ADALINE AND MADALINE

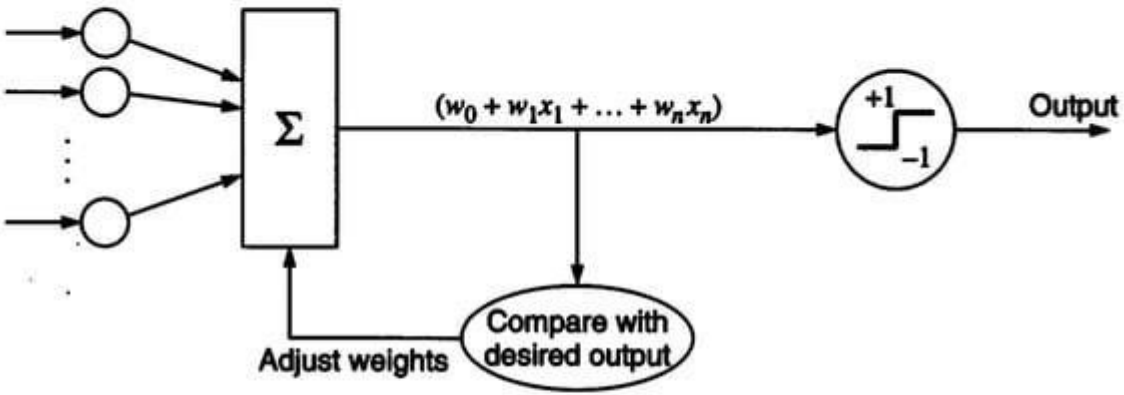
ARTIFICIAL NEURAL NETWORK

ADALINE

What is an ADALINE Network?

- Stands for **Ada**ptive **Lin**ear **E**lement
- It is a simple perceptron-like system that accomplishes classification by modifying weights in such a way as to diminish the MSE(MEAN SQUARE ERROR) at every iteration. This can be accomplished using gradient Adaptive linear element (Adaline).
- Used in neural nets for
  - **Adaptive filtering**
  - **Pattern recognition**

ADALINE - ARCHITECTURE



Adaptive linear element (Adaline).

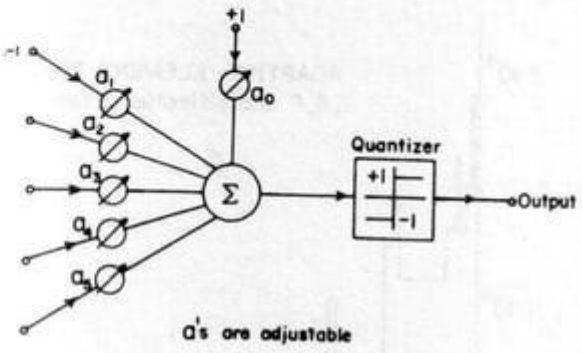
Using ADALINE Networks

- Initialize
  - **Assign random weights to all links**
- Training
  - **Feed in known inputs in random sequence**
  - **Simulate the network**
  - **Compute *error* between the input and the output (*Error Function**)**
  - **Adjust weights (*Learning Function**)**
  - **Repeat until *total error* < ε**
- Thinking
  - **Simulate the network**
  - **Network will respond to any input**
  - **Does not guarantee a correct solution even for trained inputs**

Adaline – Widrow-Hoff Learning

The learning idea is as follows:

- Define an error function that measure the performance of the performance in terms of the weights, input, output and desired output.
- Take the derivative of this function with respect to the weights, and modify the weights accordingly such that the error is decreased.
- Also known as the Least Mean Square (LMS) error algorithm, the Widrow-Hoff rule, the Delta rule.



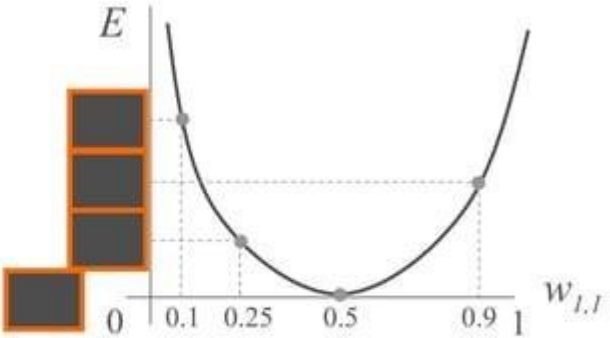
An adjustable neuron.

The ADALINE

- The Widrow-Hoff rule (also known as Delta Rule)
  - **Minimizes the error between desired output *t* and the net input *y_in***

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha(t_i - y\_in_i)x_j$$

  - **Minimizes the squared error for each pattern:** $E = (t - y\_in)^2$
  - **Example: if *s* = 1 and *t* = 0.5, then the graph of *E* against $w_{1,1}$ would be:**



  - **Gradient decent**

The ADALINE learning algorithm

Step 0 Initialize all weights and set learning rate

w_{ij} = (small random values)

$\alpha = 0.2$ (for example)

Step 1 While stopping condition is false

Step 1.1 For each training pair **s:t**:

Step 1.1.1 Set activations on input units

$x_j = s_j$

Step 1.1.2 Compute net input to output units

$y\_in_i = b_i + \Sigma x_j w_{ij}$

Step 1.1.3 Update bias and weights

$b_i(\text{new}) = b_i(\text{old}) + \alpha(t_i - y\_in_i)$

$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha(t_i - y\_in_i)x_j$

Least Square Minimization

- Find gradient of error over all examples. Either calculate the minimum or move opposite to gradient.
- Widrow-Hoff(LMS): Use instantaneous example as approximation to gradient.
  - **Advantages: No memory; on-line; serves similar function as noise to avoid local problems.**
  - **Adjust by $\mathbf{w}$(new) = $\mathbf{w}$(old) + α (δ) $\mathbf{x}$ for each $\mathbf{x}$.**
  - **Here δ = (desired output – $\Sigma \mathbf{wx}$)**

LMS (Least Mean Square Alg.)

1. Apply input to Adaline input

2. Find the square error of current input
   - **Errsq(k) = (d(k) - W x(k))**2**

3. Approximate Grad(ErrorSquare) by
   - **differentiating Errsq**
   - **approximating average Errsq by Errsq(k)**
   - **obtain -2Errsq(k)x(k)**

5. Update W: $W(new) = W(old) + 2\mu Errsq(k)X(k)$

7. Repeat steps 1 to 4.

Mean Square Error

- Supervised neural networks that use an MSE cost function can use formal statistical methods to determine the confidence of the trained model.
- The MSE on a validation set can be used as an estimate for variance.
- This value can then be used to calculate the confidence interval of the output of the network, assuming a normal distribution.

Training Set:

$$\{\mathbf{p}_1,\mathbf{t}_1\}, \{\mathbf{p}_2,\mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q,\mathbf{t}_Q\}$$

Input: $\mathbf{p}_q$ Target: $\mathbf{t}_q$

Notation:

$$\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \qquad a = {}_1\mathbf{w}^T\mathbf{p} + b \Rightarrow a = \mathbf{x}^T\mathbf{z}$$

Mean Square Error:

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

Error Analysis

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

$$F(\mathbf{x}) = E[t^2 - 2t\mathbf{x}^T\mathbf{z} + \mathbf{x}^T\mathbf{z}\mathbf{z}^T\mathbf{x}]$$

$$F(\mathbf{x}) = E[t^2] - 2\mathbf{x}^T E[t\mathbf{z}] + \mathbf{x}^T E[\mathbf{z}\mathbf{z}^T]\mathbf{x}$$

$$\boxed{F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}}$$

$$c = E[t^2] \qquad \mathbf{h} = E[t\mathbf{z}] \qquad \mathbf{R} = E[\mathbf{z}\mathbf{z}^T]$$

The mean square error for the ADALINE Network is a quadratic function:

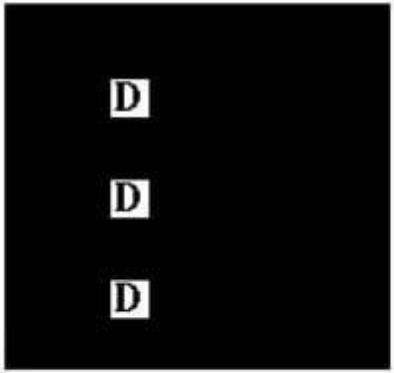
$$F(\mathbf{x}) = c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}$$

$$\mathbf{d} = -2\mathbf{h} \qquad \mathbf{A} = 2\mathbf{R}$$

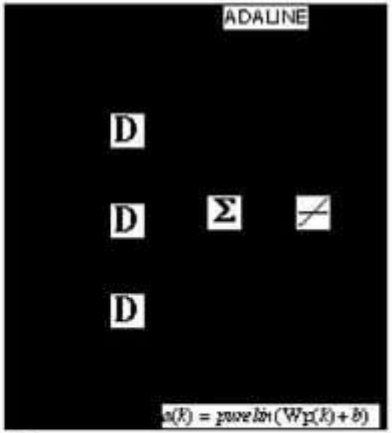
Adaptive Filtering

An **adaptive filter** is a filter that ***self-adjusts*** its transfer function according to an optimizing algorithm. Because of the complexity of the optimizing algorithms, most adaptive filters are digital filters that perform digital signal processing and adapt their performance based on the input signal.

Tapped Delay Line

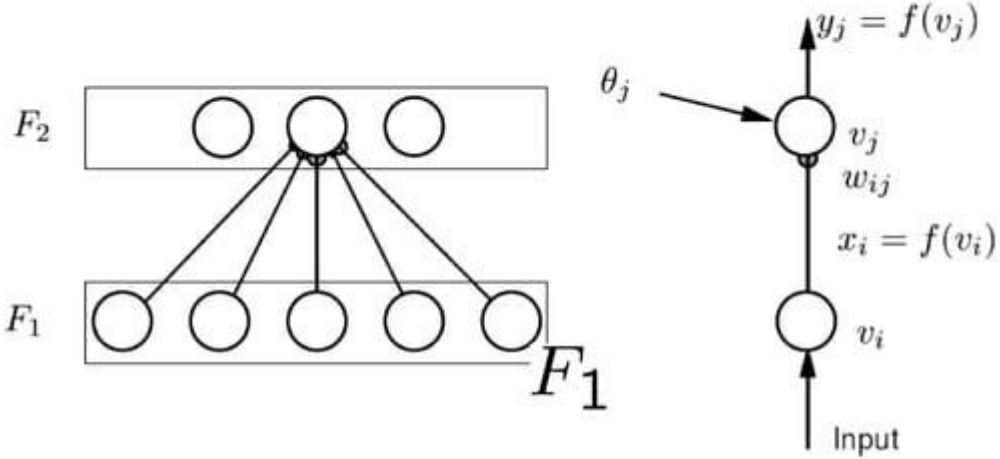


Adaptive Filter



$$a(k) = purelin(\mathbf{W}\,\mathbf{p} + b) = \sum_{i=1}^{R} w_{1,i} y(k-i+1) + b$$

Adaptive filter



$$\sum_i s_i w_{ij} = S \cdot W_j = |S||W_j| cos(S, W_j)$$

- $\mathbf{F_1}$ registers the input pattern.
- Signals $\mathbf{S_i}$ are modulated through weighted connections.
- $\mathbf{F_2}$ computes the pattern match between the input and the weights.

$$\sum_i x_i w_{ij} = X \cdot W_j = |X| |W_j| \cos(X, W_j)$$

Adaptive filter elements

- The dot product computes the projection of one vector on another. The term $|X||W_j|$ denotes the energy, whereas $\cos(X,W_j)$ denotes the pattern.
- If the both vectors are normalized

  $(|X| = |W_j| = |X||W_j| = 1)$,

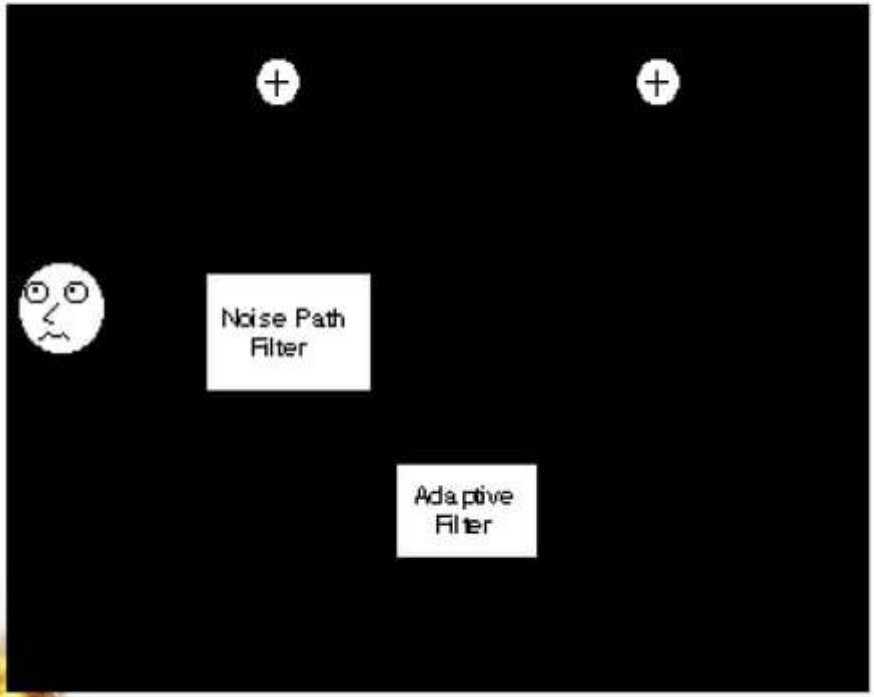
  then $X.W_j = \cos(X,W_j)$.

  This indicates how well the weight vector of the neuron matched with the input vector.
- The neuron with the largest activity at F_2 has the weights that are most close to the input.

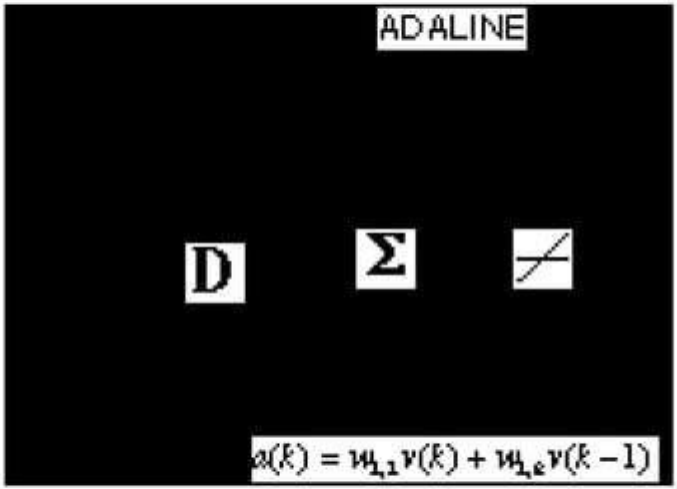
Applications

- Adaline has better convergence properties than Perceptron
- Useful in noise correction
- Adaline in every modem.

Example: Noise Cancellation

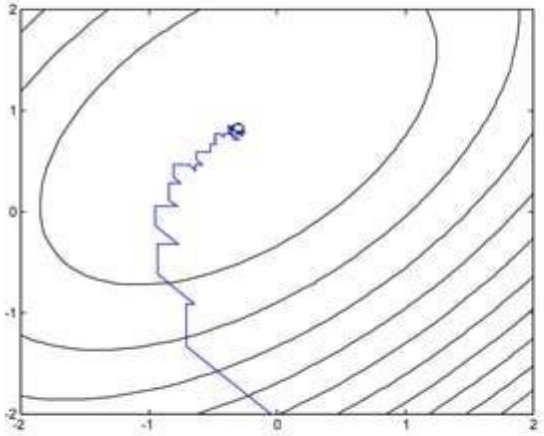


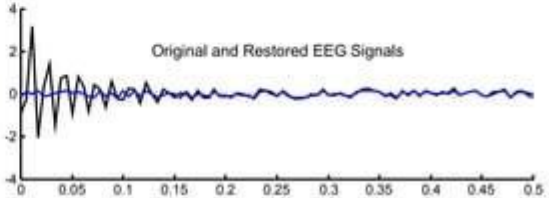
Noise Cancellation Adaptive Filter

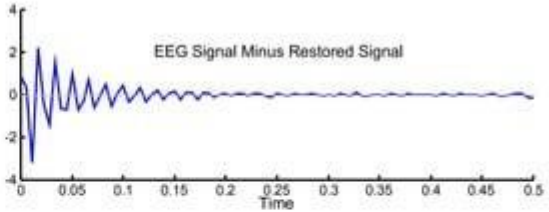


LMS Response

Adaptive Filter Cancellation of Contaminating Noise







Echo Cancellation

- ECHO - in long distance telephone lines
- Adaptive filter: deals with the choppy issue, which mimics the leakage of the incoming voice for suppressing the choppy speech from the outgoing signals.

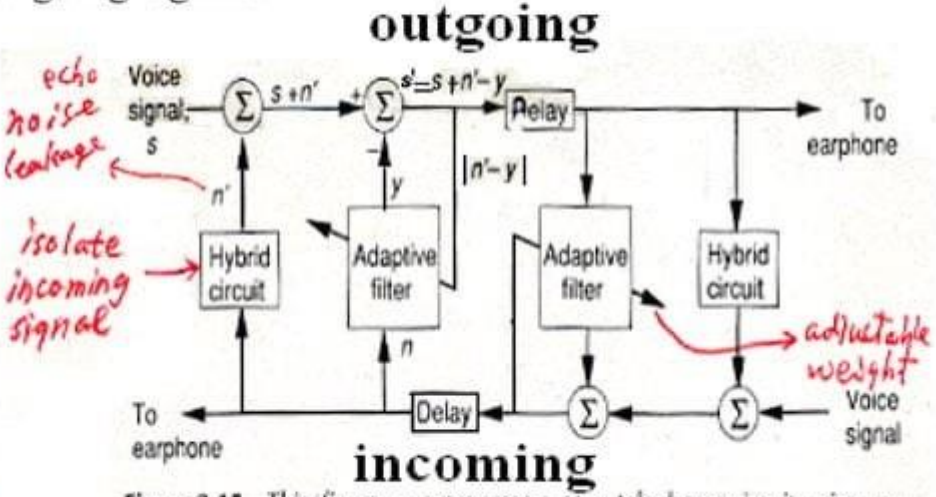


Figure 2.15 This figure is a schematic of a telephone circuit using an adaptive filter to cancel echo. The adaptive filter is depicted as a box; the slanted arrow represents the adjustable weights.

- n : incoming voice, s : outgoing voice
- n' : noise (leakage of the incoming voice)
- y : the output of the filter mimics

$$< s'^2 >=< (s+n'-y)^2 >=< s^2 > + < (n'-y)^2 >$$
$$+2 < s(n'-y) >=< s^2 > + < (n'-y)^2 >$$

(s not correlated with y,) $< s(n'-y) >= 0$

- $< \varepsilon^2 > = < s'^2 > - < s^2 > = < (n'-y)^2 >$
- minimize $< \varepsilon^2 >$ = minimize $< (n'-y)^2 >$

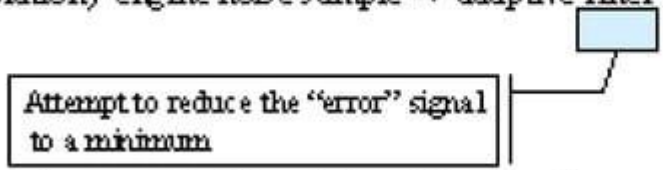
Adaline Device For Medical

- This adaline device input devices allow the computer to see, "feel," or "hear" its instructions. This is the training phase.
- Let's illustrate this by the way a doctor observes a multitude of symptoms, some precisely measured, such as temperature or blood pressure, and some more subtle, such as coloring, pain patterns, or demeanor. What the doctor does is almost subconsciously to attach a weight or asignificance to each of the symptoms, based on his gross experience with many diseases and many patients, and he combines these effects to arrive at a diagnosis.

EXAMPLE FOR ADALINE

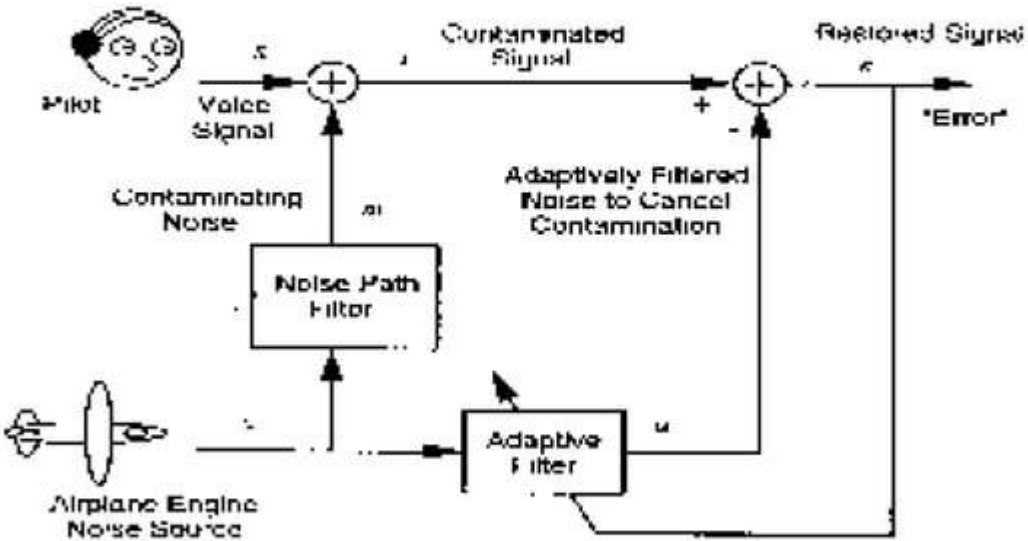
Solved Problem 10.7(1)

- The pilot of an airplane is talking into a microphone in his cockpit. The sound received by the air traffic controller in the tower is garbled because the pilot's voice signal has been contaminated by engine noise that reaches his microphone. Can you suggest an adaptive ADALINE filter that might help reduce the noise in the signal received by the control tower?
  - Solution) engine noise sample ⇒ adaptive filter through a microphone



therefore, can do this only by subtracting the component of the contaminated signal that is linearly correlated with the engine noise

Solved Problem 10.7(2)



Filtering Engine Noise from Pilot's Voice Signal

Comparison with Perceptron

- Both use updating rule changing with each input
- One fixes binary error; the other minimizes continuous error
- Adaline always converges; see what happens with XOR
- Both can REPRESENT Linearly separable functions

Summary

- Single layer nets have limited representation power (linear separability problem)
- Adaline – Widrow-Hoff Learning

  Define an error function that measure the performance of the performance in terms of the weights, input, output and desired output.
- The ADALINE learning algorithm
- Adaptive filter: deals with the choppy issue, which mimics the leakage of the incoming voice for suppressing the choppy speech from the outgoing signals.
- Adaline has better convergence properties than Perceptron
- Useful in noise correction
- Adaline in every modem.

MADALINE

Madaline : Many adaline

- **Madaline: Multiple Adalines connected.** A Madaline is a combination of many Adalines.

- This also enables the network to solve non-separable problems

- Learning algorithms for Madalines have gone through three stages of development. All three algorithms adhere to the "Minimum Disturbance" principle proposed by Widrow (1962), instead of explicitly computing the gradient of the network error.

- Nodes whose net input is sufficiently small are selected as candidates for weight changes. The possible result of changing the output of such a node is examined.

- If the change results in a decrease in network error, then weights of connections leading into that node are changed using the LMS (or similar) algorithm; otherwise, these weights remain unchanged.

- The magnitude of the weight change may be large enough to force the node output to change, or may be small so that a multitude of such changes are needed to achieve network error reduction.

- This process is repeated for all input patterns and for all nodes until network error becomes acceptably small.

Architecture

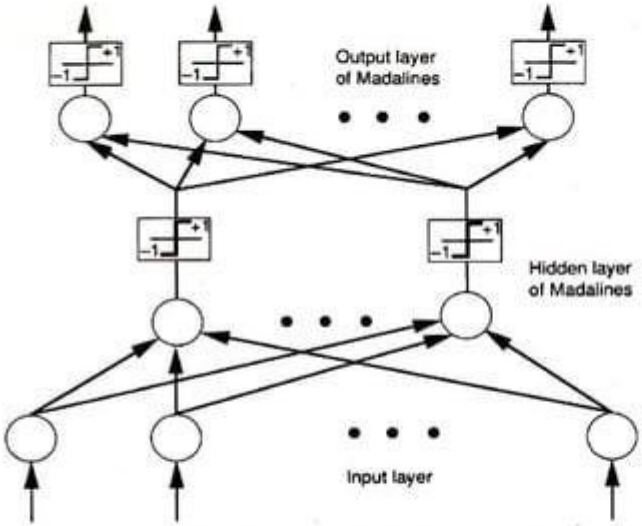


Figure 2.18 Many Adalines (the Madaline) can compute the XOR function of two inputs. Note the addition of the bias terms to each Adaline. A positive analog output from an ALC results in a +1 output from the associated Adaline; a negative analog output results in a −1. Likewise, any inputs to the device that are binary in nature must use ±1 rather than 1 and 0.

Madaline Rule I (MRI) training algorithm.

Madaline Rule I (MRI) training algorithm.

- **The *Madaline Rule I* (MRI) training algorithm. The goal is to make the smallest possible perturbation to the network, by modifying the weights on connections leading into some Adaline** (hidden node), so as to decrease network error on the current input sample.
- Note that the Adaline output must change in sign in order to have any effect on the network output.

Madaline Rule I (MRI) training algorithm.

repeat

Present a training pattern i to the network;

Compute outputs of various hidden nodes and output node;

if the pattern is misclassified,

then

Sort the Adalines in the network, in the order of increasing net input magnitude ($|\sum_j w_j i_j|$);

Let $S = (A_1, \ldots, A_k)$ be the sorted sequence, omitting nodes for which $|\sum_j w_j i_j| > \theta$,

where θ is a predetermined threshold;

while network output differs from desired output,

and some nodes in S remain to be examined

in this iteration, **do**

if reversing output of the next element $A_j \in S$

can improve network performance,

then Modify connection weights leading into A_j

to accomplish the output reversal;

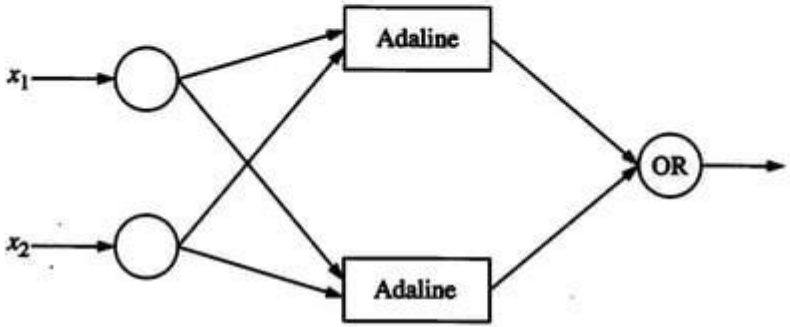
end-if

end-while

end-if

until performance is considered satisfactory or the upper bound on the number of iterations has been reached.

Madaline Rule I (MRI) training algorithm.



A Madaline with an output node that computes the OR logical function.

Madaline Rule II (MRII) training algorithm.

Madaline Rule II (MRI) training algorithm.

- The ***Madaline Rule 11*** (MRII) training algorithm, which is considerably different from backpropagation.

- The weights are initialized to small random values, and training patterns are repeatedly presented.

- The algorithm modifies the first hidden layer of Adalines (i.e., connection weights from input nodes to layer number 1), then the second hidden layer (weights from layer 1 to layer 2), and so on.

Madaline Rule II (MRII)

- Training algorithm – A trial–and–error procedure with a minimum disturbance principle (those nodes that can affect the output error while incurring the least change in their weights should have precedence in the learning process)

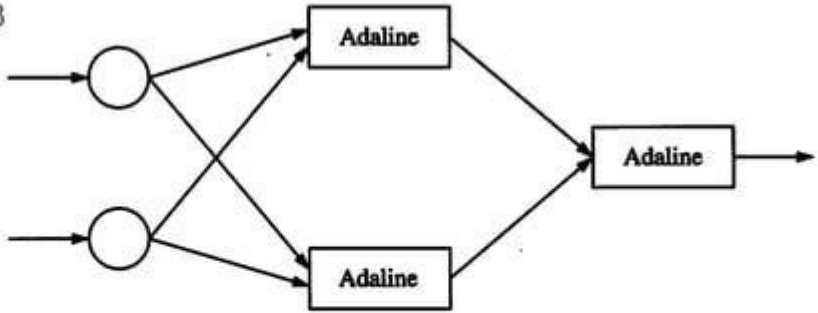
Madaline Rule II (MRI) training algorithm.

```
Algorithm MRII;
  repeat
    Present a training pattern i to the network;
    Compute outputs of all hidden nodes and the output node;
      Let h = 1;
      while the pattern is misclassified
              and h ≤ the number of hidden layers, do
            Sort the Adalines in layer h, in the order of
              increasing net input magnitude (|Σ_j w_j i_j|),
              but omitting nodes for which |Σ_j w_j i_j| > θ,
              where θ is a predetermined threshold;
            Let S = (A_1, ..., A_k) be the sorted sequence;
            while network output differs from desired output,
                    and S contains nodes not yet examined
                    in this iteration, do
                  if reversing output of the next element A_j ∈ S
                    can improve network performance,
                  then Modify connection weights leading into A_j
                    to accomplish the output reversal;
                  end-if;
            end-while
            h := h + 1;
    end-while
until performance is considered satisfactory or the upper
  bound on the number of iterations has been reached.
```

Madaline Rule II (MRI) training algorithm.

The *Madaline* Il architecture, shown in figure 4.3, improves on the capabilities of Madaline I, by using Adalines with modifiable weights at the Output layer of the network, instead of fixed logic devices.

Figure 4.3



High-level structure of a Madaline 11 with two Adalines at the first level and one Adaline at the second level.

Madaline Rule III (MRIII) training algorithm.

Madaline Rule III (MRI) training algorithm

- The MR III training algorithm was developed by Andes et al. (1988) to train feedforward networks with sigmoid node functions.
- This algorithm, described in figure 4.5, also follows the minimum disturbance principle using trial adaptations of nodes, instead of assuming the derivative of the node function to be known. Unlike MR II, weights of all nodes are adapted in each iteration.
- The MR III algorithm has been shown to be mathematically equivalent to backpropagation (Widrow and Lehr, 1990). However, each weight change involves considerably more computation than m backpropagation.
- MRIII has been advocated for some hardware implementations where the sigmoid node function is inaccurately implemented, so that the mathematically derived gradient is inapplicable.
- In such cases, MRIII is more useful since it effectively computes an approximation to the gradient~ without assuming a specific sigmoid node function. Note that all nodes in each layer are perturbed in each iteration, unlike MRII.

Madaline Rule III (MRI) training algorithm

repeat

Present a training pattern i to the network;

Compute outputs of hidden nodes and output nodes;

for $h = 1$ **to** the number of hidden layers, **do**

for each node k in layer h, **do**

Let E be the current network error;

Let network error be E_k if a small quantity $\epsilon > 0$ is added to the kth node's net input;

Adjust weights on connections to kth node using one of the following update rules:

$$\Delta w = -\eta i (E_k - E)^2 / \epsilon^2$$

or

$$\Delta w = -\eta i E (E_k - E) / \epsilon$$

end-for;

end-for

until current network error is considered satisfactory or the upper bound on number of iterations has been reached.

Comparison of MR III with MR II

G.7.3 Comparison of MRIII with MRII

- MRIII was derived from MRII by replacing the signum nonlinearities with sigmoids.

- The MRII network is highly discontinuous and nonlinear. Using an instantaneous gradient to adjust the weights is not possible. In fact, from the MSE surface for the signum Adaline it is clear that even gradient descent techniques which use the true gradient could run into severe problems with local minima. The idea of adding a perturbation to the linear sum of a selected Adaline element is workable, however. If the Hamming error has been reduced by the perturbation, the Adaline is adapted to reverse its output decision. This weight change is in the LMS direction, along its **X**-vector.

Comparison of MR III with MR II

If adapting the Adaline would not reduce network output error, it is not adapted. This is in accord with the minimal disturbance principle. The Adalines selected for possible adaptation are those whose analog sums are closest to zero, that is, the Adalines which can be adapted to give opposite responses with the smallest weight changes. It is useful to note that with binary ± 1 desired responses, the Hamming error is equal to $1/4$ the sum square error. Minimizing the output Hamming error is therefore equivalent to minimizing the output sum square error.

- The MRIII algorithm works in a similar manner. All the Adalines in the MRIII network are adapted, but those whose analog sums are closest to zero will usually be adapted most strongly, because the sigmoid has its maximum slope at zero, contributing to high gradient values. As with MRII, the objective is to change the weights for the given input presentation to reduce the sum square error at the network output. In accord with the minimal disturbance principle, the weight vectors of the Adaline elements are adapted in the LMS direction, along their **X**-vectors, and are adapted in proportion to their capabilities for reducing the sum square error (the square of the Euclidean error) at the output.

MADALINE- XOR EXAMPLE

x_1	x_2	x_1'	x_2'	XOR'	XOR
0	0	-1	-1	-1	0
0	1	-1	1	1	1
1	0	1	-1	1	1
1	1	1	1	-1	0

$$x' = 2x - 1, \quad x = \frac{x' + 1}{2}$$

MADALINE- XOR EXAMPLE

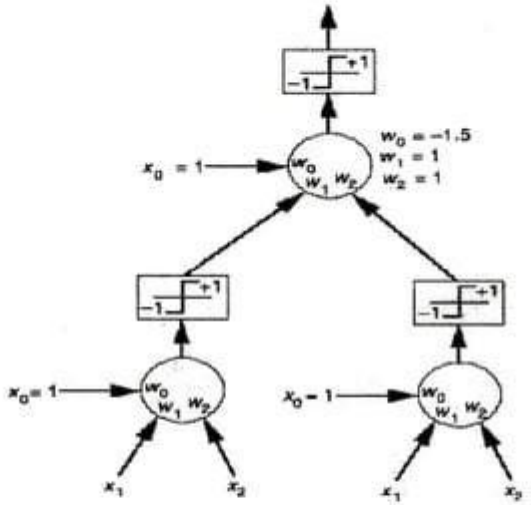
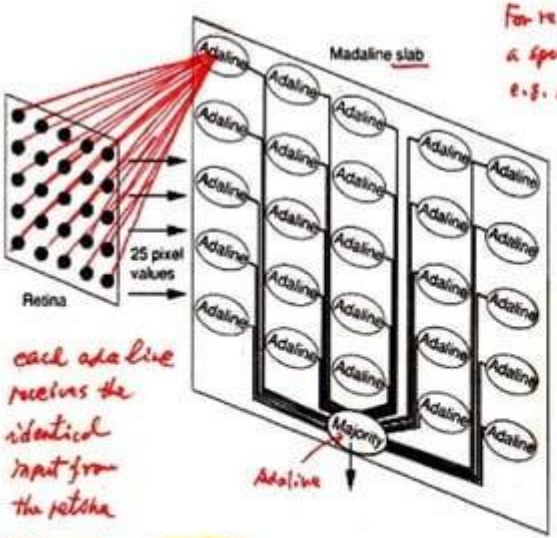


Figure 2.18 Many Adalines (the Madaline) can compute the XOR function of two inputs. Note the addition of the bias terms to each Adaline. A positive analog output from an ALC results in a +1 output from the associated Adaline; a negative analog output results in a −1. Likewise, any inputs to the device that are binary in nature must use ±1 rather than 1 and 0.

A Madaline for Translation – Invariant Pattern Recognition

A Madaline for Translation – Invariant Pattern Recognition

- Difficulties for pattern recognition -- noise, incompletion, distortion, transformation, occlusion
- ○ Translation-invariant pattern recognition



This single slab of Adalines will give the same output (either +1 or −1) for a particular pattern on the retina, regardless of the horizontal or vertical alignment of that pattern on the retina. All 25 individual Adalines are connected to a single Adaline that computes the majority function: If most of the inputs are +1, the majority element responds with a +1 output. The network derives its translation-invariance properties from the particular configuration of the weights. See the text for details.

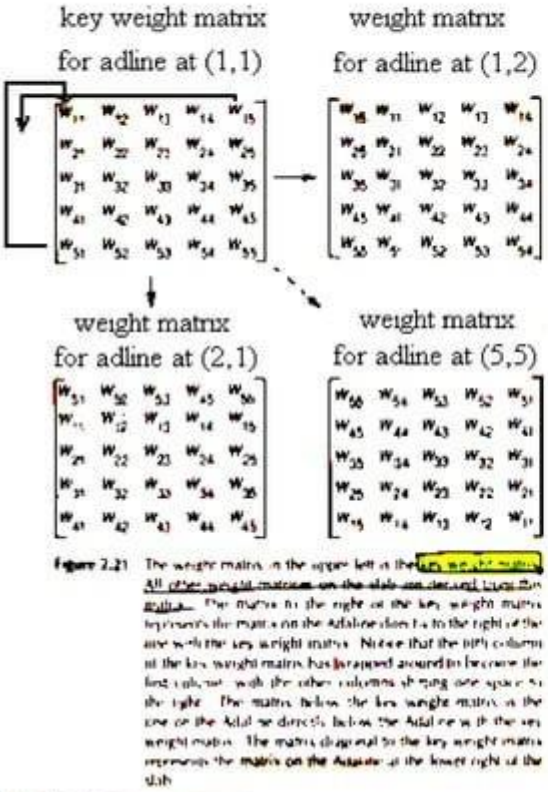
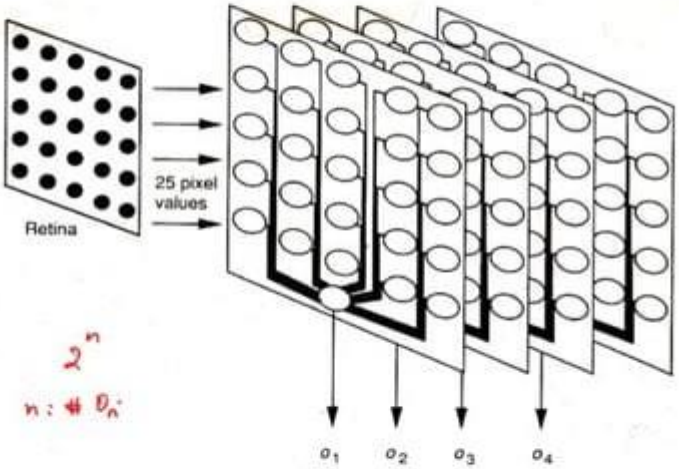


Figure 2.21 The weight matrix in the upper left is the key weight matrix. All other weight matrices on the slab are derived from this matrix. The matrix to the right of the key weight matrix represents the matrix on the Adaline sites to the right of the one with the key weight matrix. Notice that the fifth column of the key weight matrix has wrapped around to become the first column, with the other columns all moving one space to the right. The matrix below the key weight matrix is the one on the Adaline directly below the Adaline with the key weight matrix. The matrix diagonal to the key weight matrix represents the matrix on the Adaline at the lower right of the slab.

- **Relationships among the weight matrices of Adalines**

  Adalines possess the identical set of weight values, which have been trained to detect a particular pattern

- Extension -- Mutiple slabs with different key weight matrices for discriminating more then two classes of patterns



2^n

n : # O_n

Figure 2.22 Each of the four slabs in the system depicted here will produce a +1 or a −1 output value for every pattern that appears on the retina. The output vector is a four-digit binary number, so the system can potentially differentiate up to 16 different classes of input patterns.

APPLICATION OF MADALINE

Vehicle inductive signatures recognition using a Madaline neural network

- The degree of difficulty to accomplish a classification task is primarily determined by the class overlap in the input space [1, 2]. The difficulty is even greater if, in addition to the overlap, there is also class unbalance and the number of available patterns is reduced. Consider, for instance, a classification problem in which the input patterns are the inductive signatures of two classes of vehicles as shown in Fig. 1.

- These signals are collected by inductive loop traffic sensors [3] and the morphology of the curves in Fig. 1 is derived from the impedance alteration of the magnetic loop when the vehicle passes over [4]. It is hypothesized that the proximity of the metal parts of the axles alters the impedance of the loops and thus the presence of the axles is signalized. This way, the vehicle can be classified by the number of axles.

- Inductive signatures are used in traffic surveillance and management systems to recognize the class of a vehicle to estimate its speed and even to identify individual vehicle among other expected results [5–9]. These information are used to build a statistical database that may help traffic surveillance and management systems in decision-making. The class of a vehicle is one of the most important information and serves, for instance, for access control to areas where the circulation is restricted to certain types of vehicles and for the collection of different values in tollgates.

2.10 SOME APPLICATION DOMAINS

Neural networks have been successfully applied for the solution of a variety of problems. However, some of the common application domains have been listed below:

Pattern recognition (PR)/image processing

Neural networks have shown remarkable progress in the recognition of visual images, handwritten characters, printed characters, speech and other PR based tasks.

Optimization/constraint satisfaction

This comprises problems which need to satisfy constraints and obtain optimal solutions. Examples of such problems include manufacturing scheduling, finding the shortest possible tour given a set of cities, etc. Several problems of this nature arising out of industrial and manufacturing fields have found acceptable solutions using NNs.

Forecasting and risk assessment

Neural networks have exhibited the capability to predict situations from past trends. They have, therefore, found ample applications in areas such as meteorology, stock market, banking, and econometrics with high success rates.

Control systems

Neural networks have gained commercial ground by finding applications in control systems. Dozens of computer products, especially, by the Japanese companies incorporating NN technology, is a standing example. Besides they have also been used for the control of chemical plants, robots and so on.

Other applications :

- Net talk
- Stack price predictions
- Forecast the weather
- Read electrocardiograms
- Type out simple sentences that are spoken to it
- Drive a car
- Fly a plane perform

SUMMARY

- **Madaline: Multiple Adalines connected**
- This also enables the network to solve non-separable problems
- ***Madaline Rule I, Madaline Rule II, Madaline Rule III* (MRI) training algorithms**
- **A Madaline for Translation – Invariant Pattern Recognition**
- **Relationships among the weight matrices of Adalines**
- **Application :Vehicle inductive signatures recognition using a Madaline neural network**

THANK YOU