Raiid Ahmed Homework 2 9/20/20

Problem 1:

Given:

Example 2.8, W_2

$$W_2 = \begin{matrix} 2 & -2 & -2 & -2 & 2, \\ -2 & 2 & 0 & 2 & -2, \\ -2 & 0 & 2 & 0 & -2, \\ -2 & 2 & 0 & 2 & -2, \\ 2 & -2 & -2 & -2 & 2 \end{matrix}$$

Find:

correct response from net

w/ X or 0 w/ 10 vals missing

Confusion after 10 mistakes

Store an extra patterns see what happens with more

"Complete" X input

1 -1 -1 -1 1,
-1 1 -1* 1 -1,
-1 -1* 1 -1* 1,
-1 1 -1* 1 -1,
1 -1 -1 -1 1,

(comma denotes line termination)

→

"Incomplete" X input S_1

0 0 0 0 0,
-1 0 0 0 -1,
-1 0 1 0 1,
0 0 0 0 0,
0 0 0 0 0,

* These are the only four values that do not vary between X and O

$S_1 \cdot W_2$ returns 6.

Still a positive value, so it can be considered an X response.

Now in order to confuse the net, we will put in an incorrect X input. There will be 11 mistakes where 1 is changed to -1 and vice versa.

"Complete" X input

1 -1 -1 -1 1,
-1 1 -1 1 -1,
-1 -1 1 -1 1,
-1 1 -1 1 -1,
1 -1 -1 -1 1,

→

"Incorrect" X input S_2

1 -1* 1* 1* 1,
-1 -1* -1 -1* -1,
1 -1 1* -1 1,
-1 -1* -1 1* -1,
1 1* 1* 1* 1,

* I chose these values as my mistakes

$S_2 \cdot W_2$ returns -2

Net is clearly confused on whether to output an X or O response

We will now add an A pattern as a possible input for not X.

We will store this pattern in the weight matrix.

"Complete" A input

-1 -1 1 -1 -1,
-1 1 -1 1 -1,
-1 1 1 1 -1,
1 -1 -1 -1 1,
1 -1 -1 -1 1,

-1 →

Updated Weights W_3

3 -1 -3 -1 3,
-1 1 1 1 -1,
-1 -1 1 -1 -1,
-3 3 1 3 -3,
1 -1 -1 -1 1

bias is now -1 showing we have
1 X inputs and 2 not X inputs stored in W

With W2

X correct → 42

O correct → -42

X missing → 6

X incorrect → -2

A correct → 14

With W3

X correct → 35

O correct → -35

X missing → 3

X incorrect → 1

A correct → -11

Will now add another pattern to W

"Correct" T Pattern

1 1 1 1 1,
-1 -1 1 -1 -1,
-1 -1 1 -1 -1,
-1 -1 1 -1 -1,
-1 -1 1 -1 -1,

$\xrightarrow{-1}$

Updated Weights W_3

2 -2 -4 -2 2
0 2 0 2 0
0 0 0 0 0
-2 4 0 4 -2
2 0 -2 0 2

With W 4

X correct → 34

O correct → -34

X missing → 0

X incorrect → -10

A correct → -6

T correct → -18

All another pattern

"Correct" N Pattern

$$\begin{matrix} 1 & -1 & -1 & -1 & 1, \\ 1 & 1 & -1 & -1 & 1, \\ 1 & -1 & 1 & -1 & 1, \\ 1 & -1 & -1 & 1 & 1, \\ 1 & -1 & -1 & -1 & 1, \end{matrix}$$

$\rightarrow$

Updated Weights W_4

$$\begin{matrix} 1 & -1 & -3 & -1 & 1, \\ -1 & 1 & 1 & 3 & -1, \\ -1 & 1 & -1 & 1 & -1, \\ -3 & 5 & 1 & 3 & -3, \\ 1 & 1 & -1 & 1 & 1 \end{matrix}$$

With WS

X correct → 25

O correct → -33

X missing → 1

X incorrect → -5

A correct → -9

T correct → -11

N Correct → -11

The bias at this stage is -3, showing that we have 3 more inputs stored in W for "Not X" than we do for X.

The net's ability to interpret missing or noisy inputs seems to weaken overall if we add more patterns. Our output values decrease when a new pattern is added to W, reducing the margin of confidence.

Inputs that already gives us values close to zero will be uncertain as we add new values. However, it seems that accuracy for detecting an "X" input from missing / incomplete data can increase depending on the type of pattern we add.

For example, adding A and T pushed outputs for X missing and X incorrect in the wrong direction, but adding the N pattern pushed ouputs in the correct direction. Overall, adding patterns decreased accuracy but the size and direction of change depends on

type of patterns used to train.

Problem 2:

Given:

Example 2.11

(1,1) (1,0)

(0,1) (0,0)

Example 2.6

Find:

Show that is impossible to find w_1 and w_2 without a bias

2.6 Training data

(x_1, x_2, t)	target
(1 1 1)	1
(1 0 1)	-1
(0 1 1)	-1
(0 0 1)	-1

all have a bias

Will start training on next page

targets same

no bias

$\theta = .2$

$\alpha = 1$

Training w/ (0,0)

input	net	out	target	Δw	W (0,0)
(0,0)	0	0	-1	(0,0)	(0,0)

Cannot train on (0,0). Weights do not change and without bias, the pattern isn't even stored. Remains as if nothing happened. Boundary equations are zero.

Training w/ (1,1), (0,1), (1,0)

input	net	out	target	Δw	w
					(0,0)
(1,1)	0	0	1	(1,1)	(1,1)
(1,0)	1	1	-1	(-1,0)	(0,1)
(0,1)	1	1	-1	(0,-1)	(0,0)

end up with the initial (0, 0) weights as the beginning.

Training w/ (0,1), (1,1), (1,0)

input	net	out	target	Δw	W (0,0)
(0,1)	0	0	-1	(0,-1)	(0,-1)
(1,1)	-1	-1	1	(1,1)	(1,0)
(1,0)	1	1	-1	(-1,0)	(0,0)

Output is still (0,0), no learning occured

Training w/ (0,1), (1,0), (1,1)

input	net	out	target	Δw	W (0,0)
(0,1)	0	0	-1	(0,-1)	(0,-1)
(1,0)	0	0	-1	(-1,0)	(-1,-1)
(1,1)	-1	-1	1	(1,1)	(0,0)

No weight change, no learning

We can ran the training for multiple epochs in the example, but lack of a bias term will make each successive iteration exactly the same. All future inputs will be classified incorrectly, so it would be impossible to find w_1 or w_2 in this case.

Problem 3!

Given:

$$f = \begin{cases} 1 & net \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$f = \begin{cases} 1 & net \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

$$f = \begin{cases} 1 & net \geq \theta \\ 0 & -\theta \leq net \leq 0 \\ -1 & net \leq \theta \end{cases}$$

Find:

What can you do with each of these activation functions

$f = \begin{cases} 1 & net \geq 0 \\ 0 & otherwise \end{cases}$ This function has one target output

Considering a 2-D space, we get

$w_1 x_1 + w_2 x + b > 0$

We can only consider a problem with one "correct" target like in 2.3/Problem 1. This problem would also need to have no noise or errors, as any input with $net > 0$ →

will be considered "correct".

$f = \begin{cases} 1 & net \geq \theta \\ -1 & otherwise \end{cases}$ This function has 2 target outputs

Considering a 2-D space, we get

$$w_1 x_1 + w_2 x_2 + b \geq \theta$$
$$w_1 x_1 + w_2 x_2 + b < \theta$$

We can consider a problem with 2 "correct" or target values like 2.9 / problem 2.
The problem's input will still need to have no noise or errors, as every input will

be assigned a target value despite how close net may be to 0. This can present issues, as outputs can not be reported with confidence if there are errors or noise in the inputs.

$$f = \begin{cases} 1 & net \geq \theta \\ 0 & -\theta \leq net \leq \theta \\ -1 & net \leq \theta \end{cases}$$

This function has 2 target outputs

Considering a 2-D space, we get

$$w_1 x_1 + w_2 x_2 + b > \theta$$

$$w_1 x_1 + w_2 x_2 + b < -\theta$$

We can consider a problem with 2 "correct" or target values like 2.9 / problem 2.

Noisy / incorrect inputs can be considered as long as θ is picked

accordingly. By having an error range encompassing $(\theta, -\theta)$, we can account for noisy data in our inputs and give outputs with high confidence since our final weights will give a net $\geq \theta$ or net $\leq \theta$.

Problem 4:

Given:

XOR Function

Example 2.21

Find:

XOR implementation using a Madaline network

Inputs

n	x_1	x_2	t
1	1	1	-1
2	1	-1	1
3	-1	1	1
4	-1	-1	-1

$\alpha = .5$

Initial weights and bias

Z_1

W_{11}	W_{21}	b_1
.01	.1	.15

Z_2

W_{12}	W_{22}	b_2
.02	.1	.3

Y

V_1	V_2	b_3
.5	.5	.5

Input 1

(1, 1) :-1

Compute net input

$z_1 in = .15 + (1)(.01) + (1)(.1) = .26$

$z_2 in = .3 + (1)(.02) + (1)(.1) = .42$

$z_1 out = 1$

$z_2 out = 1$

$t = -1$

weight update

$y in = .5 + .5 + .5 = 1.5$

$y = 1$

Update weights for 3

$$W_{11} = (.01) + (.5)(-1 - .26)(1) = -.62$$

$$W_{21} = (.1) + (.5)(-1 - .26)(1) = -.53$$

$$W_{12} = (.02) + (.5)(-1 - .42)(1) = -.69$$

$$W_{22} = (.1) + (.5)(-1 - .42)(1) = -.61$$

$$b_1 = (.15) + (.5)[-1 - .26) = -.48$$

$$b_2 = (.3) + (.5)(-1 - .42) = -.41$$

New weights

Z_1

W_{11}	W_{21}	b_1
-.62	-.53	-.48

Z_2

W_{12}	W_{22}	b_2
-.69	-.61	-.41

Y

V_1	V_2	b_3
.5	.5	.5

Input 2

(1, -1) : 1

Compute net input

$Z_{1in} = -.57$

$Z_{2in} = -.49$

$y = -1$

$t = 1$

Update z_2

New weights

Z_1

W_{11}	W_{21}	b_1
-.62	-.53	-.48

Z_2

W_{12}	W_{22}	b_2
.055	-1.355	.335

Y

V_1	V_2	b_3
.5	.5	.5

Input 3

(-1, 1) : 1

$z_1 = -.39$

$z_2 = -1.055$

$y = -1$

$t = 1$

update z_1

New weights

Z_1

W_{11}	W_{21}	b_1
-1.315	.165	.215

Z_2

W_{12}	W_{22}	b_2
.055	-1.355	.335

Y

V_1	V_2	b_3
.5	.5	.5

Input 4

(-1, -1) : -1

$z_1 = 1.365$

$z_2 = 1.635$

$y = 1$

$t = -1$

update z_1 and z_2

New weights

z_1

W_{11}	W_{21}	b_1
-.1325	1.3475	-.9675

z_2

W_{12}	W_{22}	b_2
1.3725	-.0375	-.9825

y

V_1	V_2	b_3
.5	.5	.5

Train for

3 more epochs

in matlab

Final weights

z_1

W_{11}	W_{21}	b_1
-1.3119	1.2794	-1.0356

z_2

W_{12}	W_{22}	b_2
1.3206	-1.3381	-1.0344

y

V_1	V_2	b_3
.5	.5	.5

Deploy

(1,1) : -1

$z_1 = (1)(-1.3119) + (1)(1.2794) - 1.0356$

$z_2 = (1)(1.3206) + (1)(1.2794) - 1.0356$

$z_1 = -1.0681$

$z_2 = -1.0519$

$y = -1$ no error

(1,-1) : 1

$z_1 = -3.6269$

$z_2 = 1.6244$

$y = 1$ no error

(-1,1) : 1

$z_1 = 1.5556$

$z_2 = -3.6931$

$y = 1$ no error

(-1, -1) : 1

$z_1 = -1.0031$

$z_2 = -1.0169$

$Y = -1$ no error

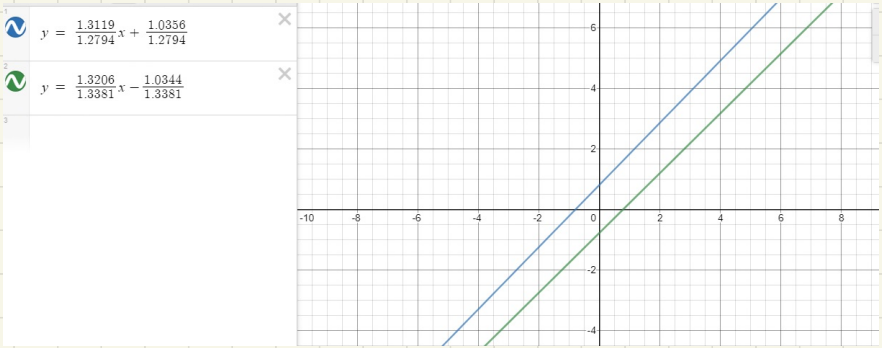
Network is trained

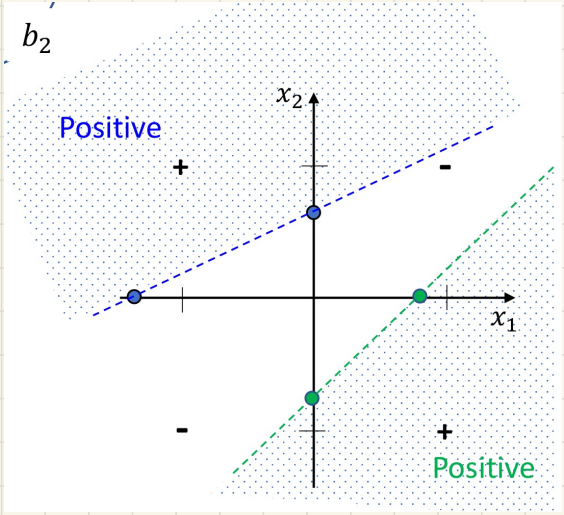
Discussion! The madeline for XOR in 2.21 had different starting weights than the one I picked. Because of that, it seems that the final weights varied a bit. The weights of Z_2 matched up with the Z_2 weights in the example give or take a bit. Z_1's weights did not match up with the example. When I loaded the initial weights from the example into my mat lab script, I got weights that matched up with the final weights for this problem.

Not sure if my script has an error or the book. The decision boundaries for my graph show 2 parallel lines, whereas the lines intersect for the example.

Example

Mine





Nevertheless, both nets were verified to be fully trained. They will both sort the prescribed inputs correctly despite having variations in weights.

Problem 5

Given:

(1, 1, 1, 1) : 1

(-1, 1, -1, -1) : 1

(1, 1, 1, -1) : -1

(1, -1, -1, 1) : -1

$\alpha = .5$

$W_{ij} = 0$

Find:

Weights that classify correctly

Weights

Z_1

w_{11}	w_{21}	w_{31}	w_{41}	b_1
0	0	0	0	0

Z_2

w_{12}	w_{22}	w_{32}	w_{42}	b_2
0	0	0	0	0

Input 1

(1, 1, 1, 1) : 1

$z_1 = 0$

$z_2 = 0$

$y = 1$

$t = 1$

No weight change

Input 2

(-1, 1, -1, -1) : 1

$z_1 = 0$

$z_2 = 0$

$y = 1$

$t = 1$

No weight change

Input 3

(1, 1, 1, -1) : -1

$z_1 = 0$

$z_2 = 0$

$y = 1$

$t = -1$

Change

weights

Z_1

w_{11}	w_{21}	w_{31}	w_{a1}	b_1
-.5	-.5	-.5	.5	-.5

Z_2

-.5	-.5	-.5	.5	-.5

Input 4

(1, -1, -1, 1) : -1

$V = .5$

$y = 1$

$t = -1$

Change

weights

z_1

w_{11}	w_{21}	w_{31}	w_{41}	b_1
-1.25	.25	.25	-.25	-1.25

z_2

w_{12}	w_{22}	w_{32}	w_{42}	b_2
-1.25	.25	.25	-.25	-1.25

Train for 3 more epochs in MatLab

Final weights

z_1

w_{11}	w_{21}	w_{31}	w_{41}	b_1
-3.1289	4.3477	.7383	3.1736	.4805

z_2

w_{12}	w_{22}	w_{32}	w_{42}	b_2
-3.1289	4.3477	.7383	3.1736	.4805

Deploy

(1,1,1,1):1

(1)(-3.1289) + (1)(4.3477) + (1)(.7383) + (1)(3.1836) + .9805

$Z_1 = Z_2 = 5.6281$

$Y = 1 \quad t = 1$ no error

(-1,1,-1,-1):1

$Z_1 = Z_2 = 4.0352$

$Y = 1 \quad t = 1$ no error

(1, 1, 1, -1) : -1

$Z_1 = Z_2 = -.7461$

$y = -1 \quad t = -1$ No error

(1, -1, -1, 1) : 1

$Z_1 = Z_2 = -4.5508$

$y = -1 \quad t = -1$ No error

Network is fully trained

Discussion:

Normally I would select an Adaline, but to save time I just used my Madaline script. Z_1 and Z_2 will not vary because of how we started all zeros for weights and not random values.