Exercise
Number of training parameters for MLP Calculations

Homework Question #1

a) A neural network is used to classify a dataset composed of 5000 grayscale images of 0 to 9 digits. Each of the image has 32 x 32 pixels. The neural network contains 4 layers. The first layer is the input layer, the second layer has 20 neurons, the third one has 15 neurons, the last one is the softmax neuron of 10 classes. Please specify clearly the dimension of the matrices of each layer (both input weights and output) of the training process given a batch size of 50?

Layer one : input vector
Layer two:
- input matrix and
- weight matrix,
- output matrix

Layer three:
- input matrix
- weight matrix,
- output matrix

Layer four:
- input matrix,
- weight matrix, and
- output

b) How many batch iterations are needed to finish one epoch ?

Answer

a) A neural network is used to classify a dataset composed of 5000 grayscale images of 0 to 9 digits. Each of the image has 32 x 32 pixels. The neural network contains 4 layers. The first layer is the input layer, the second layer has 20 neurons, the third one has 15 neurons, the last one is the softmax neuron of 10 classes. Please specify clearly the dimension of the weight (only w) matrices of each layer (both input and output) of the training process given a batch size of 50? How many batch iterations are needed to finish one epoch?

b) What is the total number of unknown parameters (weights, w and bias, b) of this network.

Layer one : input vector : 32 x 32 = 1024

Layer two: input matrix = 50 x 1024
weight matrix = 1024 x 20

Layer three: input matrix = 50 x 20
weight matrix = 20 x 15

Layer four (softmax) : input matrix = 50 x 15
weight matrix = 15 x 10
output matrix = 50 x 10

Batch iteration needed : 100

Question #2 (2.0%) – page 2

A neural network is used to classify a dataset composed of 5000 grayscale images of 0 to 9 images. Each of the image has 32 x 32 pixels. The neural network contains 4 layers. The first layer is the input layer, the second layer has 20 neurons, the third one has 15 neurons, the last one is the softmax neuron of 10 classes.

What is the total number of unknown parameters (weights, w and bias, b) of this network.

Using a batch size of 50, so input layer has a matrix size of 50 x 1024 to the second layer

The second layer has 20 neurons, so the matrix is 1024 x 20, so
number of weight – 1024x20 = 20480, number of bias = 20, total = 20500

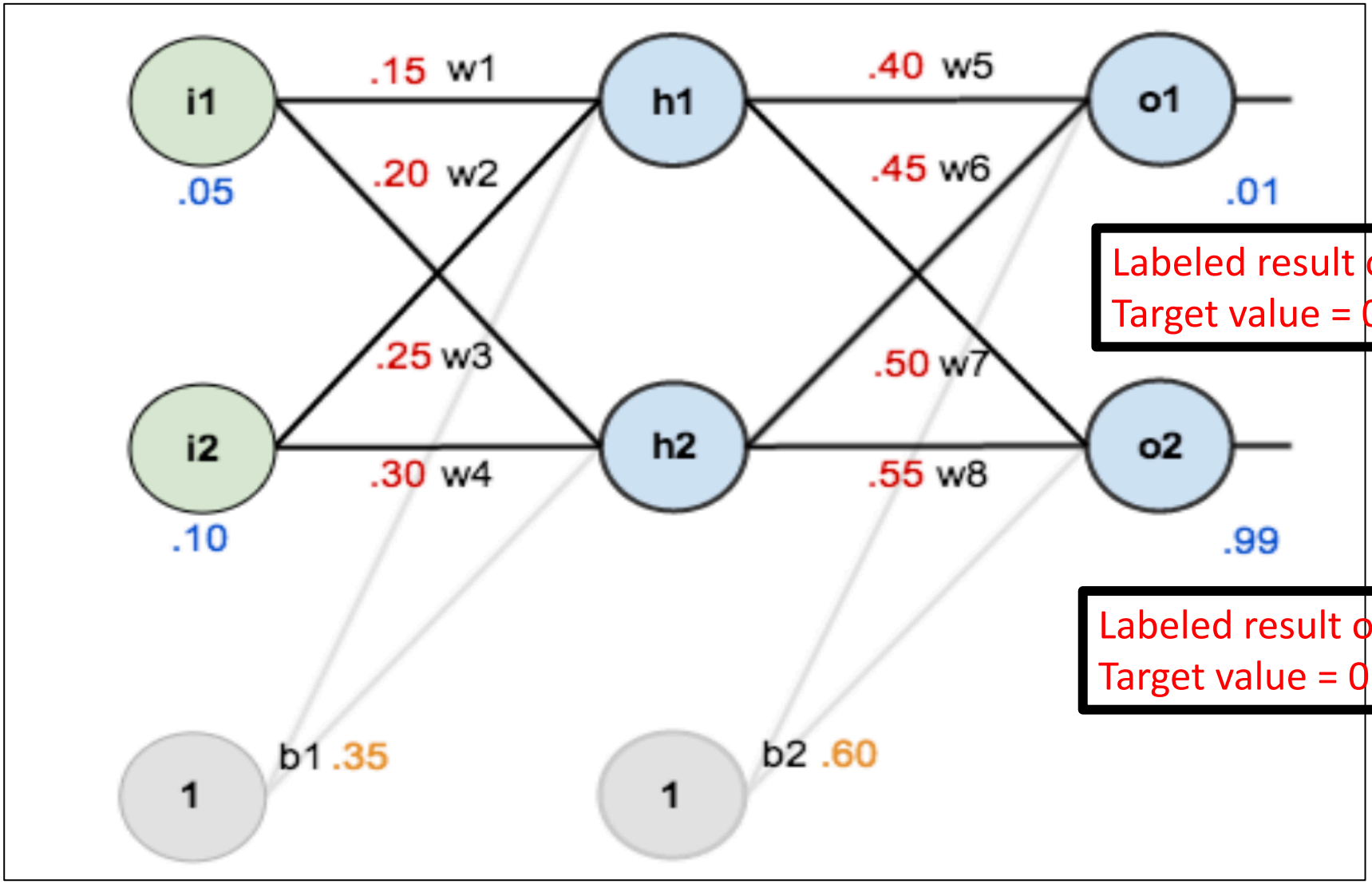
the third layer has 15 neurons, so the matrix is 20 x 15.
Total parameter = 300 (w) + 15 (b) = 315

The output of the softmax layer of 10 classes, 15 x 10
Total parameter = 150 (w) + 10 (b) = 160

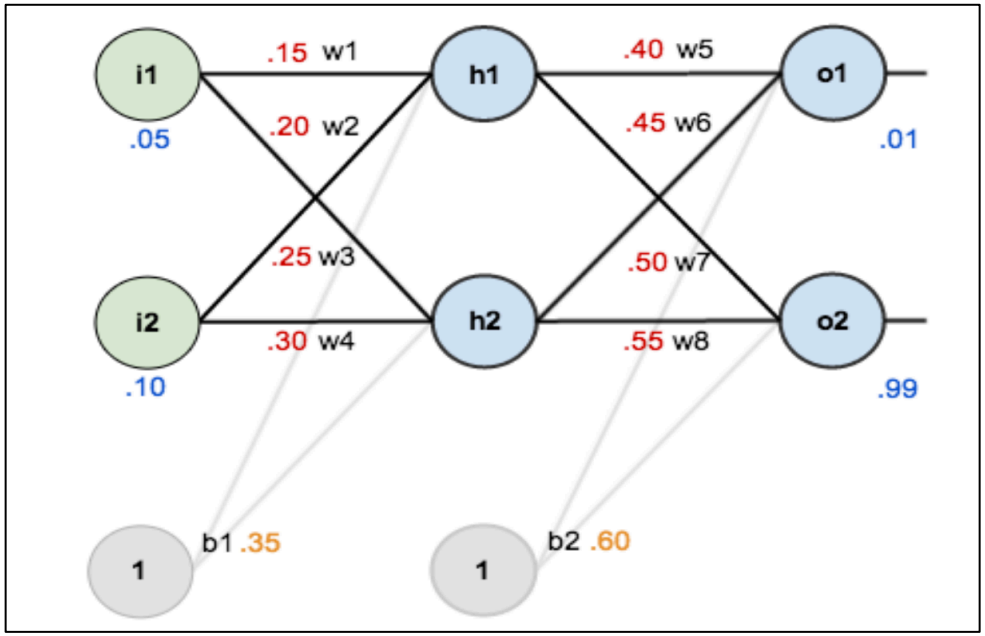
The number of trainable parameter is 20500 + 315 + 160 = 20975

Exercise
MLP Calculations

DNN Example


i1
.05
i2
.10
.15 w1
.20 w2
.25 w3
.30 w4
h1
h2
.40 w5
.45 w6
.50 w7
.55 w8
o1
.01
o2
.99
Labeled result of out1
Target value = 0.01
Labeled result of out2
Target value = 0.99
1
b1 .35
1
b2 .60

DNN Example (1)



w1=0.15

i1 = 0.05

*

w2=0.2

i2 = 0.1

*

+

b1=0.35

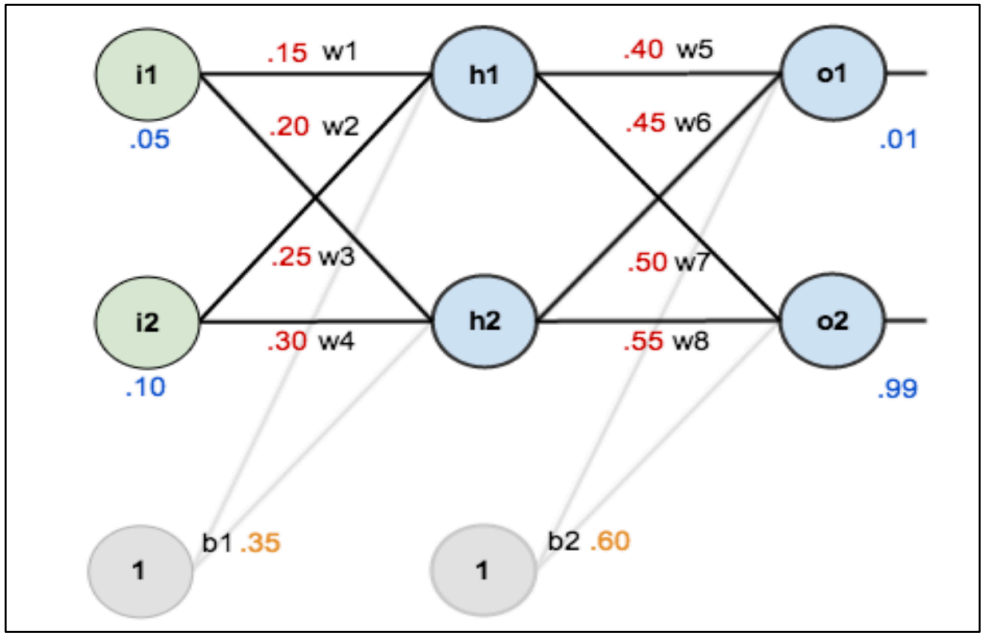
1

+

σ

h1 = 0.59326992

DNN Example (2)



w3=0.25

i1 = 0.05

*

w4=0.3

*

i2 = 0.1

+

+

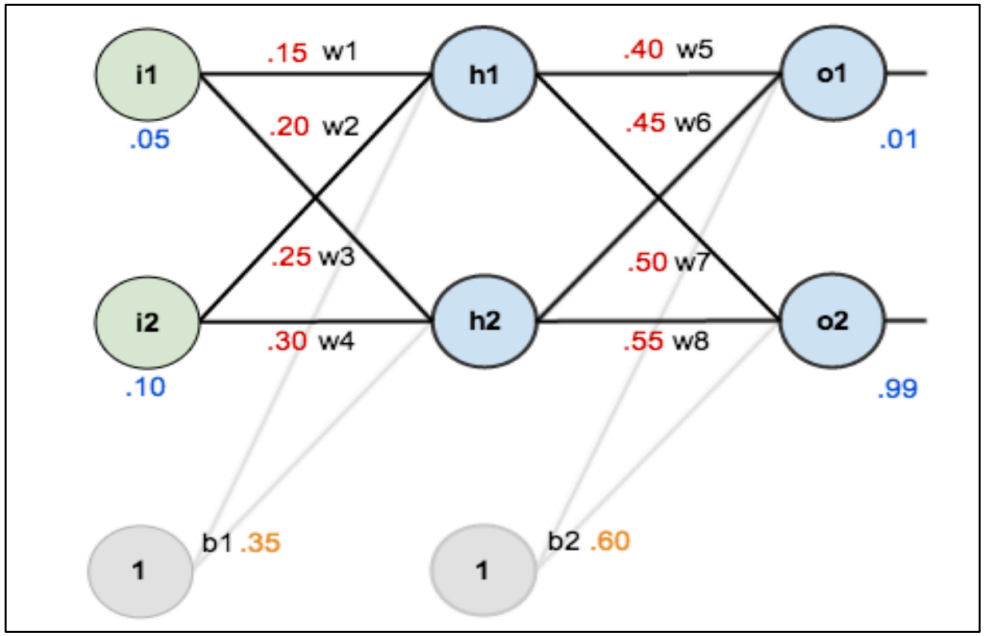
σ

h2 = 0.596884378

b1=0.35

1

DNN Example (3)



w5=0.4

h1= 0.59327

*

w6=0.45

h2 = 0.59688

*

+

+

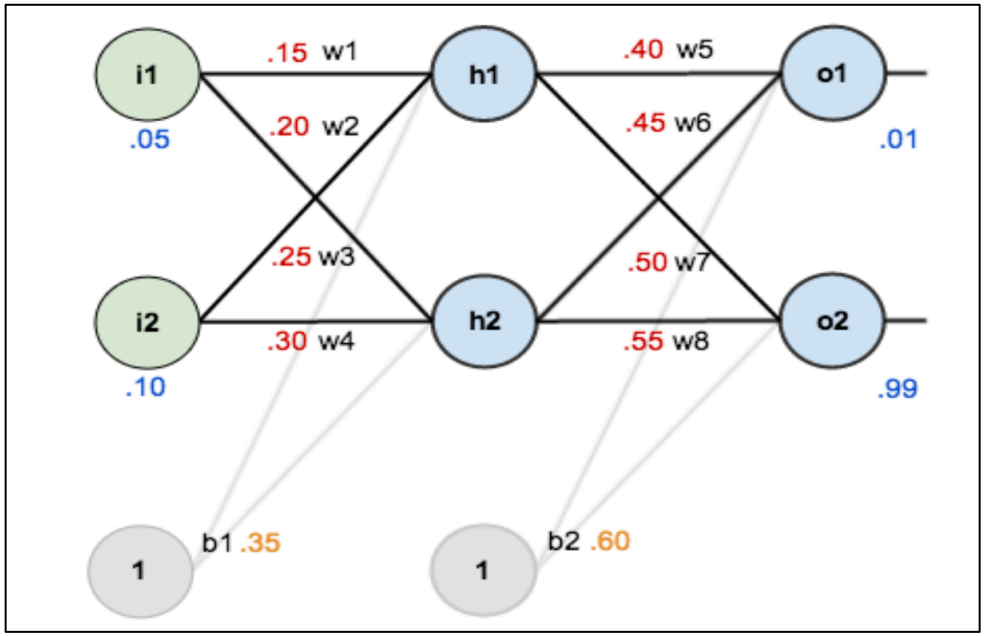
σ

o1 = 0.75136

b2=0.6

1

DNN Example (4)



w7=0.5

h1= 0.59327

*

w8=0.55

h2 = 0.59688

*

+

+

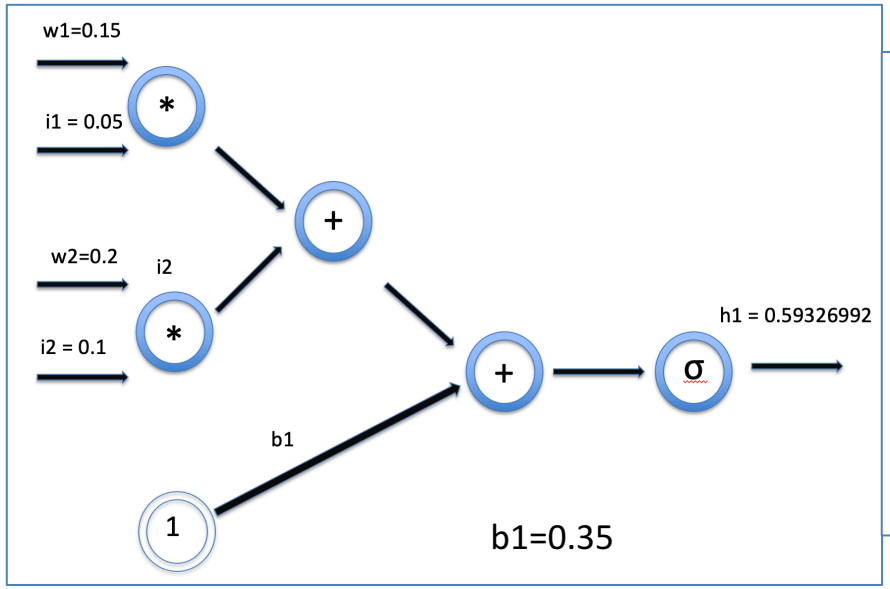
σ

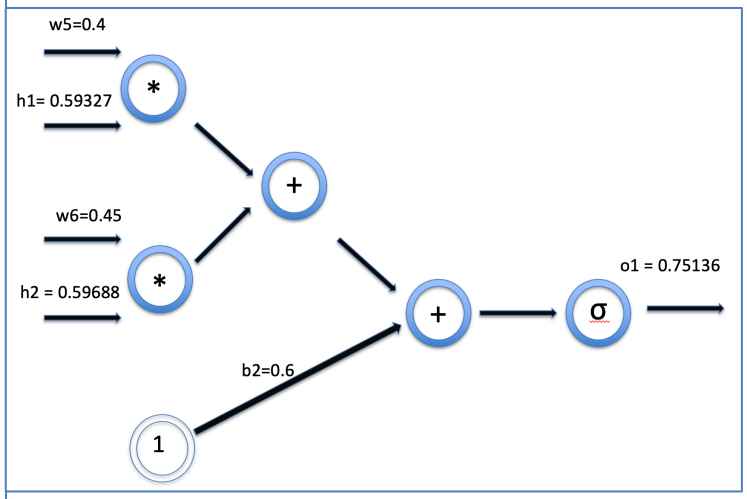
o2 = 0.77293

B2=0.6

1

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$





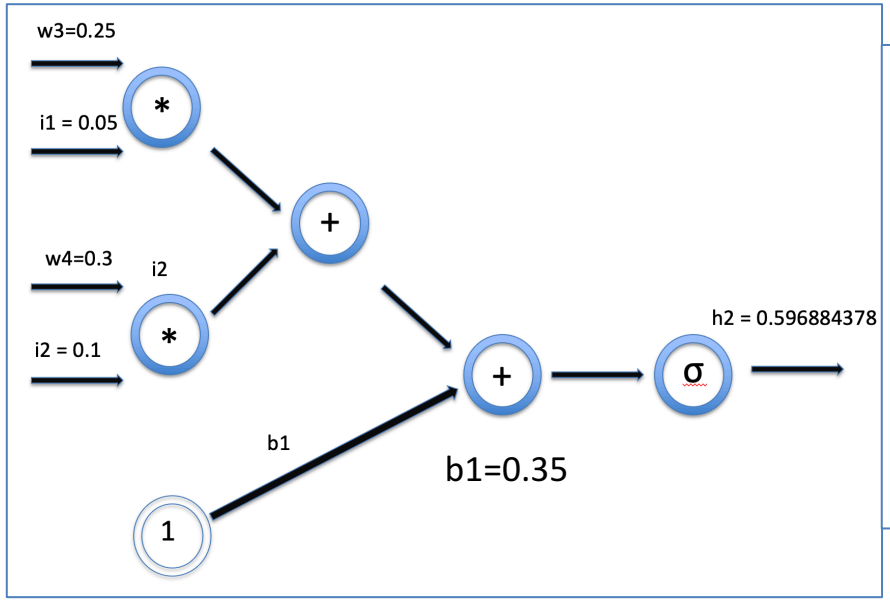
target=0.01

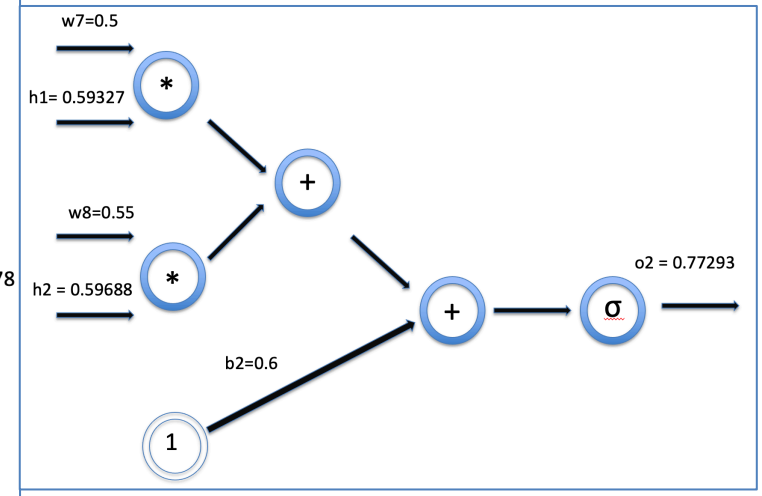
o1=0.75136

Eo1=0.27481

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Etotal = Eo1 + Eo2 = 0.298371109





Eo2=0.02356

o2=0.77293

target=0.99

$$E_{o2} = 0.023560026$$

```
# e constant
e = 2.7182818284
# initial values
i1 = 0.05
i2 = 0.10
# initial weights
w1 = 0.15
w2 = 0.20
w3 = 0.25
w4 = 0.30
w5 = 0.40
w6 = 0.45
w7 = 0.50
w8 = 0.55
# bias
b1 = 0.35
b2 = 0.60
# targets
To1 = 0.01
To2 = 0.99
```

```
# forward propagation
h1 = 1/(1+e**(-(w1*i1 + w2*i2+b1)))
print("h1: " + str(h1))

h2 = 1/(1+e**(-(w3*i1 + w4*i2+b1)))
print("h2: " + str(h2))

o1 = 1/(1+e**(-(w5*h1 + w6*h2+b2)))
print("o1: " + str(o1))

o2 = 1/(1+e**(-(w7*h1 + w8*h2+b2)))
print("o2: " + str(o2))
```

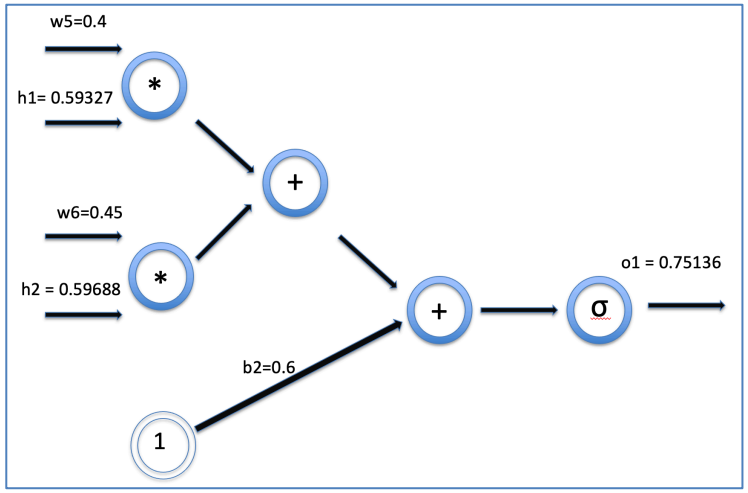
```
h1: 0.5932699921052087 h2: 0.5968843782577157 o1:
0.7513650695475076 o2: 0.772928465316421
```

```
# Error
Eo1 = 0.5*(To1-o1)**2
print("Error o1: " + str(Eo1))
Eo2 = 0.5*(To2-o2)**2
print("Error o2: " + str(Eo2))

E = Eo1 + Eo2
print("Total Error: " + str(E))
```

```
Error o1: 0.2748110831725904
Error o2: 0.023560025584942117
Total Error: 0.29837110875753253
```

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$



target=0.01 | o1=0.75136 | Eo1=0.27481

Forward Path to Objective Function

target1 = 0.01

o1 = 0.75136

Cost f

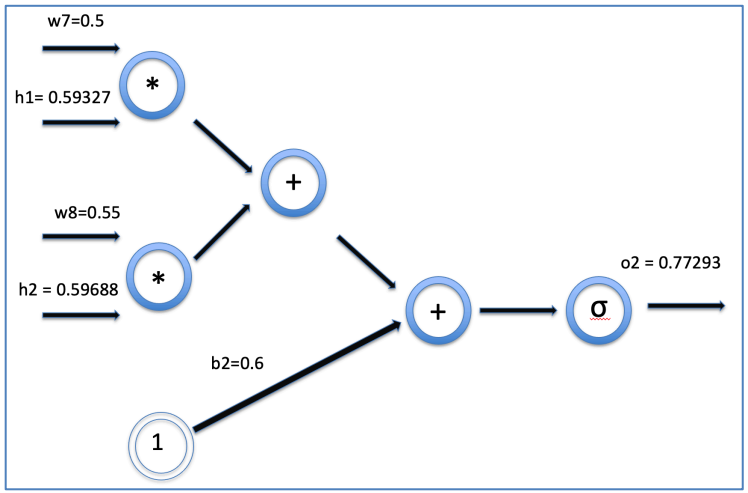
Eo1 = 0.27481

ET = 0.298371

grad ET = 1.0

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Etotal = Eo1 + Eo2 = 0.298371109



o2 = 0.77293

Cost f

Eo2 = 0.02356

target2 = 0.99

target=0.99 | o2=0.77293 | Eo2=0.02356

Training

What are the training parameters
How does it work?

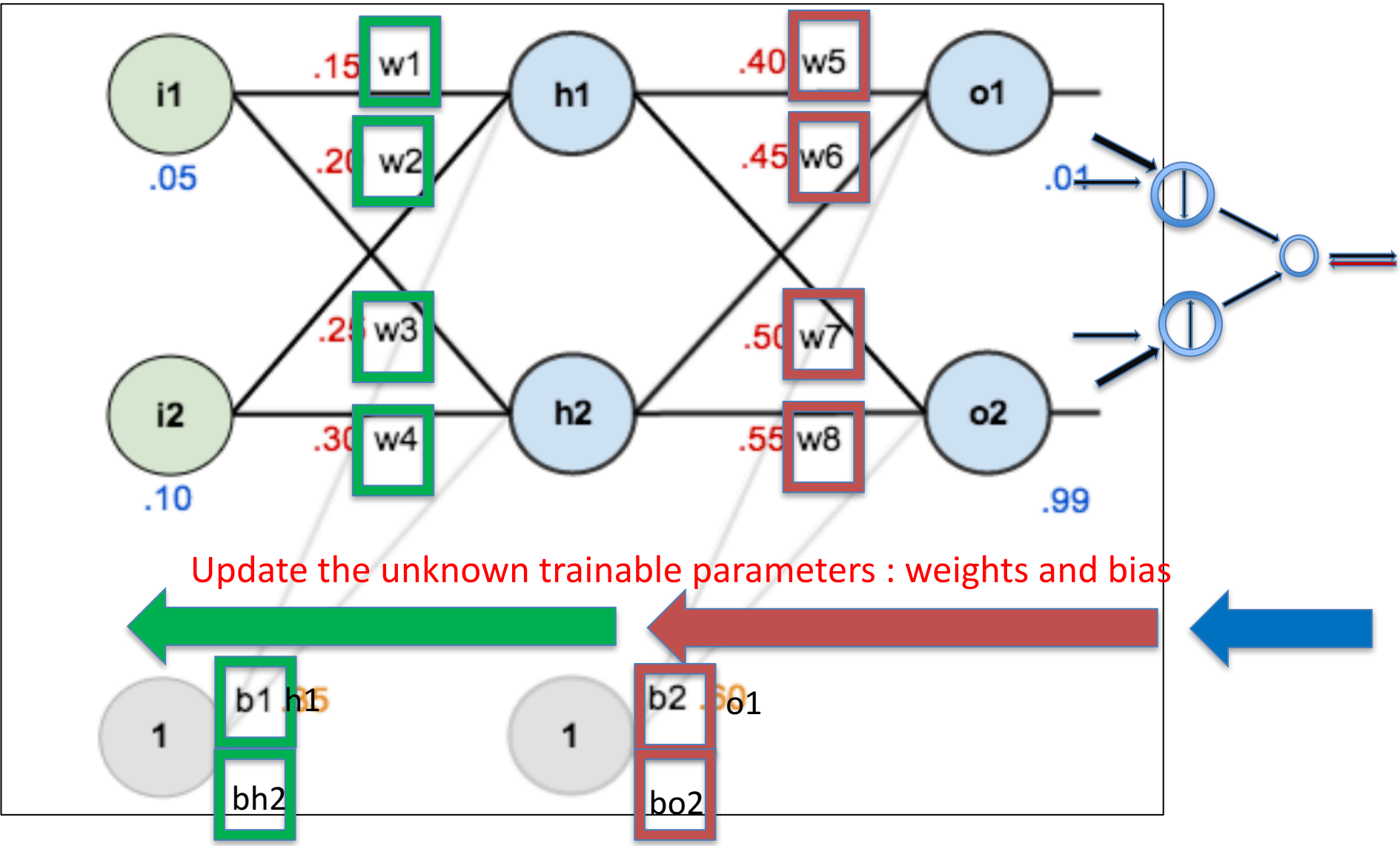
Weights and bias

Weights = w1,w2,w3,w4,w5,w6,w7,w8
= number of links connecting different layers of neurons

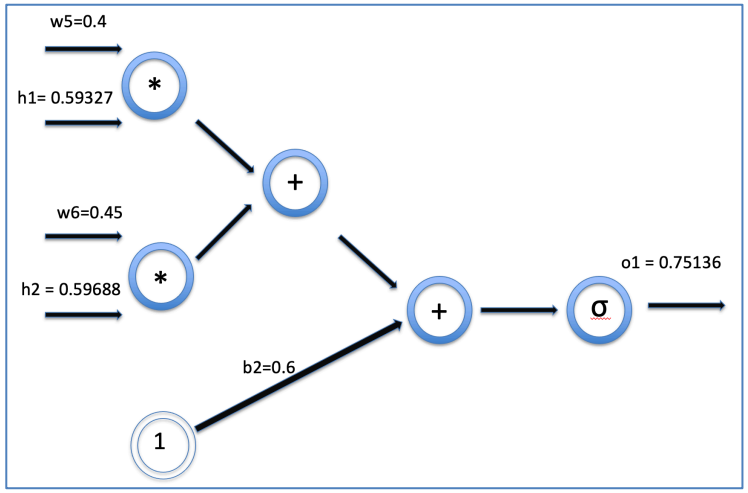
Bias = b1-h1, b1-h2, b2-o1, b2-o2 = number of neurons

Find all w+ and b+

Multilayer Perceptron : Backward Path Calculation



$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$



target=0.01 | o1=0.75136 | Eo1=0.27481

Forward Path to Objective Function

target1 = 0.01

o1 = 0.75136

Cost f

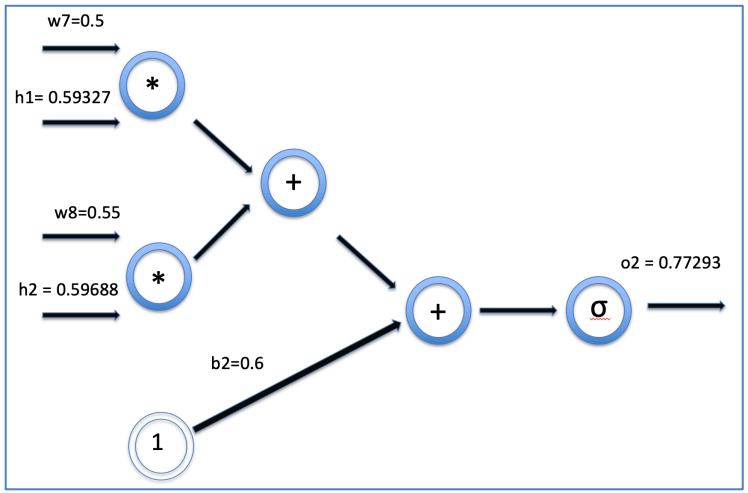
Eo1 = 0.27481

ET = 0.298371

grad ET = 1.0

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Etotal = Eo1 + Eo2 = 0.298371109



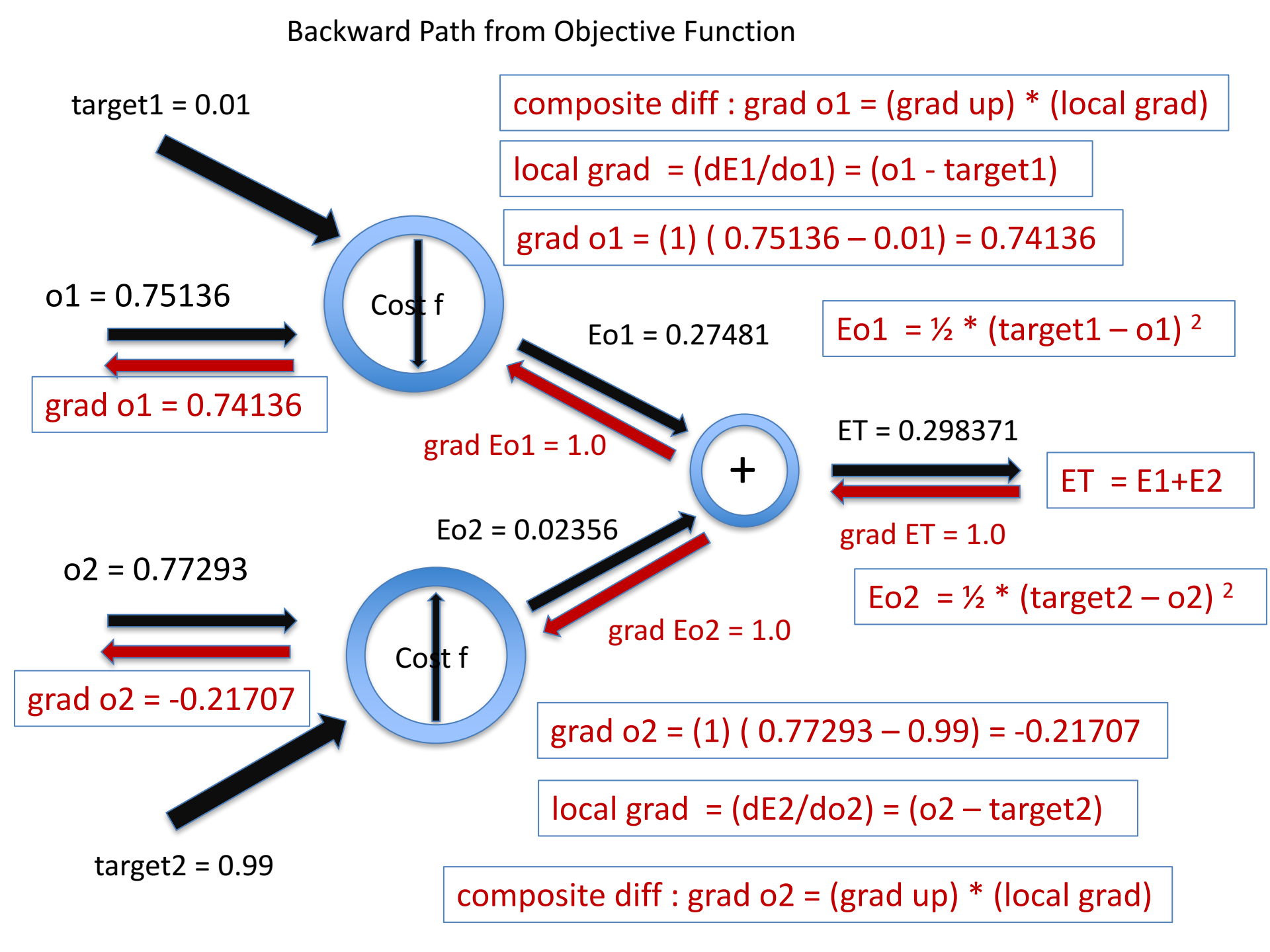
o2 = 0.77293

Cost f

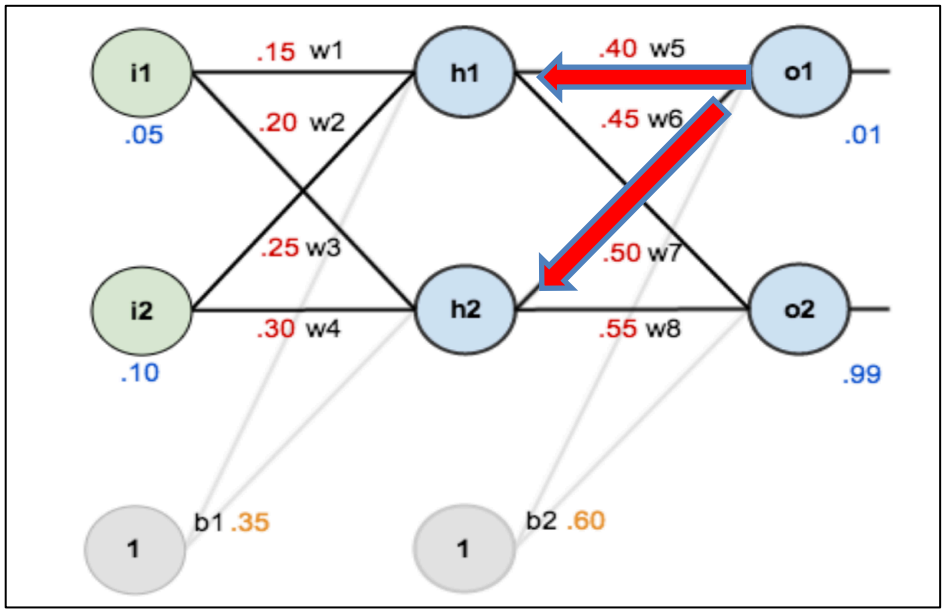
Eo2 = 0.02356

target2 = 0.99

target=0.99 | o2=0.77293 | Eo2=0.02356


Backward Path from Objective Function
target1 = 0.01
composite diff : grad o1 = (grad up) * (local grad)
local grad = (dE1/do1) = (o1 - target1)
grad o1 = (1) (0.75136 – 0.01) = 0.74136
o1 = 0.75136
Cost f
Eo1 = 0.27481
Eo1 = ½ * (target1 – o1) 2
grad o1 = 0.74136
grad Eo1 = 1.0
+
ET = 0.298371
ET = E1+E2
grad ET = 1.0
Eo2 = 0.02356
Eo2 = ½ * (target2 – o2) 2
o2 = 0.77293
grad Eo2 = 1.0
Cost f
grad o2 = -0.21707
grad o2 = (1) (0.77293 – 0.99) = -0.21707
local grad = (dE2/do2) = (o2 – target2)
target2 = 0.99
composite diff : grad o2 = (grad up) * (local grad)

Backward Path from o1



w5=0.4

h1= 0.59327

*

w6=0.45

h2 = 0.59688

*

+

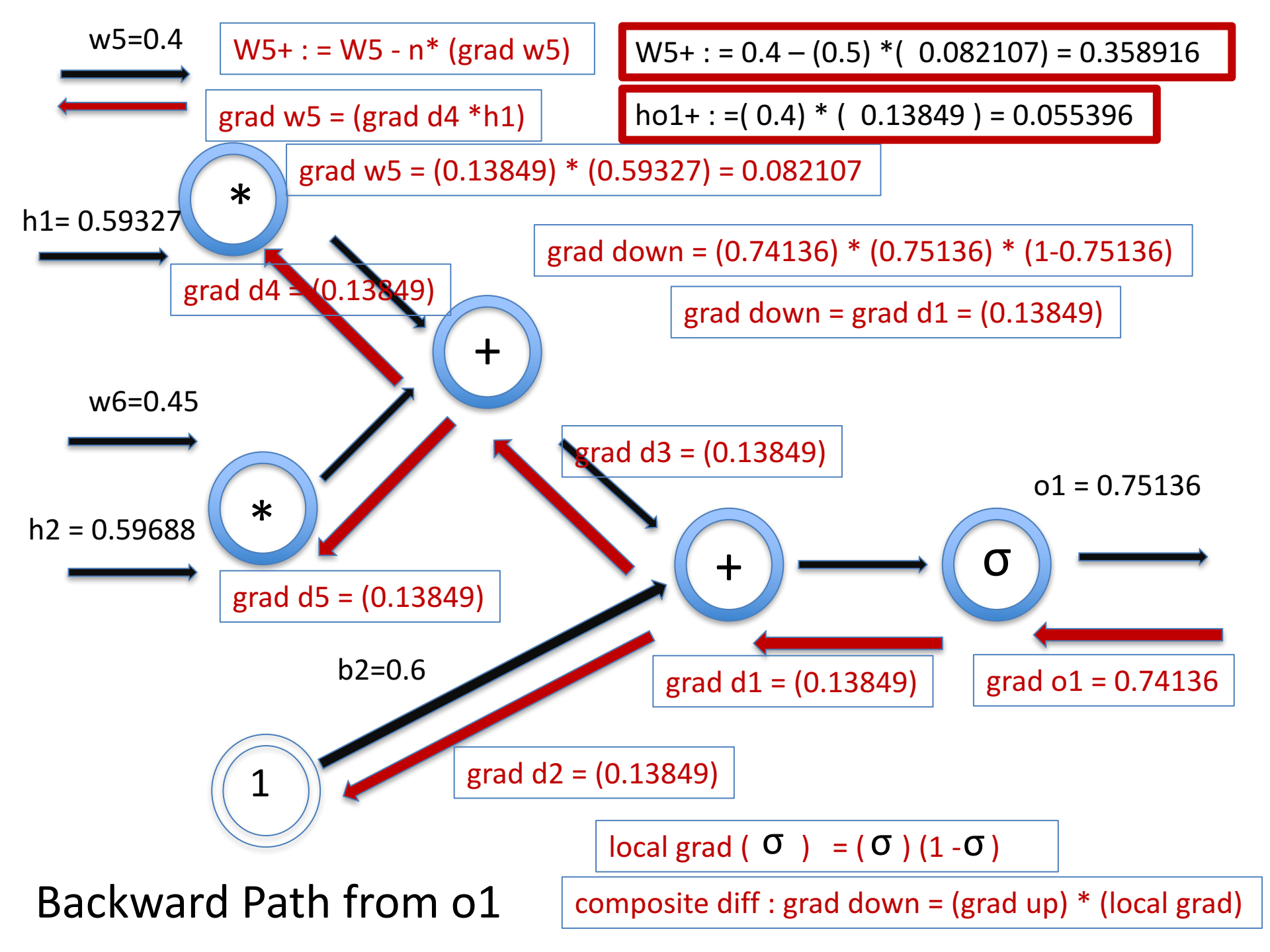
+

σ

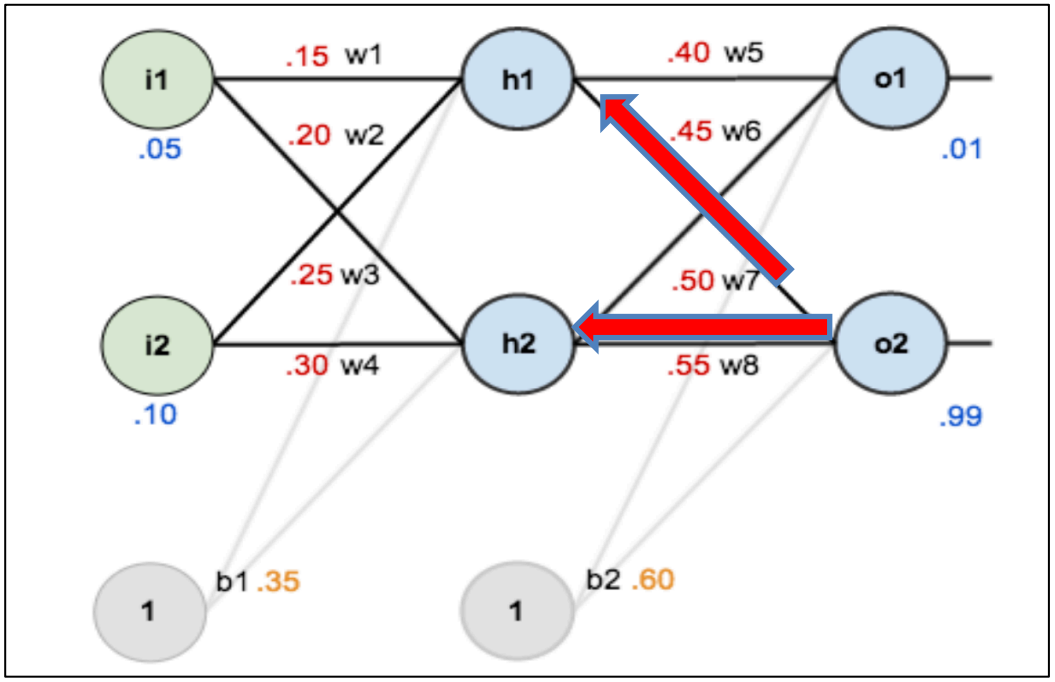
o1 = 0.75136

b2=0.6

1


w5=0.4
W5+ : = W5 - n* (grad w5)
W5+ : = 0.4 – (0.5) *(0.082107) = 0.358916
grad w5 = (grad d4 *h1)
ho1+ : =(0.4) * (0.13849) = 0.055396
grad w5 = (0.13849) * (0.59327) = 0.082107
*
h1= 0.59327
grad down = (0.74136) * (0.75136) * (1-0.75136)
grad d4 = (0.13849)
grad down = grad d1 = (0.13849)
+
w6=0.45
grad d3 = (0.13849)
o1 = 0.75136
*
h2 = 0.59688
+
σ
grad d5 = (0.13849)
b2=0.6
grad d1 = (0.13849)
grad o1 = 0.74136
1
grad d2 = (0.13849)
local grad (σ) = (σ) (1 -σ)
Backward Path from o1
composite diff : grad down = (grad up) * (local grad)

Backward Path from o2



w7=0.5

h1= 0.59327

*

w8=0.55

h2 = 0.59688

*

+

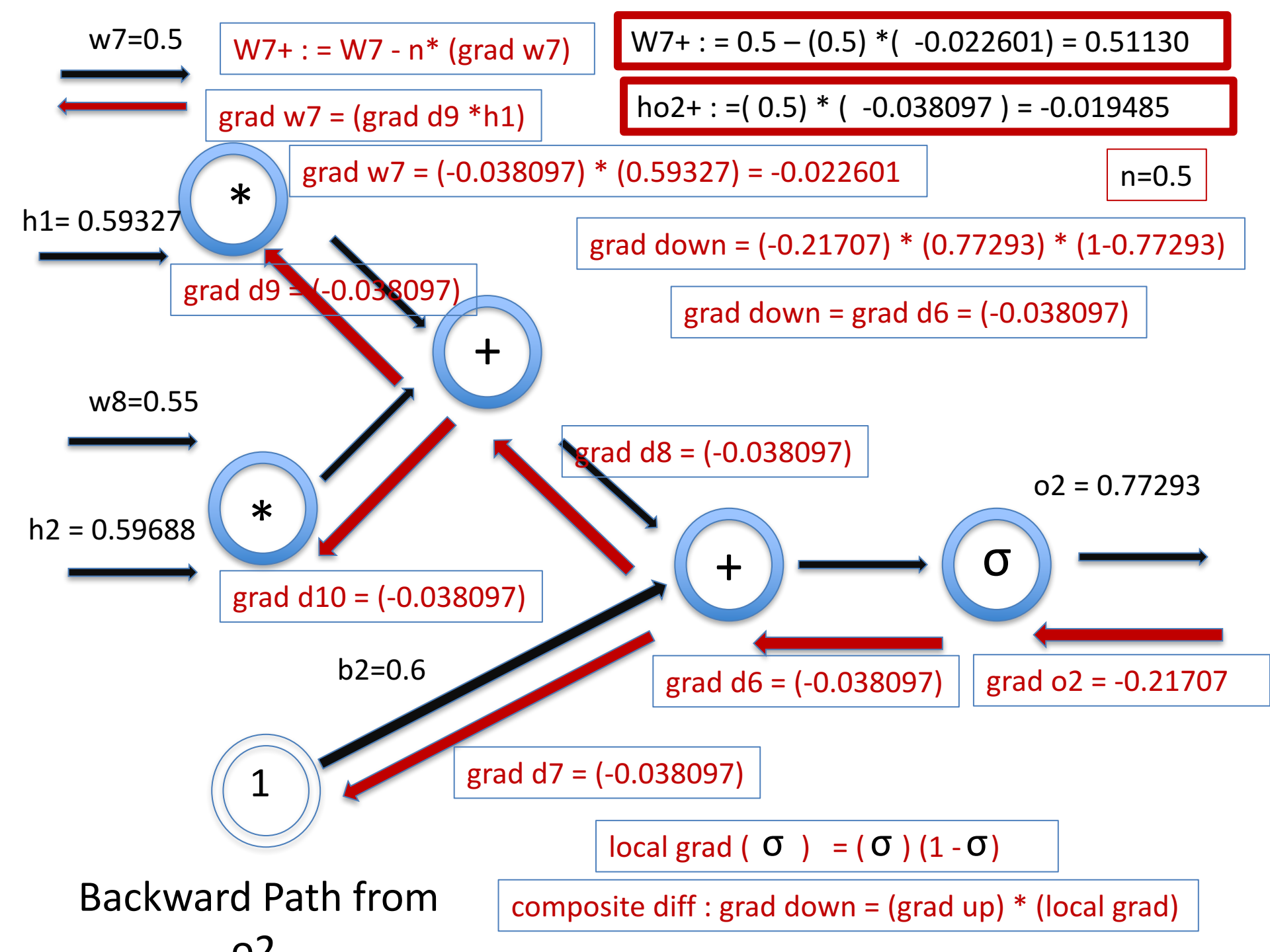
+

σ

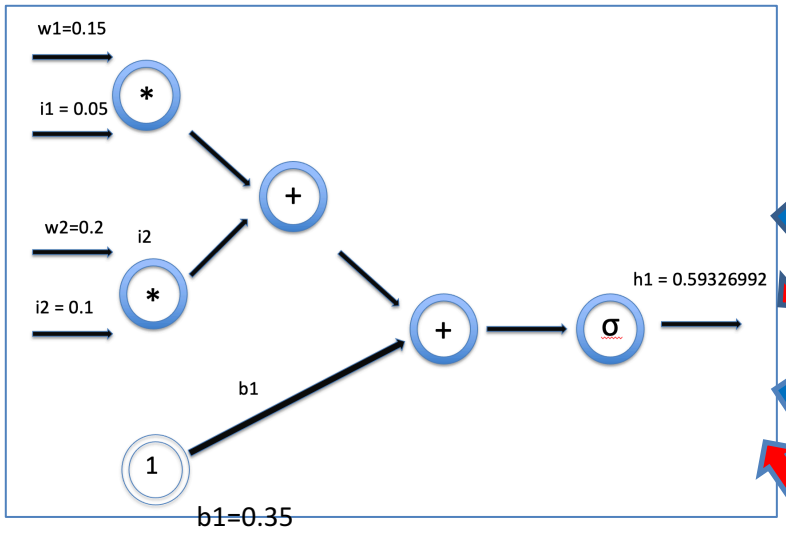
o2 = 0.77293

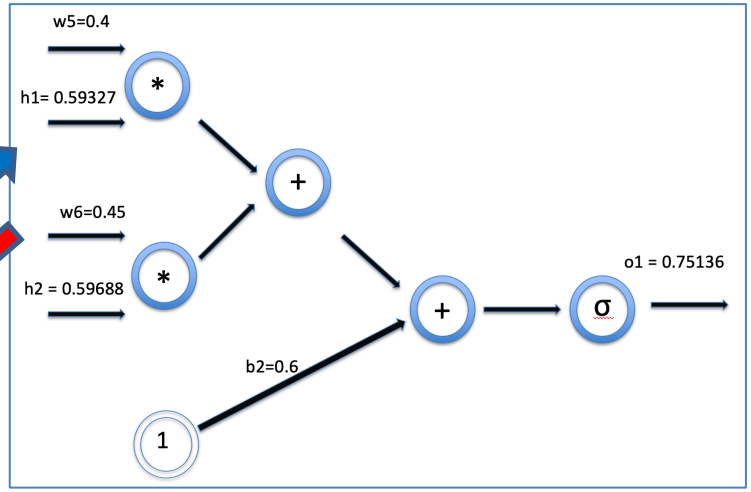
B2=0.6

1


w7=0.5
W7+ : = W7 - n* (grad w7)
W7+ : = 0.5 – (0.5) *(-0.022601) = 0.51130
grad w7 = (grad d9 *h1)
ho2+ : =(0.5) * (-0.038097) = -0.019485
grad w7 = (-0.038097) * (0.59327) = -0.022601
n=0.5
h1= 0.59327
*
grad down = (-0.21707) * (0.77293) * (1-0.77293)
grad d9 = (-0.038097)
grad down = grad d6 = (-0.038097)
+
w8=0.55
grad d8 = (-0.038097)
o2 = 0.77293
h2 = 0.59688
*
+
σ
grad d10 = (-0.038097)
b2=0.6
grad d6 = (-0.038097)
grad o2 = -0.21707
1
grad d7 = (-0.038097)
local grad (σ) = (σ) (1 - σ)
composite diff : grad down = (grad up) * (local grad)
Backward Path from o2

Backward Path from o1 and o2 to h1, from h1 to w1



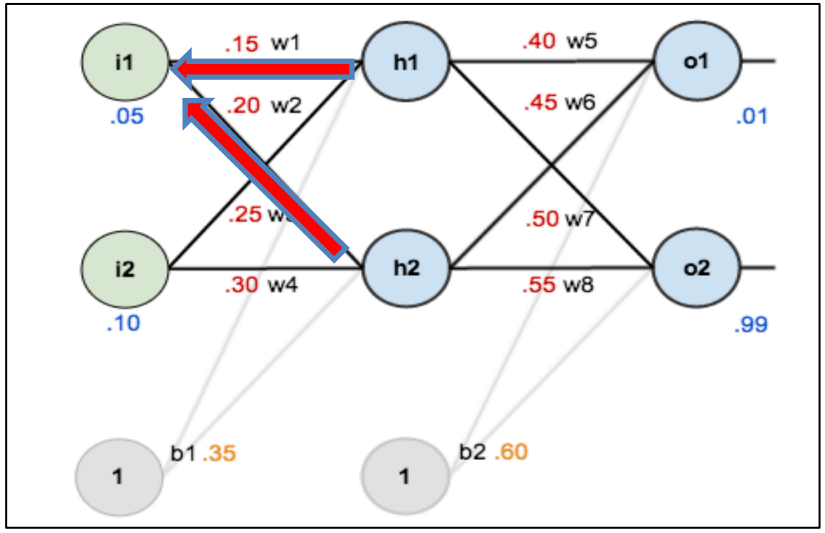


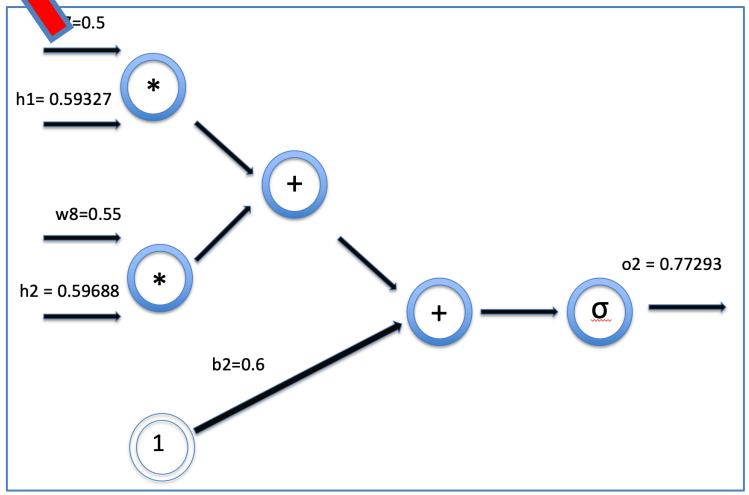
target=0.01

o1=0.75136

Eo1=0.27481

Etotal = Eo1 + Eo2 = 0.298371109

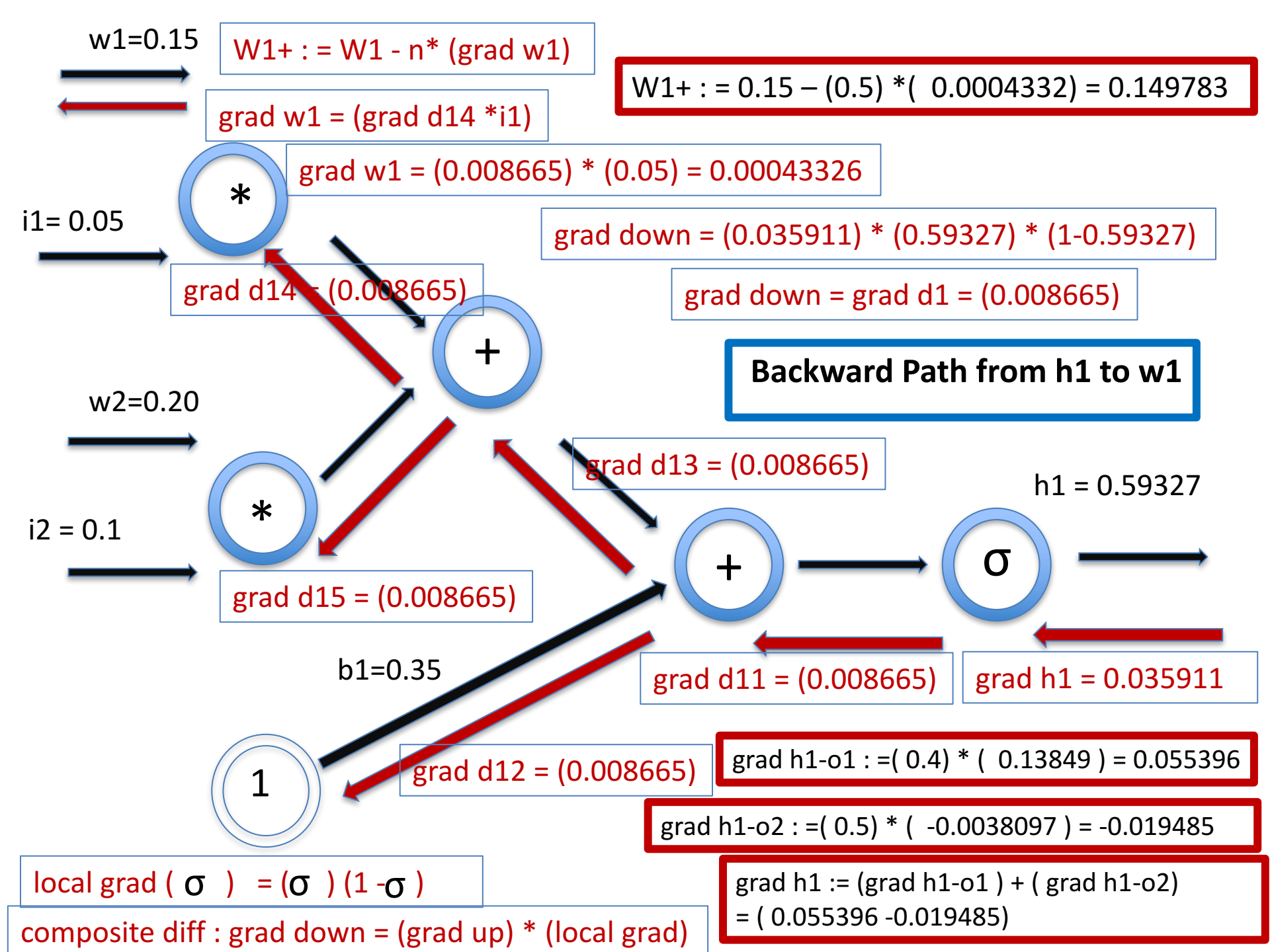




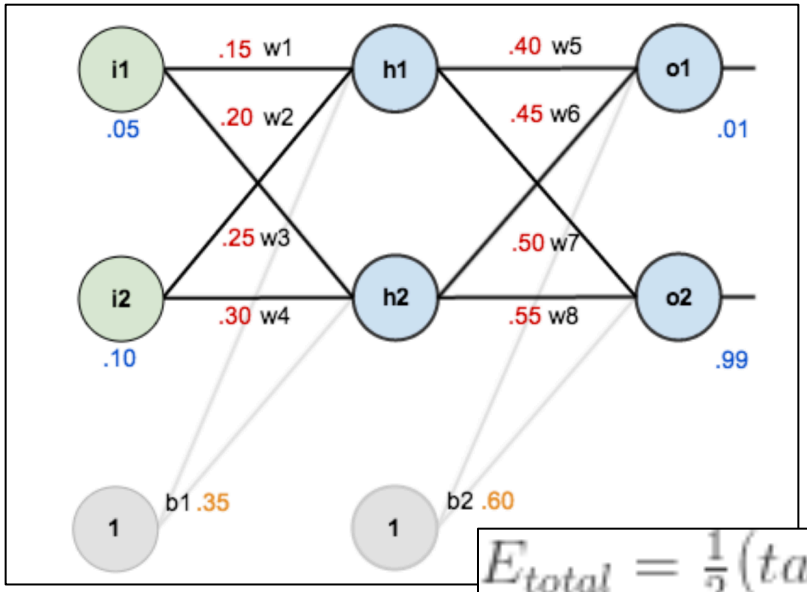
Eo2=0.02356

o2=0.77293

target=0.99

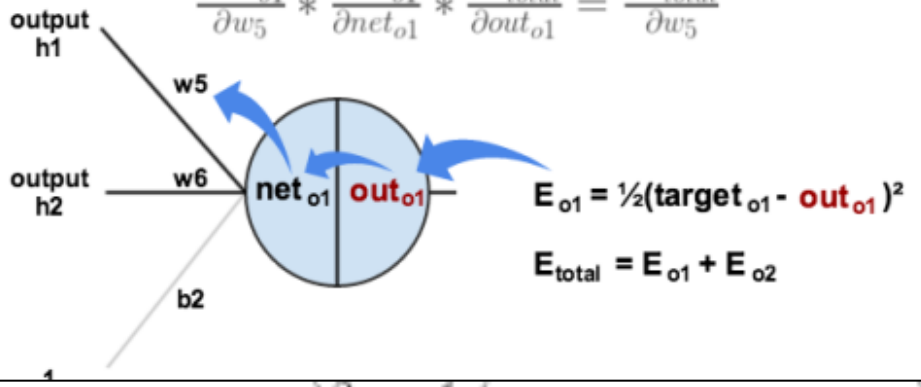

w1=0.15
W1+ : = W1 - n* (grad w1)
W1+ : = 0.15 – (0.5) *(0.0004332) = 0.149783
grad w1 = (grad d14 *i1)
grad w1 = (0.008665) * (0.05) = 0.00043326
*
i1= 0.05
grad down = (0.035911) * (0.59327) * (1-0.59327)
grad d14 = (0.008665)
grad down = grad d1 = (0.008665)
+
Backward Path from h1 to w1
w2=0.20
grad d13 = (0.008665)
h1 = 0.59327
*
i2 = 0.1
+
σ
grad d15 = (0.008665)
b1=0.35
grad d11 = (0.008665)
grad h1 = 0.035911
1
grad d12 = (0.008665)
grad h1-o1 : =(0.4) * (0.13849) = 0.055396
grad h1-o2 : =(0.5) * (-0.0038097) = -0.019485
local grad (σ) = (σ) (1 -σ)
grad h1 := (grad h1-o1) + (grad h1-o2)
= (0.055396 -0.019485)
composite diff : grad down = (grad up) * (local grad)

Backpropagation Example



$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

$$\delta_{o1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = \frac{\partial E_{total}}{\partial net_{o1}}$$

$$\delta_{o1} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$$

$$\frac{\partial E_{total}}{\partial w_5} = \delta_{o1} out_{h1}$$

$$\frac{\partial E_{total}}{\partial w_5} = -\delta_{o1} out_{h1}$$

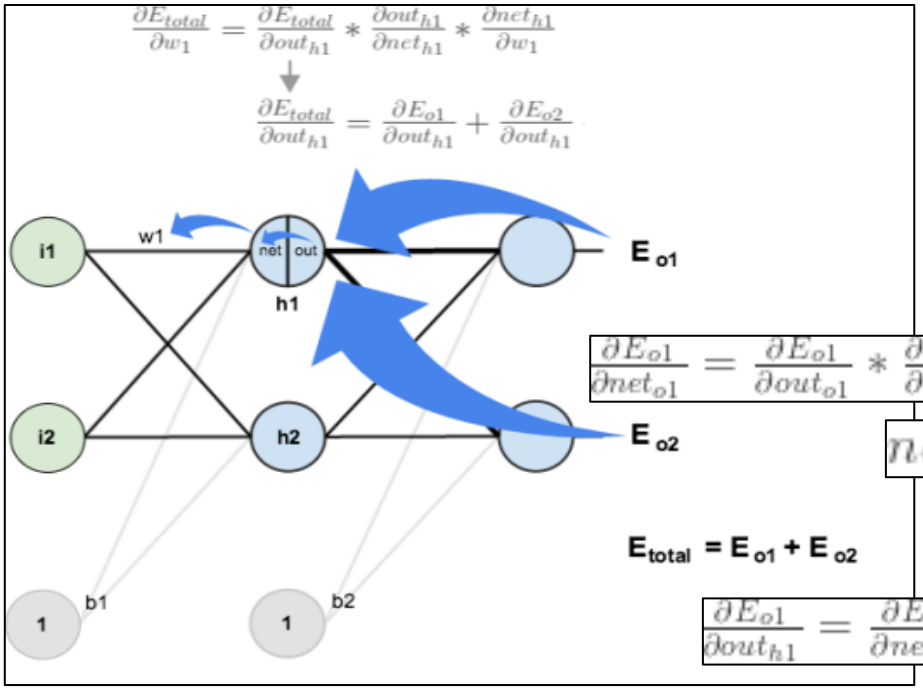
$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

Backpropagation Example



$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}}$$

$$net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \frac{\partial E_{total}}{\partial out_o} * \frac{\partial out_o}{\partial net_o} * \frac{\partial net_o}{\partial out_{h1}}\right) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = \left(\sum_o \delta_o * w_{ho}\right) * out_{h1}(1 - out_{h1}) * i_1$$

$$\frac{\partial E_{total}}{\partial w_1} = \delta_{h1} i_1$$

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

$$w_2^+ = 0.19950143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

```
# Backpropagation
n = 0.5

grad_o1 = o1 - To1
grad_o2 = o2 - To2

print("grad_o1: "+str(grad_o1))
print("grad_o2: "+str(grad_o2))

grad_d1 = (grad_o1)*o1*(1-o1)
grad_d2 = (grad_o2)*o2*(1-o2)

print("grad_d1: "+str(grad_d1))
print("grad_d2: "+str(grad_d2))

grad_w5 = grad_d1*h1
print("grad_w5: "+str(grad_w5))

w5_final = w5 - n*grad_w5

grad_w6 = grad_d1*h2
print("grad_w6: "+str(grad_w6))
w6_final = w6 - n*grad_w6

grad_w7 = grad_d2*h1
print("grad_w7: "+str(grad_w7))
w7_final = w7 - n*grad_w7

grad_w8 = grad_d2*h2
print("grad_w8: "+str(grad_w8))
w8_final = w8 - n*grad_w8
```

```
grad_h1 = w5*grad_d1 + w7*grad_d2
print("grad_h1: "+str(grad_h1))
grad_d11 = grad_h1*h1*(1-h1)
print("grad_d11: "+str(grad_d11))

grad_w1 = grad_d11*i1
print("grad_w1: "+str(grad_w1))
w1_final = w1 - n*grad_w1

grad_w2 = grad_d11*i1
print("grad_w2: "+str(grad_w2))
w2_final = w2 - n*grad_w2

grad_h2 = w6*grad_d1 + w8*grad_d2
print("grad_h2: "+str(grad_h2))
grad_d22 = grad_h2*h2*(1-h2)
print("grad_d22: "+str(grad_d22))

grad_w3 = grad_d22*i1
print("grad_w3: "+str(grad_w3))
w3_final = w3 - n*grad_w3

grad_w4 = grad_d22*i2
print("grad_w4: "+str(grad_w4))
w4_final = w4 - n*grad_w4
```

```
print("w1+: "+str(w1_final))
print("w2+: "+str(w2_final))
print("w3+: "+str(w3_final))
print("w4+: "+str(w4_final))
print("w5+: "+str(w5_final))
print("w6+: "+str(w6_final))
print("w7+: "+str(w7_final))
print("w8+: "+str(w8_final))
```

w1+: 0.1497807161327648
w2+: 0.1997807161327648 2
w3+: 0.2497511436323716 4
w4+: 0.2995022872647432 6
w5+: 0.3589164797177565 3
w6+: 0.4086661860761087
w7+: 0.5113012702391395
w8+: 0.5613701211083925

grad_o1: 0.7413650695475076
grad_o2: -0.2170715346835789 8
grad_d1: 0.1384985616294507 6
grad_d2: -0.0380982365180384 4
grad_w5: 0.0821670405644870 1
grad_w6: 0.0826676278477826 3
grad_w7: -0.0226025404782790 4
grad_w8: -0.0227402422167847 7
grad_h1: 0.0363530639276108 6
grad_d11: 0.0087713546894077 9
grad_w1: 0.0004385677344703895
grad_w2: 0.0004385677344703895
grad_h2: 0.0413703226483317 1
grad_d22: 0.0099542547051342 71
grad_w3: 0.0004977127352567136
grad_w4: 0.0009954254705134271