function.
Output	binary outputs (0 or 1) or Bipolar (1 or -1)	continuous-valued outputs.
Learning Rule	Perceptron learning rule	Delta rule (Widrow-Hoff rule)
Weight Update Rule	$w(n+1) = w(n) + \Delta w(n)$ $\Delta w(n) = \boldsymbol{\eta}\ (t - y)\ x(n)$	$w(n+1) = w(n) + \Delta w(n)$ $\Delta w(n) = \boldsymbol{\eta}\ (t - y\_in)\ x(n)$ Similar to the Perceptron but applied before the activation function.
Application	Suitable for simple binary classification tasks.	Suitable for regression problems and continuous-valued output prediction.