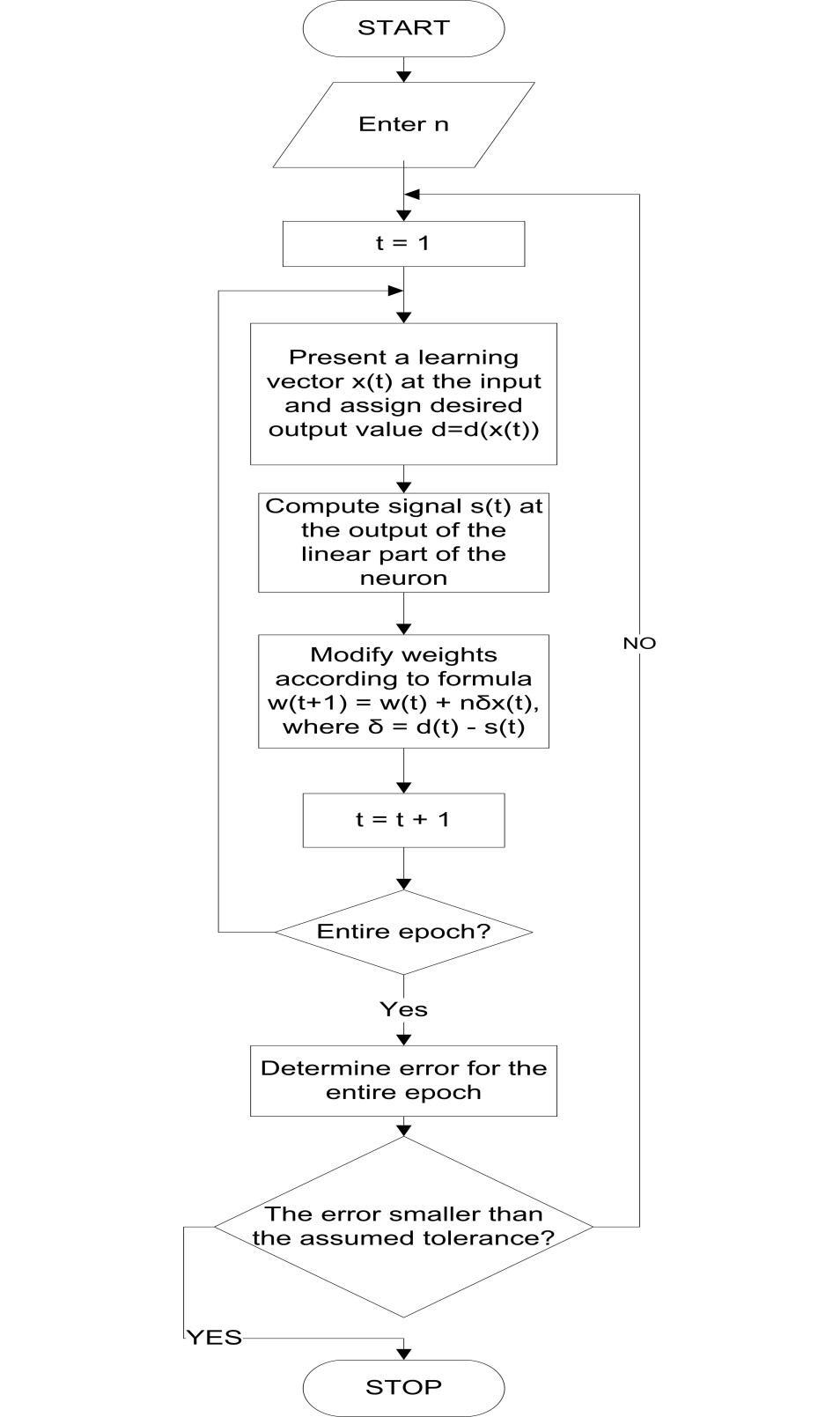
# **ADALINE**



**PSEUDO CODE**

1. Initialise weights (W1...Wn) and threshold (W0)
2. Set all weights and threshold to small random bipolar random values
3. Present new input and desired output
   1. present input vector x1,x2,...xn along with the desired output d(t)
4. Calculate the actual output[y(t)]
   1. y(t) = Fh[Σwi(t)\*xi(t) ]
5. Adapt weights

Wit+1()=Wi(t)+ηd(t)−{i=1n}∑Wi(t)Xi(t)Xi(t)

0 < i < n and η is the learning rate and usually is a small number ranging from 0 to 1.

6.Repeat step 2 to 4

Repeat until the desired outputs and the actual network outputs are all equal for all the input vectors of the training set.

**ALGORITHM AND IMPLEMENTATION**

This program decides if a given point is above the line y = (2x-1)/4

The adaline has two input nodes, receiving the coordinates of a point that it has to decide on.

**# defining a set of 500 patterns, type float, with 2 inputs and 1 output**

pat = Pattern("float", 2, 1, 500)

**# adaline network with two input nodes**

net = AdalineNet(2)

for i in range(500):

x = random.random()\*2.0 - 1.0 # adaline network need inputs -1 =< x =< 1

y = random.random()\*2.0 - 1.0

f = lambda a: (2\*a - 1)/4

res = (y < f(x)) and 1.0 or -1.0 # if y < f(x), the net should eval to 1.0

pat.setInputs([x, y], i)

pat.setOutputs([res], i)

**# Train the network with it patterns**

while good < 500 and itrn < 1000:

good = 0

for i in range(500):

net.loadInput(pat.getIn(i))  **# set the input values**

if net.getValue() != pat.getOut(i)[0]:

net.train()  **# network produced an error, train it**

**#output result**

print "x = %s, y = %s, result should be: %s" % (x,y,res)

print "The network produced: %s" % net\_res