Adaptive linear neuron (Adaline) and LMS algorithm

ADALINE (**Adaptive Linear Neuron** or later **Adaptive Linear Element**) is an early single-layer artificial neural network and the name of the physical device that implemented this network. The network uses memistors. It was developed by Professor Bernard Widrow and his graduate student Ted Hoff at Stanford University in 1960. It is based on the McCulloch–Pitts neuron. It consists of a weight, a bias and a summation function.

The difference between Adaline and the standard (McCulloch–Pitts) perceptron is that in the learning phase, the weights are adjusted according to the weighted sum of the inputs (the net). In the standard perceptron, the net is passed to the activation (transfer) function and the function's output is used for adjusting the weights.

A multilayer network of ADALINE units is known as a **MADALINE.**

Definition

Adaline is a single layer neural network with multiple nodes where each node accepts multiple inputs and generates one output. Given the following variables as:

- x is the input vector
- w is the weight vector
- n is the number of inputs
- θ some constant
- y is the output of the model

then we find that the output is $y = \sum_{j=1}^{n} x_j w_j + \theta$. If we further assume that

- $x_0 = 1$
- $w_0 = \theta$

then the output further reduces to: $y = \sum_{j=0}^{n} x_j w_j$

Learning algorithm

Let us assume:

- η is the learning rate (some positive constant)
- y is the output of the model
- o is the target (desired) output

then the weights are updated as follows

$$w \leftarrow w + \eta(o - y)x.$$

The ADALINE converges to the least squares error which is

$$E = (o - y)^2.$$

This update rule is in fact the stochastic gradient descent update for linear regression

MADALINE

MADALINE (Many ADALINE) is a three-layer (input, hidden, output), fully connected, feed-forward artificial neural network architecture for classification that uses ADALINE units in its hidden and output layers, i.e., its activation function is the sign function. The three-layer network uses memistors. Three different training algorithms for MADALINE networks, which cannot be learned using backpropagation because the sign function is not differentiable, have been suggested, called Rule I, Rule II and Rule III.

MADALINE Rule 1 (MRI) - The first of these dates back to 1962 and cannot adapt the weights of the hidden-output connection.

MADALINE Rule 2 (MRII) - The second training algorithm improved on Rule I and was described in 1988. The Rule II training algorithm is based on a principle called "minimal disturbance". It proceeds by looping over training examples, then for each example, it:

- finds the hidden layer unit (ADALINE classifier) with the lowest confidence in its prediction,
- tentatively flips the sign of the unit,
- accepts or rejects the change based on whether the network's error is reduced,
- stops when the error is zero.

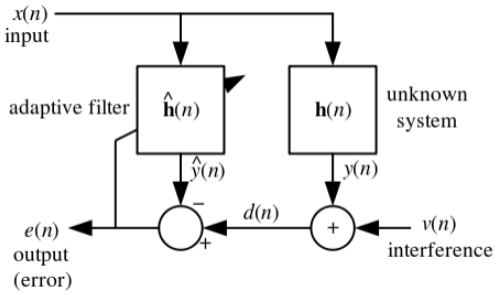
MADALINE Rule 3 - The third "Rule" applied to a modified network with sigmoid activations instead of signum; it was later found to be equivalent to backpropagation.

Additionally, when flipping single units' signs does not drive the error to zero for a particular example, the training algorithm starts flipping pairs of units' signs, then triples of units, etc.

LMS algorithm (Least mean squares filter)

Least mean squares (LMS) algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean square of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in that the filter is only adapted based on the error at the current time. It was invented in 1960 by Stanford University professor Bernard Widrow and his first Ph.D. student, Ted Hoff.

Problem formulation



Relationship to the Wiener filter

The realization of the causal Wiener filter looks a lot like the solution to the least squares estimate, except in the signal processing domain. The least squares solution, for input matrix ***X*** and output vector ***y*** is:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\boldsymbol{y}.$$

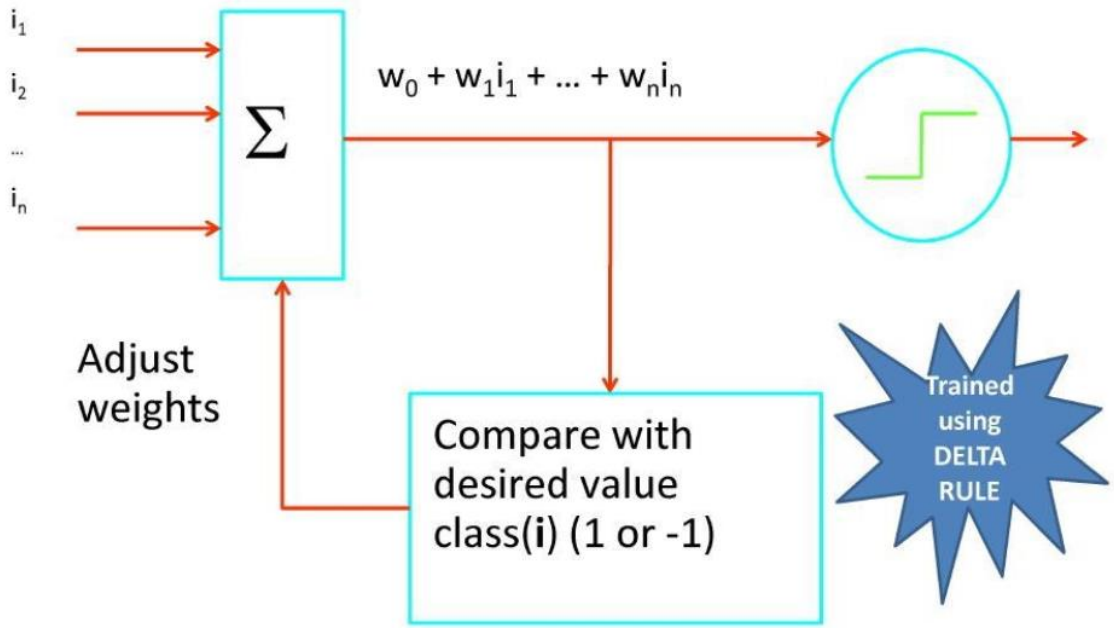
The FIR least mean squares filter is related to the Wiener filter, but minimizing the error criterion of the former does not rely on cross-correlations or auto-correlations. Its solution converges to the Wiener filter solution. Most linear adaptive filtering problems can be formulated using the block diagram above. That is, an unknown system **h**(n) is to be identified and the adaptive filter attempts to adapt the filter **h'**(n) to make it as close as possible to **h**(n) , while using only observable signals:

$x(n)$, $d(n)$ and $e(n)$; but $y(n)$, $v(n)$

And **h**(n) are not directly observable. Its solution is closely related to the Wiener filter.

ADALINE – ADAptive Linear NEuron

Architecture



ALGORITHM

Step 0	Initialize weights; set learning rate α
Step 1	While stopping condition is false do steps 2-6
Step 2	For each bipolar trining pair s:t do steps 3-5

Step 3 Set activations of input unit i = 1…..n ; Xi = Si

Step 4 Compute net input to output unit

$$b + \sum_{i=1}^{n} x_i w_i = 0$$

Step 5 Update bias and weights

$$b(new) = b(old) + \propto (t - y\_in) \qquad \text{i = 1…..n}$$

$$wi(new) = wi(old) + \propto (t - y\_in)xi$$

Step 6 Test for stopping condition

If the largest weight change that occurred in step 2 is similar than a specified tolerance the stop else continue.

LMS Learning Rule:

To train Adaline to perform a given processing function.

Let x be the input

vector w: weights

y: output values

d_k: Desired or correct output value

Manual Calculation of w*(weight adjustment)

Given the set of input, desired output pairs {(x1,d1), (x2,d2)}, the best value of w* needs to be calculated.

Let yk be the actual output for k^{th}

input vector. Error term $s_k = d_k —$

y_k (Equation 1)

Mean squared error s_k^2 is defined

as follows:

$$s_k^2 = \frac{1}{L}\sum_{k=1}^{L} s_k^2$$ (Eqn 2) where L is the number of input vectors in training set

$$y^k = w^t x_k$$

Substituting equations 3, 2 in 1

$$s_k^2 = (d_k — w^t x_k)^2$$

Back Propagation Network

−Supervised Learning

- Feed forward network

- Multilayer Perceptron

BPN Rule: Adjusting the weights in previous level of layers to reduce error. This leads to Delta learning rule

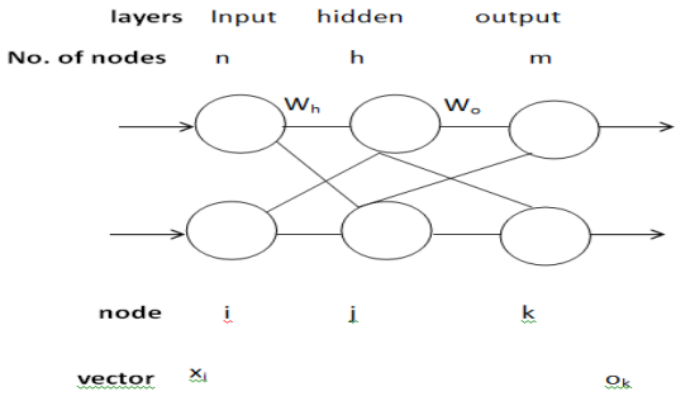


Fig. BPN topology

BPN algorithm

Read 'n' number of input nodes

Read 'h' number of hidden nodes

Read 'm' number of output nodes

Read the input vector x_i

    For i=1 to n

        Read x[i]

Read the output vector o_i

    For i=1 to m

        Read o[j]

Read the input hidden weights wh_{ij}

For i=1 to n

For j=1 to h

Read wh[i][j]

Read the output hidden weights wo_{ij}

For i=1 to h

For j=1 to m

Read wo[i][j]

Calculate Net h_j

For j= 1 to h

$$net[j] = \sum_{k=1}^{n} x[k] \times wh[k][j]$$

Calculate the output of hidden layer oh[j]: $oh[j] = \frac{1}{(1 + e^{-neth[j]})}$

For j= 1 to h

Calculate the net output

Calculate net o_j

For j=1 to m

For k=1 to h

net o[j]+= oh[k] × wo[k][j]

Calculate

For i=1 to m

Calculate error output

For i=1 to m

$eo[i] = o[i] - oo[i]$

Calculate error hidden

For i=1 to h

For j=1 to m

$eh[i] += eo_j \times wh[j][i]$

Calculate the new weights(nwo_{ij}) for the hidden output

For i=1 to h

For j=1 to m

$nwo[i][j] = wo[i][j] + (y \times eo[j] \times oh[i])$

New weight for input hidden layer is calculated as follows

For i=1 to n

For j=1 to h

$nw[i][j] = wh[i][j] + (y \times eh[j] \times x[i])$

The new weight obtained for hidden output layer is nwo_{ij} and the new weight obtained for input hidden

QUESTIONS FOR PRACTICE

Part – A

1. Define Neural network.
2. Compare bio neural network with artificial neural network.
3. What are the applications of Neural network?
4. Differentiate between Supervisory and unsupervisory learning.
5. What is Activation Function?
6. What are the types of activation functions?
7. Define Delta Rule.
8. What is the significance of delta rule?
9. What is meant by linear separability?
10. Give some examples for linear separability.
11. Distinguish between Adaline and Madaline Algorithm.

12. What are the merits of Perceptron Algorithm?
13. Define Hebb Rule.
14. What is ADALINE algorithm?
15. Draw the architecture of McCulloch Pitts Network.

Part – B

1. a. Compare biological neurons with neural networks. b. What are the various types of neural network?
2. Train the given logic gates using perceptron algorithm
   a. AND b. OR
3. Explain about back propagation
4. Explain the Adaline Architecture with the neat sketch.
5. a. Explain delta rule
   b. Explain LMS algorithm

REFERENCE

1. Anderson, James A.; Rosenfeld, Edward (2000). Talking Nets: An Oral History of Neural Networks. ISBN 9780262511117.

2. Adaline (Adaptive Linear)- (PDF). CS 4793: Introduction to Artificial Neural Networks. Department of Computer Science, University of Texas at San Antonio.

3. Rodney Winter; Bernard Widrow (1988). MADALINE RULE II: A training algorithm for neural networks (PDF). IEEE International Conference on Neural Networks. pp. 401–408.

4. Widrow, Bernard; Lehr, Michael A. (1990). "30 years of adaptive neural networks: perceptron, madaline, and backpropagation". Proceedings of the IEEE.

5. Monson H. Hayes: Statistical Digital Signal Processing and Modeling, Wiley, 1996, ISBN 0-471-59431-8

6.Simon Haykin: Adaptive Filter Theory, Prentice Hall, 2002, ISBN 0-13-048434-2

7. Simon S. Haykin, Bernard Widrow (Editor): Least-Mean-Square Adaptive Filters, Wiley, 2003, ISBN 0-471-21570-8

8. Bernard Widrow, Samuel D. Stearns: Adaptive Signal Processing, Prentice Hall, 1985, ISBN 0-13-004029-0

9. Weifeng Liu, Jose Principe and Simon Haykin: Kernel Adaptive Filtering: A Comprehensive Introduction, John Wiley, 2010, ISBN 0-470-44753-2

10. Paulo S.R. Diniz: Adaptive Filtering: Algorithms and Practical Implementation, Kluwer Academic Publishers, 1997, ISBN 0-7923-9912-9