



Seoul National University
College of Engineering
Department of Naval Architecture and Ocean Engineering
1, Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea

Fall 2020

딤러닝

PA # 2

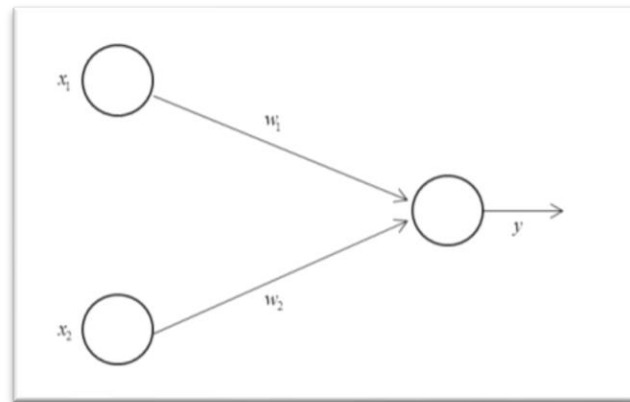
Instructor name	김태완 교수님
Student name	이용준
Department	조선해양공학과
Student ID	2015-19595
Submission date	2020.10.11

Contents

1. Problem Definition	3
1.1 Problem 1	3
1.2 Problem 2	3
1.3 Problem 3	3
1.4 Problem 4	3
1.5 Problem 5	4
1.6 Problem 6	4
1.7 Problem 7	4
 2. Problem Analysis and Design	 4
2.1 Analysis	4
2.2 Data Flow Diagram	6
 3. Code Explanation	 8
3.1 Class	8
3.2 Function	8
 4. Conclusion	 11
4.1 Result	11
4.2 Conclusion	14

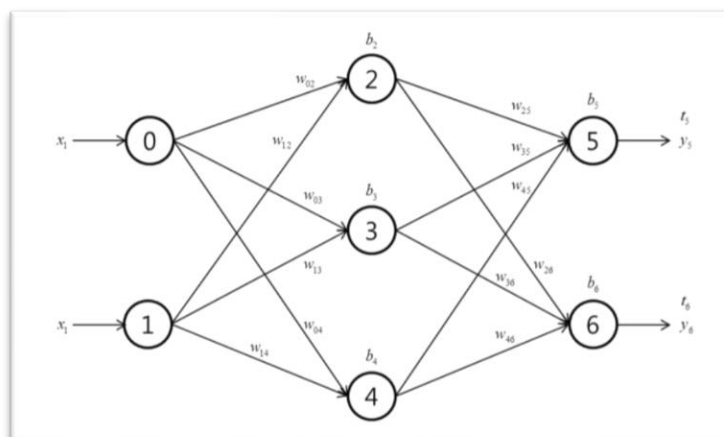
1. Problem Definition

1.1 Problem 1



- Perceptron에서 w_1, w_2 를 계산하라. 또한 이를 Python으로 작성 후 그 결과를 비교하라.

1.2 Problem 2



- Backpropagation을 실행하고 결과를 정리하시오. 또한 이를 Python으로 작성 후 그 결과를 비교하라.

1.3 Problem 3

- Backpropagation을 한 번 수행하고 결과를 정리하시오. 또한 이를 Python으로 작성 후 그 결과를 비교하라

1.4 Problem 4

- Logistic error function으로 backpropagation을 한 번 수행하고 결과를 정리하시오. 또한 이를 Python으로 작성 후 그 결과를 비교하라.

1.5 Problem 5

- “SSE”로 error 정의, hidden layer 2개일 때 backpropagation을 1번 수행하고 정리하시오. 또한 이를 Python으로 작성 후 그 결과를 비교하시오.

1.6 Problem 6

- “Logistic error function”로 error 정의 hidden layer가 1개일 때 1번 backpropagation을 수행하고 정리하라. 또한 이를 Python으로 작성 후 그 결과를 비교하라.

1.7 Problem 7

- “SSE”로 error를 정의, hidden layer 2개일 때 1번 backpropagation을 수행하고 결과를 정리하라. 또한 이를 Python으로 작성 후 그 결과를 비교하라.

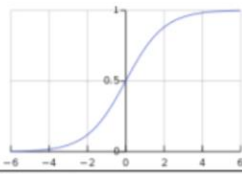
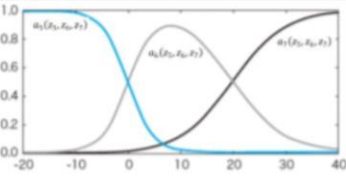
2. Problem Analysis and Design

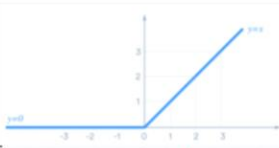

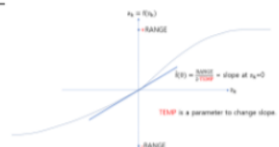
2.1 Analysis

- Error Function

Error of one node	
Cost	Equation
Sum of squared error (SSE)	$C_s = \frac{1}{2} (y_s - a_s)^2$
Cross entropy cost function (CECF)	$C_s = -[y_s \ln a_s + (1 - y_s) \ln(1 - a_s)]$
Logistic error	$C_s = -y_s \ln a_s$

- Activation Function

Sigmoid (standard logistic)	$a(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$		$\hat{a}(z) = \hat{\sigma}(z) = \sigma(1 - \sigma)$ $\frac{da_i(z_i)}{dz_i} = a_i(1 - a_i)$
Softmax	$a_5(z_5, z_6, z_7) = \frac{e^{z_5}}{e^{z_5} + e^{z_6} + e^{z_7}}$ Note: $a_5 + a_6 + a_7 = 1$		If $i = j$: $\frac{\partial a_i(z_{j_0}, \dots, z_{j_{N-1}})}{\partial z_i} = a_i(1 - a_i)$ If $i \neq j$: $\frac{\partial a_i(z_{j_0}, \dots, z_{j_{N-1}})}{\partial z_i} = -a_i a_j$

ReLU: rectified linear unit (ReLU)	$a(z) = \max(z, 0)$ $= \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases}$		$\hat{a}(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$ $\frac{da_i(z_i)}{dz_i} = \text{iif } (z_i > 0, 1, 0)$
Leaky ReLU	$f(\alpha, z) = \begin{cases} \alpha z & \text{for } z < 0 \\ z & \text{for } z \geq 0 \end{cases}$		$f(\alpha, z) = \begin{cases} \alpha & \text{for } z < 0 \\ 1 & \text{for } z \geq 0 \end{cases}$ $\frac{da_i(z_i)}{dz_i} = \text{iif } (z < 0, \alpha, 1)$
'general' logistic	$f(z) = \frac{2 \cdot \text{RANGE}}{1 + e^{-\frac{z}{\text{TEMP}}}} - \frac{1}{2}$		$\hat{f}(z) = \frac{2 \cdot \text{RANGE}}{\text{TEMP}} \left[\frac{1}{4} - \left(\frac{f(z)}{2 \cdot \text{RANGE}} \right)^2 \right]$ $\frac{da_i(z_i)}{dz_i} = \frac{2 \cdot \text{RANGE}}{\text{TEMP}} \left[\frac{1}{4} - \left(\frac{a_i}{2 \cdot \text{RANGE}} \right)^2 \right]$

- 이중 분류의 문제를 풀기 위해서 hidden layer의 활성화 함수로 sigmoid를, output layer의 활성화 함수로 softmax 함수를 사용하기로 하였다.

Activation function of output layer	Cost(error) function	δ_k
Identity	SSE	$\delta_5 = a_5 - y_5$
Sigmoid	SSE	$\delta_5 = (a_5 - y_5)a_5(1 - a_5)$
Sigmoid	CECF	$\delta_5 = a_5 - y_5$
Softmax	SSE	$\delta_5 = (a_5 - y_5)a_5(1 - a_5) + (a_6 - y_6)a_6(-a_5) + (a_7 - y_7)a_7(-a_5)$
Softmax	Logistic	$\delta_5 = a_5 - y_5$

- Backpropagation

의미: 오차(δ_k)를 weight의 크기에 비례하게 배분하여 역방향으로 signal을 보낸다.

따라서 오차(δ_k)를 구하는 것이 핵심이다.

그 이유는 다음과 같다.

구하고자 하는 최적해는 $\mathbf{w}^*, \mathbf{b}^*$ 이다.

역방향 j layer와 k layer의 \mathbf{w}, \mathbf{b} update:

$$w_{jk}^{new} \leftarrow w_{jk}^{old} - \eta \nabla w_{jk} = w_{jk}^{old} - \eta a_j \delta_k$$

$$b_k^{new} \leftarrow b_k^{old} - \eta \nabla b_k = b_k^{old} - \eta \delta_k$$

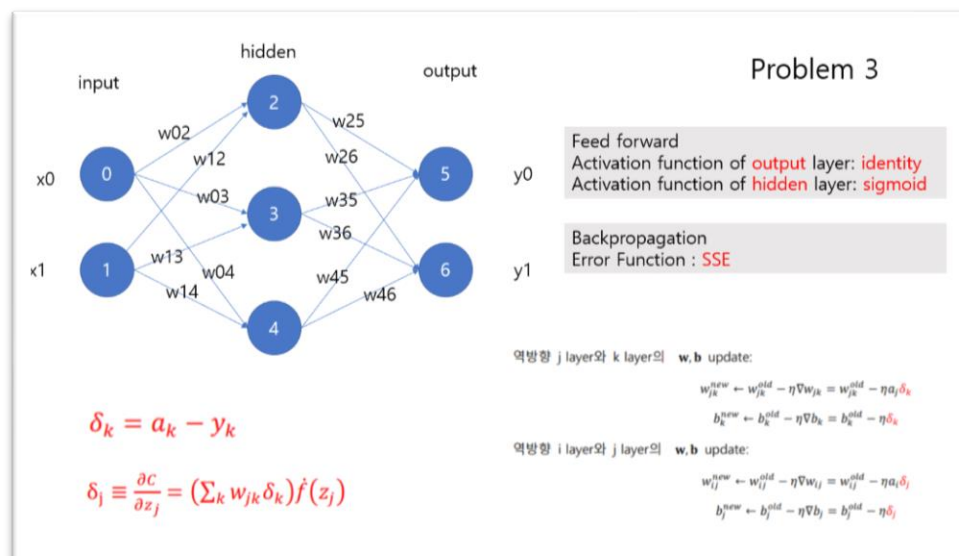
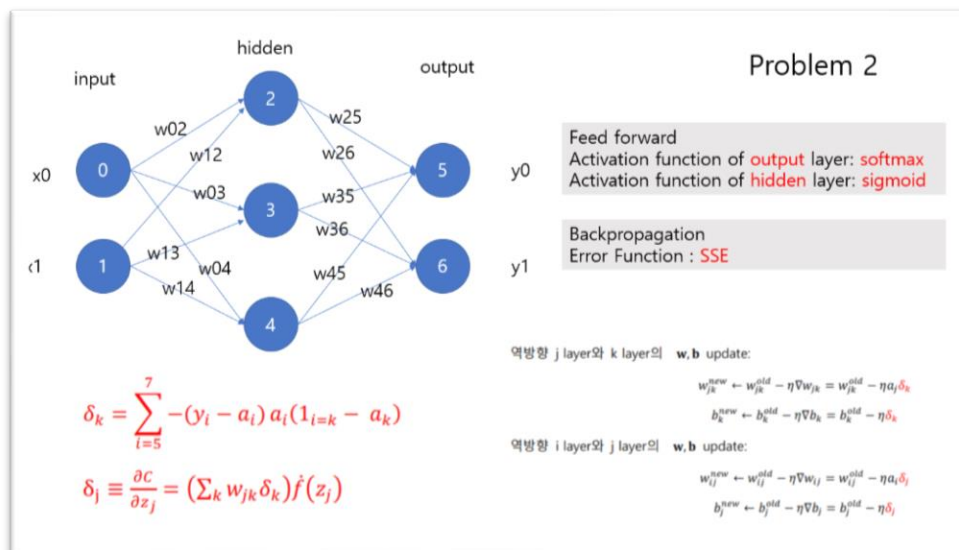
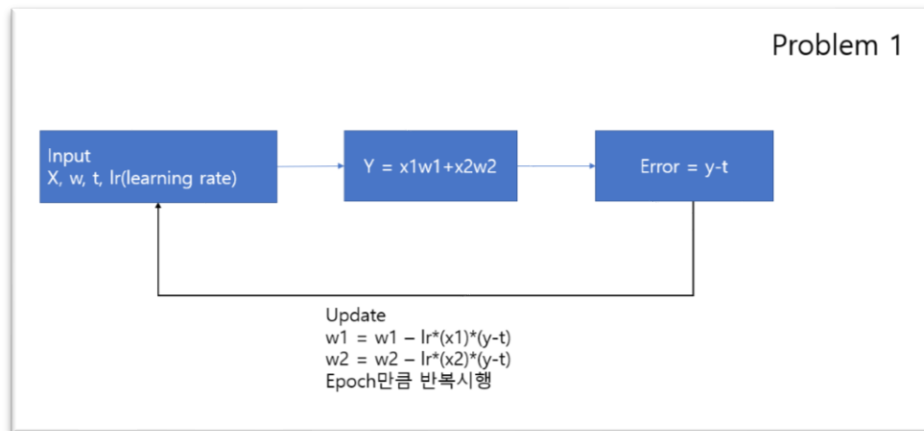
역방향 i layer와 j layer의 \mathbf{w}, \mathbf{b} update:

$$w_{ij}^{new} \leftarrow w_{ij}^{old} - \eta \nabla w_{ij} = w_{ij}^{old} - \eta a_i \delta_j$$

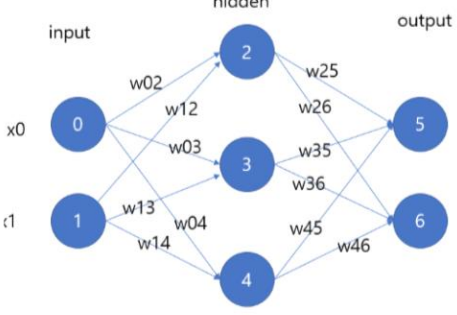
$$b_j^{new} \leftarrow b_j^{old} - \eta \nabla b_j = b_j^{old} - \eta \delta_j$$

2.2 Data Flow Diagram

각 문제에 대한 DFD이다.



Problem 4



Feed forward
Activation function of **output** layer: **softmax**
Activation function of **hidden** layer: **sigmoid**

Backpropagation
Error Function : **SSE**

역방향 j layer와 k layer의 **w, b** update:

$$w_{jk}^{new} \leftarrow w_{jk}^{old} - \eta \nabla w_{jk} = w_{jk}^{old} - \eta a_j \delta_k$$

$$b_k^{new} \leftarrow b_k^{old} - \eta \nabla b_k = b_k^{old} - \eta \delta_k$$

역방향 i layer와 j layer의 **w, b** update:

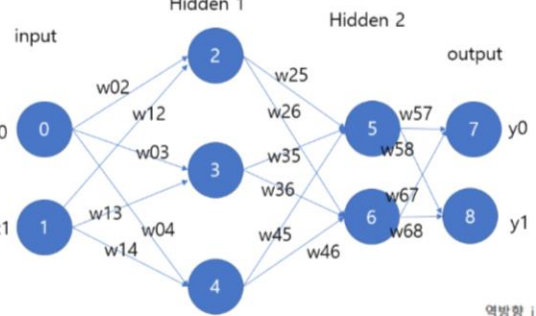
$$w_{ij}^{new} \leftarrow w_{ij}^{old} - \eta \nabla w_{ij} = w_{ij}^{old} - \eta a_i \delta_j$$

$$b_j^{new} \leftarrow b_j^{old} - \eta \nabla b_j = b_j^{old} - \eta \delta_j$$

$$\delta_k = \sum_{i=5}^7 -(y_i - a_i) a_i (1 - a_k)$$

$$\delta_j \equiv \frac{\partial C}{\partial z_j} = (\sum_k w_{jk} \delta_k) f'(z_j)$$

Problem 5



Feed forward
Activation function of **output** layer: **softmax**
Activation function of **hidden** layer: **sigmoid**

Backpropagation
Error Function : **SSE**

역방향 j layer와 k layer의 **w, b** update:

$$w_{jk}^{new} \leftarrow w_{jk}^{old} - \eta \nabla w_{jk} = w_{jk}^{old} - \eta a_j \delta_k$$

$$b_k^{new} \leftarrow b_k^{old} - \eta \nabla b_k = b_k^{old} - \eta \delta_k$$

역방향 i layer와 j layer의 **w, b** update:

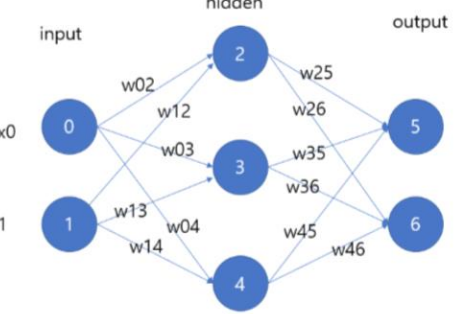
$$w_{ij}^{new} \leftarrow w_{ij}^{old} - \eta \nabla w_{ij} = w_{ij}^{old} - \eta a_i \delta_j$$

$$b_j^{new} \leftarrow b_j^{old} - \eta \nabla b_j = b_j^{old} - \eta \delta_j$$

Output delta $\delta_k = \sum_{i=5}^7 -(y_i - a_i) a_i (1 - a_k)$

Hidden delta $\delta_j \equiv \frac{\partial C}{\partial z_j} = (\sum_k w_{jk} \delta_k) f'(z_j)$

Problem 6



Feed forward
Activation function of **output** layer: **softmax**
Activation function of **hidden** layer: **sigmoid**

Backpropagation
Error Function : **Logistic Error Function**

역방향 j layer와 k layer의 **w, b** update:

$$w_{jk}^{new} \leftarrow w_{jk}^{old} - \eta \nabla w_{jk} = w_{jk}^{old} - \eta a_j \delta_k$$

$$b_k^{new} \leftarrow b_k^{old} - \eta \nabla b_k = b_k^{old} - \eta \delta_k$$

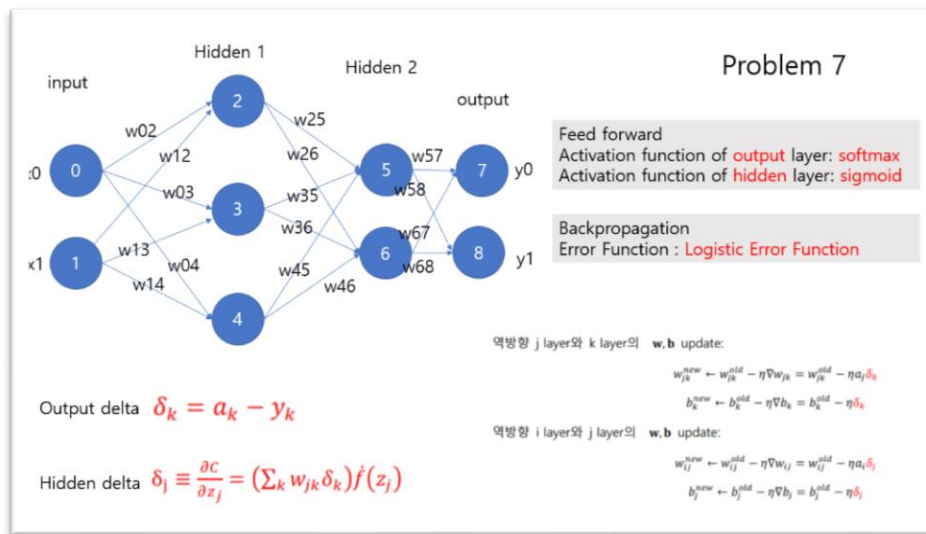
역방향 i layer와 j layer의 **w, b** update:

$$w_{ij}^{new} \leftarrow w_{ij}^{old} - \eta \nabla w_{ij} = w_{ij}^{old} - \eta a_i \delta_j$$

$$b_j^{new} \leftarrow b_j^{old} - \eta \nabla b_j = b_j^{old} - \eta \delta_j$$

$$\delta_k = a_k - y_k$$

$$\delta_j \equiv \frac{\partial C}{\partial z_j} = (\sum_k w_{jk} \delta_k) f'(z_j)$$



3. Code Explanation

3.1 Class

프로젝트에 사용된 클래스에 관한 설명이다. 이번 과제에서는 클래스는 거의 사용하지 않았다.

Class Name	Explanation
<code>activate_function</code>	여러 활성화 함수를 담고 있는 클래스

3.2 Function

과제에 사용된 함수에 대한 설명이다.

<code>perceptron(x, w, t, lr, epoch)</code>		
Parameter	Type	Explanation
x	Array	Input value
w	Array	weight
t	float	Target value
lr	float	Learning rate
epoch	int	Number of training

<code>sigmoid_function(z)</code>		
Parameter	Type	Explanation
z	float	활성화할 z 값

softmax(z1,z2, index)		
Parameter	Type	Explanation
Z1	float	활성화할 z1 값
Z2	float	활성화할 z2 값
Index	Int	Softmax 함수의 분자를 결정하는 인덱스

identity(z)		
Parameter	Type	Explanation
z	float	활성화할 z 값

ReLU(z)		
Parameter	Type	Explanation
z	float	활성화할 z 값

feed_forward(num_input ,num_hidden, num_output, x, w_ij, b_j, w_jk,b_k,activate_option_in_output_layer)		
Parameter	Type	Explanation
num_input	Int	Input layer의 node 개수
num_hidden	Int	hidden layer의 node 개수
num_output	Int	output layer의 node 개수
x	Array	Input value
w_ij	Array	weight
b_j	float	bias
w_jk	Array	weight
b_k	float	bias
activate_option_in_output_layer	int	Output layer에서의 활성화함수를 선택하는 인덱스 0이면 softmax, 1이면 identity

backpropagation(x, w_ij,b_j,w_jk,b_k,hidden_a,output_a,target,lr,error_function_option)		
Parameter	Type	Explanation
x	Array	Input value
w_ij	Array	weight
b_j	float	bias
w_jk	Array	weight
b_k	float	bias

hidden_a	array	Hidden layer의 a값들을 배열로 저장
output_a	Array	output layer의 a값들을 배열로 저장
target	array	Target value
lr	Float	Learning rate
error_function_option	int	Error 함수를 선택하는 인덱스

다음 함수들은 hidden layer가 2개인 문제에서 사용할 함수들이다.

feed_forward_2_hidden_layer(x,w_ij,b_j,w_jk,b_k,w_kl,b_l)		
Parameter	Type	Explanation
x	Array	Input value
w_ij	Array	weight
b_j	float	bias
w_jk	Array	weight
b_k	float	bias
w_kl	Array	weight
b_l	float	bias

Backpropagation2(x,w_ij,b_j,w_jk,b_k,w_kl,b_l,hidden_1_a,hidden_2_a,output_a,target,lr,error_function_option)		
Parameter	Type	Explanation
x	Array	Input value
w_ij	Array	weight
b_j	float	bias
w_jk	Array	weight
b_k	float	Bias
w_kl	Array	weight
b_l	float	Bias
hidden_1_a	array	첫번째 Hidden layer의 a값들을 배열로 저장
hidden_2_a	array	두번째 Hidden layer의 a값들을 배열로 저장
output_a	Array	output layer의 a값들을 배열로 저장
target	array	Target value
lr	Float	Learning rate
error_function_option	int	Error 함수를 선택하는 인덱스

4. Conclusion

4.1 Result

Problem 1)

```

Problem 1
w1 = 0.7 , w2 = 0.5 , y = 0.049999999999999982
w1 = 0.8425 , w2 = 0.309999999999999994 , y = 0.64375
w1 = 0.8959375 , w2 = 0.238749999999999996 , y = 0.86640625000000002
w1 = 0.9159765625 , w2 = 0.21203125 , y = 0.94990234375
w1 = 0.9234912109375001 , w2 = 0.20201171875 , y = 0.98121337890625
w1 = 0.9263092041015626 , w2 = 0.19825439453125002 , y = 0.9929550170898438
w1 = 0.927365951538086 , w2 = 0.19684539794921876 , y = 0.9973581314086915
w1 = 0.9277622318267823 , w2 = 0.19631702423095707 , y = 0.9990092992782593
w1 = 0.9279108369350434 , w2 = 0.19611888408660894 , y = 0.9996284872293473
w1 = 0.9279665638506414 , w2 = 0.1960445815324784 , y = 0.9998606827110053

```

Epoch = 10으로 학습시킨 결과, 최종 결과 값이 1에 매우 근사하게 되었다. 학습이 잘 이루어졌음을 확인하였다. 그때의 weight는 대략 $w1 = 0.928$, $w2 = 0.196$ 였다. 직접 계산하여 weight를 세 번 정도 업데이트를 해주었는데, 같은 결과를 얻었다.

Problem 2)

Problem 2	
[0.539286074115832, 0.46071392588416815]	[0.07509424111351072, 0.9249057588864893]
[0.48892343151937595, 0.5110765684806241]	[0.07447209991745046, 0.9255279000825495]
[0.44318165293630507, 0.556818347063695]	[0.07386393771008075, 0.9261360622899193]
[0.40295454897403815, 0.5970454510259618]	[0.07326925326158787, 0.9267307467384122]
[0.3682471998560229, 0.6317528001439771]	[0.07268756974884787, 0.9273124302511522]
[0.33855969775411043, 0.6614403022458897]	[0.07211843326495707, 0.9278815667350431]
[0.3132018963921317, 0.6867981036078684]	[0.07156141143760839, 0.9284385885623917]
[0.29147655231390285, 0.7085234476860971]	[0.07101609214710353, 0.9289839078528964]
[0.2727606375429772, 0.7272393624570229]	[0.07048208233567597, 0.9295179176643241]
[0.2565289881951735, 0.7434710118048264]	[0.06995900690058517, 0.9300409930994148]
[0.24235134103831138, 0.7576486589616886]	[0.06944650766414998, 0.93055349233585]
[0.2298796322402499, 0.7701203677597501]	[0.06894424241451971, 0.9310557575854802]
[0.2188333985447997, 0.7811666014552002]	[0.06845188401154792, 0.9315481159884521]
[0.20898640612447875, 0.7910135938755212]	[0.06796911955264334, 0.9320308804473567]
[0.20015544759393378, 0.7998445524060662]	[0.06749564959392958, 0.9325043504060704]
[0.19219133920361653, 0.8078086607963835]	[0.06703118742245828, 0.9329688125775417]
[0.18497183029770764, 0.8150281697022924]	[0.06657545837558995, 0.9334245416244101]
[0.17839606797128607, 0.821603932028714]	[0.06612819920399476, 0.9338718007960053]
[0.1723802864583572, 0.8276197135416429]	[0.06568915747502639, 0.9343108425249735]
[0.16685444599590776, 0.8331455540040923]	[0.06525809101349735, 0.9347419089865027]
[0.1617596028299622, 0.8382403971700378]	[0.06483476737713213, 0.9351652326228678]
[0.15704584136094812, 0.8429541586390519]	[0.06441896336420024, 0.9355810366357997]
[0.1526706391649514, 0.8473293608350486]	[0.06401046455103454, 0.9359895354489655]
[0.14859756650776698, 0.8514024334922331]	[0.06360906485732755, 0.9363909351426724]
[0.1447952455265936, 0.8552047544734064]	[0.06321456613726566, 0.9367854338627343]
[0.14123651206586668, 0.8587634879341333]	[0.06282677779471635, 0.9371732222052838]
	[0.0624455164208229, 0.937554483579177]
	[0.06207060545248963, 0.9379293945475103]
	[0.06170187485035645, 0.9382981251496436]

문제 2의 경우, epoch=100으로 학습시킨 결과, 목표값이었던 [0.0, 1.0]에 근접하였다. Epoch를 늘리면 더 목표치에 가까워진다. 직접 계산하여 w를 3번 업데이트 해주었는데 프로그래밍 결과와 동일한 결과를 얻었다.

Problem 3)

```

Problem 3
[0.3388443141533153, 0.1557537129244292]
[5.9905905301245665, 29.27562193302409]
[7.59811806976116, 37.85512419306009]
[7.91962360670005, 39.571024799905985]
[7.983924719889602, 39.91420495224013]
[7.996784943687841, 39.982840988899824]
[7.999356988679551, 39.99656819747032]
[7.9998713977243066, 39.99931363943214]
[7.999974279542541, 39.999862727874046]
[7.999994855908045, 39.999972545572334]
[7.999998971181515, 39.99999450911397]
[7.99999794236285, 39.99999890182269]
[7.999999958847253, 39.99999978036452]
[7.99999999176945, 39.9999999560729]
[7.99999998353889, 39.9999999121458]
[7.99999999670778, 39.9999999824291]
[7.99999999934155, 39.9999999964858]
[7.9999999998683, 39.9999999992971]
[7.99999999997367, 39.99999999985945]
[7.99999999999472, 39.99999999997186]
[7.99999999999895, 39.99999999999446]
[7.99999999999998, 39.99999999999886]
[7.99999999999995, 39.99999999999997]
[7.99999999999998, 40.0]
[8.0, 40.0]
[8.0, 40.0]

```

Epoch = 30으로 학습시켜주었는데, 매우 빨리 목표치였던 [8.0, 40.0]에 도달함을 확인하였다. 직접 계산하여 w 를 3번 업데이트 해주었는데 프로그래밍 결과와 동일한 결과를 얻었다.

Problem 4)

```

Problem 4
[0.539286074115832, 0.46071392588416815]
[0.48892343151937595, 0.5110765684806241]
[0.44318165293630507, 0.556818347063695]
[0.40295454897403815, 0.5970454510259618]
[0.3682471998560229, 0.6317528001439771]
[0.33855969775411043, 0.6614403022458897]
[0.3132018963921317, 0.6867981036078684]
[0.29147655231390285, 0.7085234476860971]
[0.2727606375429772, 0.7272393624570229]
[0.2565289881951735, 0.7434710118048264]
[0.24235134103831138, 0.7576486589616886]
[0.2298796322402499, 0.7701203677597501]
[0.2188333985447997, 0.7811666014552002]
[0.20898640612447875, 0.7910135938755212]
[0.20015544759393378, 0.7998445524060662]
[0.19219133920361653, 0.8078086607963835]
[0.18497183029770764, 0.8150281697022924]
[0.17839606797128607, 0.821603932028714]
[0.1723802864583572, 0.8276197135416429]
[0.16685444599590776, 0.8331455540040923]
[0.1617596028299622, 0.8382403971700378]
[0.15704584136094812, 0.8429541586390519]
[0.1526706391649514, 0.8473293608350486]
[0.14859756650776698, 0.8514024334922331]
[0.1447952455265936, 0.8552047544734064]
[0.1412365120658668, 0.8587634879341333]
[0.1378977365747595, 0.8621022634252405]
[0.13475827058634032, 0.8652417294136596]
[0.1317999929339125, 0.8682000070660086]
[0.12900693564620258, 0.8709930643537974]
[0.1263649738636235, 0.8736350261363766]
[0.12386156748916041, 0.8761384325108396]
[0.12148554487011727, 0.8785144551298827]
[0.07773356849952871, 0.9222664315004713]
[0.07704995175711159, 0.9229500482428884]
[0.0763825972118353, 0.9236174027881646]
[0.0757308885442776, 0.9242691114557223]
[0.07509424111351072, 0.9249057588864893]
[0.07447209991745046, 0.9255279000825495]
[0.07386393771008075, 0.9261360622899193]
[0.07326925326158787, 0.9267307467384122]
[0.07268756974884787, 0.9273124302511522]
[0.07211843326495707, 0.927881566735043]
[0.07156141143760839, 0.9284385885623917]
[0.07101609214710353, 0.9289839078528964]
[0.07048208233567597, 0.9295179176643241]
[0.06995900690058517, 0.9300409930994148]
[0.06944650766414998, 0.93055349233585]
[0.06894424241451971, 0.9310557575854802]
[0.06845188401154792, 0.9315481159884521]
[0.06796911955264334, 0.9320308804473567]
[0.06749564959392958, 0.9325043504060704]
[0.06703118742245828, 0.9329688125775417]
[0.06657545837558995, 0.9334245416244101]
[0.06612819920399476, 0.9338718007960053]
[0.06568915747502639, 0.9343108425249735]
[0.06525809101349735, 0.9347419089865027]
[0.06483476737713213, 0.9351652326228678]
[0.06441896336420024, 0.9355810366357997]
[0.06401046455103454, 0.9359895354489655]
[0.06360906485732755, 0.9363909351426724]
[0.06321456613726566, 0.9367854338627343]
[0.06282677779471635, 0.9371732222052838]
[0.0624455164208229, 0.937554483579177]
[0.06207060545248963, 0.9379293945475103]
[0.06170187485035645, 0.9382981251496436]

```

문제 4의 경우, epoch=100으로 학습시킨 결과, 목표값이었던 [0.0, 1.0]에 근접하였다. Epoch를 늘리면 더 목표치에 가까워진다. 직접 계산하여 w를 3번 업데이트 해주었는데 프로그래밍 결과와 동일한 결과를 얻었다.

Problem 5)

```

Problem 5
[0.4766391688241321, 0.5233608311758678]
[0.44200059884555654, 0.5579994011544436]
[0.4107179413048294, 0.5892820586951706]
[0.3827863617167134, 0.6172136382832867]
[0.358003946271137, 0.6419960537288629]
[0.33606989476869564, 0.6639301052313044]
[0.31665203475016146, 0.6833479652498384]
[0.29942634120573247, 0.7005736587942676]
[0.2840965322916646, 0.7159034677083354]
[0.27040142088736524, 0.7295985791126349]
[0.25811555485917903, 0.741884445140821]
[0.2470466075029746, 0.7529533924970254]
[0.2370314895105762, 0.7629685104894238]
[0.2279322174983156, 0.7720677825016844]
[0.21963203211134574, 0.7803679678886543]
[0.21203196377194095, 0.7879680362280591]
[0.20504789313385, 0.79495210686615]
[0.19860808202510982, 0.8013919179748902]
[0.19265112165901732, 0.8073488783409827]
[0.18712423723756383, 0.8128757627624361]
[0.18198189045152885, 0.8180181095484711]
[0.17718462788227388, 0.822815372117726]
[0.172698130966803, 0.827301869033197]
[0.16849243060757618, 0.8315075693924239]
[0.16454125611600326, 0.8354587438839968]
[0.1608214938056538, 0.8391785061943462]
[0.157312735221806, 0.842687264778194]
[0.15399689881156145, 0.8460031011884386]
[0.15085791193077375, 0.8491420880692263]
[0.1478814425754331, 0.852118557424567]
[0.06587533408096932, 0.9341246659190308]
[0.06555851756782158, 0.9344414824321784]
[0.06524592784152068, 0.9347540721584794]
[0.06493747384039222, 0.9350625261596077]
[0.06463306719055838, 0.9353669328094417]
[0.06433262210569377, 0.9356673778943062]
[0.06403605529128291, 0.9359639447087171]
[0.06374328585314305, 0.936256714146857]
[0.06345423520999112, 0.936545764790009]
[0.06316882700984643, 0.9368311729901535]
[0.06288698705007262, 0.9371130129499273]
[0.06260864320087398, 0.937391356799126]
[0.06233372533207273, 0.9376662746679273]
[0.062062165243002156, 0.9379378347569978]
[0.061793896595361684, 0.9382061034046383]
[0.06152885484888761, 0.9384711451511124]
[0.06126697719970163, 0.9387330228002984]
[0.061008202521207276, 0.9389917974787926]
[0.06075247130741121, 0.9392475286925888]
[0.06049972561855341, 0.9395002743814466]
[0.06024990902893631, 0.9397500909710637]
[0.06000296657684929, 0.9399970834231507]
[0.05975884471648997, 0.94024115528351]
[0.05951749127178964, 0.9404825087282103]
[0.059278855392054644, 0.9407211446079453]
[0.05904288750934033, 0.9409571124906597]
[0.05880953929747844, 0.9411904607025214]
[0.058578763632683335, 0.9414212363673167]
[0.05835051455566569, 0.9416494854443342]
[0.05812474723518641, 0.9418752527648137]
[0.0579014179329868, 0.9420985820670131]
[0.05768048397003443, 0.9423195160299656]

```

문제 5의 경우, epoch = 150으로 학습을 시켜보았다. 그 결과 목표치였던 [0.0, 1.0]에 도달함을 확인하였다. Epoch를 늘리면 목표치에 더욱 가까워졌다. 직접 계산하여 w를 3번 업데이트 해주었는데 프로그래밍 결과와 동일한 결과를 얻었다.

Problem 6)

```

Problem 6
[0.539286074115832, 0.46071392588416815]
[0.4381974616568335, 0.5618025383431665]
[0.35921153641375925, 0.6407884635862408]
[0.299139900721084, 0.700860099278916]
[0.2534036678174629, 0.7465963321825372]
[0.21811879111351343, 0.7818812088864865]
[0.19042301697180655, 0.8095769830281934]
[0.1682957233169182, 0.8317042766830818]
[0.15031970416558013, 0.8496802958344198]
[0.13549258077689014, 0.8645074192231099]
[0.12309508271203955, 0.8769049172879605]
[0.1126025262151591, 0.8873974737848409]
[0.10362574235710047, 0.8963742576428996]
[0.0958714100561286, 0.9041285899438715]
[0.08911509454534379, 0.9108849054546562]
[0.08318265294232434, 0.9168173470576756]
[0.07793721256647718, 0.9220627874335229]
[0.07326991220774072, 0.9267300877922594]
[0.06909322166913755, 0.9309067783308625]
[0.06533605394931767, 0.9346639460506824]
[0.061940141748264364, 0.9380598582517355]
[0.05885731797188198, 0.941142682028118]
[0.05604745105898018, 0.9439525489410199]
[0.05347686049688405, 0.9465231395031161]
[0.051117088559826065, 0.948882911440174]
[0.04894393919566008, 0.95105606080434]
[0.04693671931647631, 0.9530632806835236]
[0.04507763491847332, 0.9549223650815267]
[0.04335130671200872, 0.9566486932879914]
[0.041744378785741866, 0.9582556212142581]

```

문제 6은 epoch = 30일 때의 결과를 출력한 것이다. 학습속도가 매우 빨라서 30회의 학습만으로 목표치였던 [0.0, 1.0]에 매우 근접하였다. error함수로 SSE가 아닌 logistic error를 사용하여 이런 결과를 얻을 수 있었다. 직접 계산하여 w를 3번 업데이트 해주었는데 프로그래밍 결과와 동일한 결과를 얻었다.

Problem 7)

```

Problem 7
[0.4766391688241321, 0.5233608311758678]
[0.40777144684436767, 0.5922285531556324]
[0.3514598108951409, 0.648540189104859]
[0.3058797073060977, 0.6941202926939023]
[0.2689414775179093, 0.7310585224820906]
[0.23879426470872547, 0.7612057352912744]
[0.2139494587463353, 0.7860505412536647]
[0.19325648807885726, 0.8067435119211428]
[0.17583868384214388, 0.8241613161578561]
[0.16102923475747014, 0.8389707652425298]
[0.14831858610959625, 0.8516814138904039]
[0.1373142809215253, 0.8626857190784746]
[0.12771120989692694, 0.872288790103073]
[0.11926980981367519, 0.8807301901863248]
[0.11180010481003996, 0.88819989518996]
[0.10514997762847637, 0.8948500223715237]
[0.09919649099398925, 0.9008035090060107]
[0.0938394129329208, 0.9061605870670792]
[0.0889963432115889, 0.9110036567884111]
[0.08459901138884415, 0.9154009886111558]
[0.08059043923338723, 0.9194095607666128]
[0.07692274634702487, 0.9230772536529752]
[0.07355543861774054, 0.9264445613822595]
[0.07045406226353408, 0.929545937736466]
[0.06758913704593295, 0.932410862954067]
[0.0649353044137971, 0.9350646955862029]
[0.062470642429882275, 0.9375293575701178]
[0.06017611110067513, 0.939823888993249]
[0.05803510040539622, 0.9419648995946038]
[0.05603305976635105, 0.943966940233649]
[0.05415719252993922, 0.9458428074700609]
[0.052396202669560264, 0.9476037973304398]
[0.05074008368942417, 0.9492599163105758]
[0.04917994182650405, 0.9508200581734959]
[0.04770784727996843, 0.9522921527200315]
[0.04631670846315253, 0.9536632915368475]
[0.04500016526085126, 0.9549998347391488]
[0.04375249805012557, 0.9562475019498744]
[0.042568549855051215, 0.9574314501449488]
[0.04144365949191104, 0.958556340508089]
[0.040373603949285096, 0.9596263960507115]
[0.03935454855870814, 0.9606454514412919]
[0.03838300376244167, 0.9616169962375584]
[0.03745578748809293, 0.962544212511907]
[0.03656999230511651, 0.9634300076948834]
[0.03572295667329972, 0.9642770433267003]
[0.03491223970415246, 0.9650877602958475]
[0.03413559894741359, 0.9658644010525864]
[0.033390970790379, 0.9666090292096209]
[0.03267645312041963, 0.9673235468795803]

```

문제 7은 $\text{epoch} = 50$ 일 때의 결과를 출력한 것이다. 50회의 학습만으로 목표치였던 $[0.0, 1.0]$ 에 근접하였다. 문제 6과 마찬가지로, error함수로 SSE가 아닌 logistic error를 사용하여 더 빠른 학습 결과를 얻을 수 있었다. 직접 계산하여 w 를 3번 업데이트 해주었는데 프로그래밍 결과와 동일한 결과를 얻었다.

4.2 Conclusion

- Weight와 bias를 프로그래밍을 통하여 직접 업데이트를 해보고 학습 결과를 확인해 볼 수 있었다.
- 다양한 활성화 함수와 error 함수를 비교해가면서, 여러 학습결과를 얻을 수 있었다.
- 문제 4번, 5번과 6번,7번을 비교한 결과 error 함수로 logistic을 사용했을 때 더 빠른 학습 결과를 얻을 수 있었다.
- PA#2을 수행함으로써 인공지능망의 기초 내용을 직접 구현하고, 결과를 확인할 수 있었다.