



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

Fall 2020

머신러닝

PA # 2

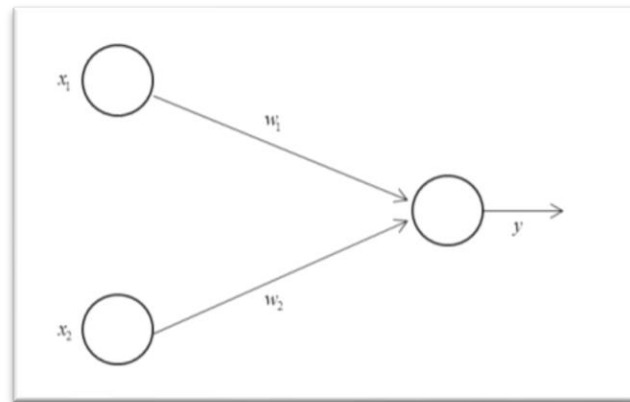
| | |
|-----------------|------------|
| Instructor name | 김태완 교수님 |
| Student name | 이용준 |
| Department | 조선해양공학과 |
| Student ID | 2015-19595 |
| Submission date | 2020.10.11 |

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 |

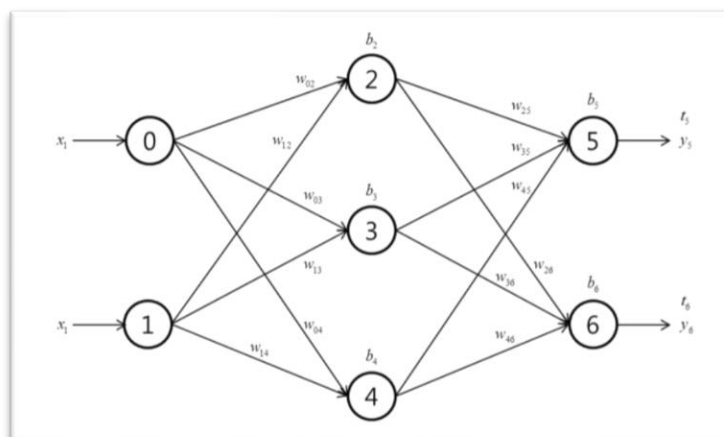
1. Problem Definition

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

- SSE으로 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

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

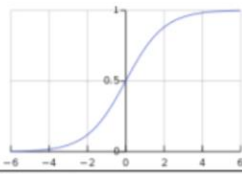
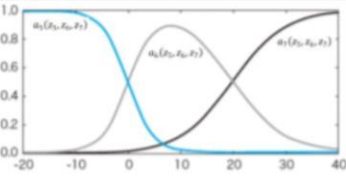
2. Problem Analysis and Design

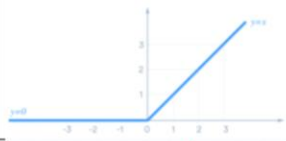

2.1 Analysis

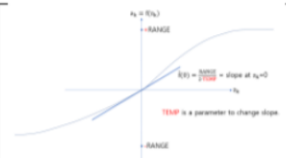
- Error Function

| Error of one node | |
|------------------------------------|---|
| Cost | Equation |
| Sum of squared error (SSE) | $C_5 = \frac{1}{2}(y_5 - a_5)^2$ |
| Cross entropy cost function (CECF) | $C_5 = -[y_5 \ln a_5 + (1 - y_5) \ln(1 - a_5)]$ |
| Logistic error | $C_5 = -y_5 \ln a_5$ |

- 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:

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

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

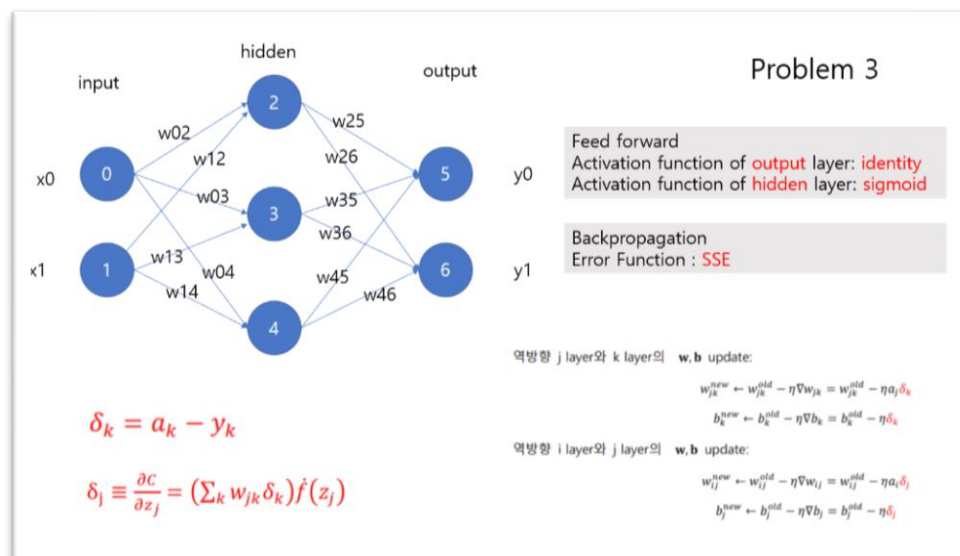
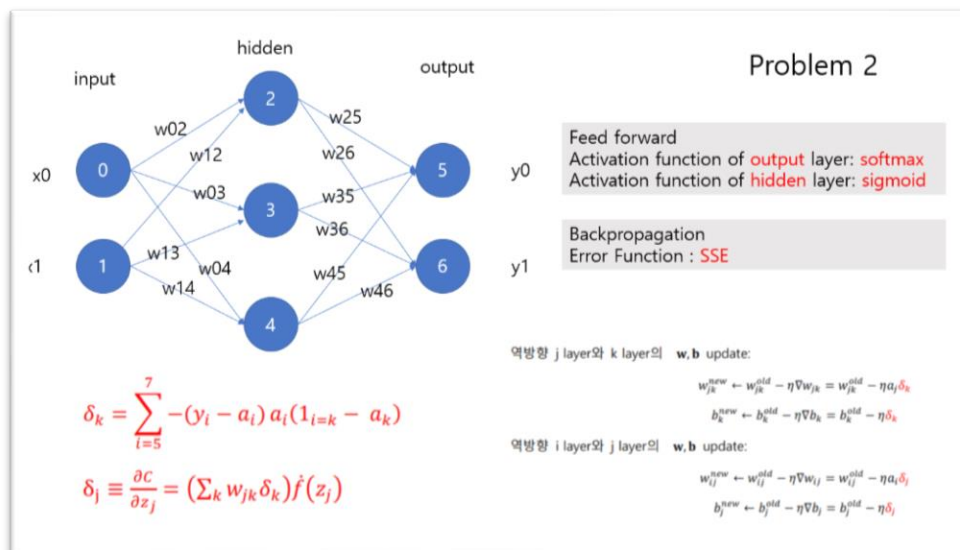
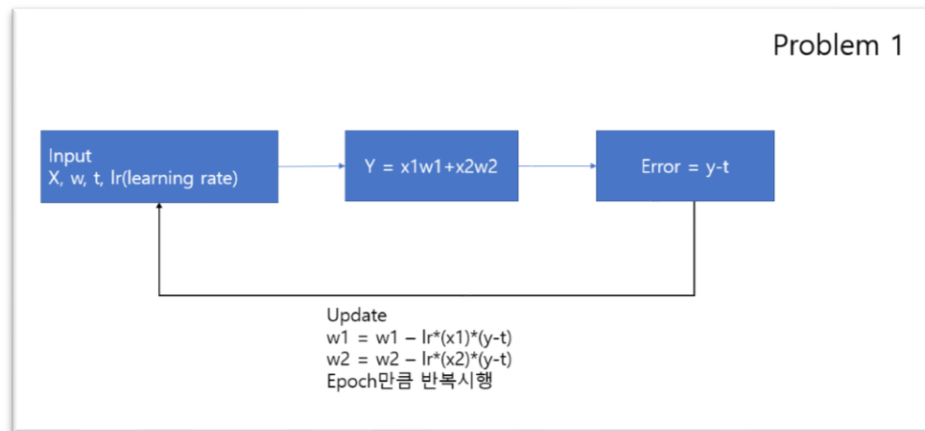
역방향 i layer와 j layer의 \mathbf{w}, \mathbf{b} update:

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

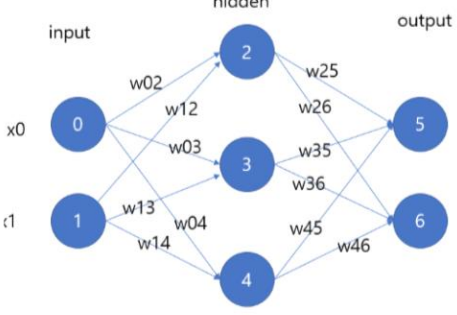
$$b_j^{new} \leftarrow b_j^{old} - \eta \nabla b_j = b_j^{old} - \eta \delta_j$$

2.2 Data Flow Diagram

각 문제에 대한 DFD이다.



Problem 4



input: x_0, x_1 hidden: 2, 3, 4 output: y_0, y_1

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:

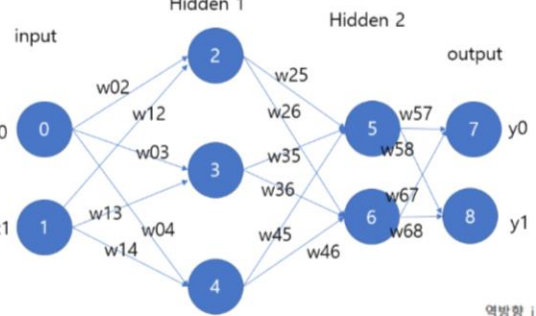
$$w_{ij}^{new} \leftarrow w_{ij}^{old} - \eta \nabla w_{ij} = w_{ij}^{old} - \eta a_i \delta_j$$

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

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

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

Problem 5



input: x_0, x_1 Hidden 1: 2, 3, 4 Hidden 2: 5, 6 output: y_0, y_1

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:

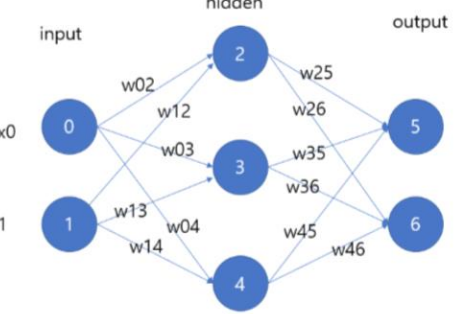
$$w_{ij}^{new} \leftarrow w_{ij}^{old} - \eta \nabla w_{ij} = w_{ij}^{old} - \eta a_i \delta_j$$

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

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



input: x_0, x_1 hidden: 2, 3, 4 output: y_0, y_1

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:

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

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

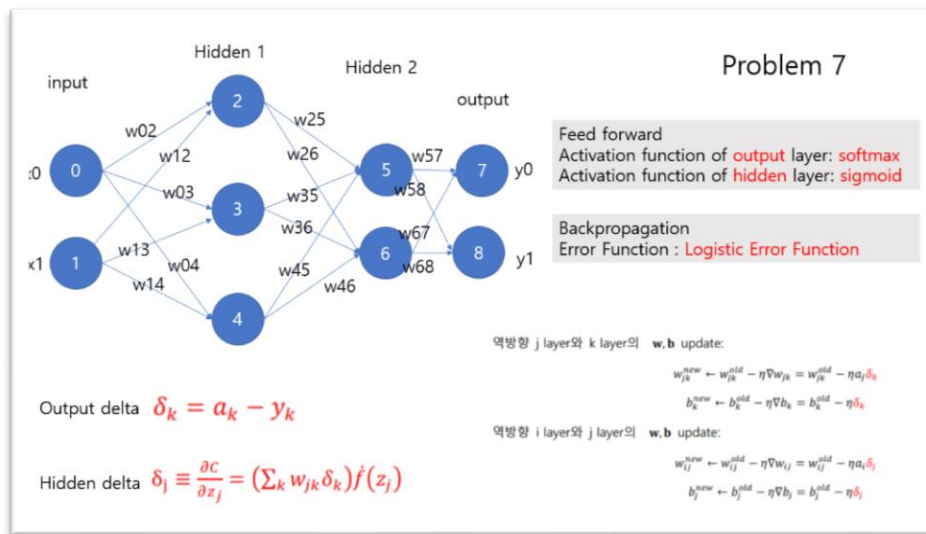
역방향 i layer와 j layer의 **w, b** update:

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

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

$$\delta_k = a_k - y_k$$

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



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 = 0.9 , w2 = 0.3 , y = 0.7500000000000001
w1 = 0.9375 , w2 = 0.25 , y = 0.90625
w1 = 0.9515625 , w2 = 0.23125 , y = 0.9648437499999999
w1 = 0.9568359375 , w2 = 0.22421875 , y = 0.9868164062499998
w1 = 0.9588134765625 , w2 = 0.22158203124999995 , y = 0.9950561523437502
w1 = 0.9595550537109375 , w2 = 0.22059326171875 , y = 0.9981460571289064
w1 = 0.9598331451416016 , w2 = 0.2202247314453125 , y = 0.9993047714233398
w1 = 0.9599374294281006 , w2 = 0.22008342742919923 , y = 0.9997392892837523
w1 = 0.9599765360355377 , w2 = 0.2200312852859497 , y = 0.9999022334814069
w1 = 0.9599912010133267 , w2 = 0.2200117319822311 , y = 0.9999633375555279

```

Epoch = 10으로 학습시킨 결과, 최종 결과 값이 1에 매우 근사하게 되었다. 학습이 잘 이루어졌음을 확인하였다. 그때의 weight는 대략 $w1 = 0.96$, $w2 = 0.22$ 였다. 직접 계산하여 weight를 세 번 정도 업데이트를 해주었는데, 같은 결과를 얻었다.

Problem 2)

| | |
|--|---|
| <pre> Problem 2 [0.539286074115832, 0.46071392588416815] [0.5141369095193018, 0.4858630904806982] [0.48997015667395205, 0.5100298433260481] [0.46696912191626355, 0.5330308780837366] [0.4452505757689585, 0.5547494242310416] [0.4248720313731654, 0.5751279686268346] [0.40584250210811035, 0.5941574978918897] [0.3881342493928367, 0.6118657506071633] [0.3716937978123171, 0.628306202187683] [0.3564512861760681, 0.6435487138239319] [0.3423278388116104, 0.6576721611883897] [0.3292410304318099, 0.6707589695681901] [0.3171087138119923, 0.6828912861880077] [0.30585154329874326, 0.6941484567012567] [0.2953945154360422, 0.7046054845639577] [0.2856678015044761, 0.7143321984955239] [0.276607090360548, 0.7233929096394519] [0.26815360654895004, 0.73184639345105] [0.26025392361288235, 0.7397460763871176] [0.25285965704742286, 0.7471403429525771] [0.2459270946442907, 0.7540729053557094] [0.23941680254549422, 0.7605831974545059] [0.23329323154593415, 0.7667067684540658] [0.22752433863251037, 0.7724756613674897] [0.22208123226265536, 0.7779187677373447] [0.21693784558007184, 0.7830621544199282] </pre> | <pre> [0.11517652149720797, 0.884823478502792] [0.11419021647738882, 0.8858097835226112] [0.1132260646424308, 0.8867739353575692] [0.11228328592523742, 0.8877167140747625] [0.1113611370835183, 0.8886388629164816] [0.1104589095427838, 0.8895410904572162] [0.10957592738899022, 0.8904240726110099] [0.1087115454989149, 0.8912884545010851] [0.10786514779740713, 0.8921348522025929] [0.10703614563162146, 0.8929638543683786] [0.10622397625320631, 0.8937760237467937] [0.10542810140020399, 0.894571898599796] [0.10464800597112638, 0.8953519940288737] [0.1038831967843108, 0.8961168032156892] [0.10313320141624123, 0.8968667985837587] [0.10239756711304648, 0.8976024328869535] [0.10167585976986468, 0.8983241402301353] [0.10096766297319772, 0.8990323370268023] [0.10027257710177358, 0.8997274228982264] [0.0995902184817957, 0.9004097815182043] [0.0989202185927826, 0.9010797814072173] [0.09826222332050381, 0.9017377766794962] [0.09761589225378717, 0.9023841077462128] [0.09698089802222383, 0.9030191019777762] [0.09635692567202382, 0.9036430743279762] [0.0957436720774838, 0.9042563279225162] [0.09514084538571899, 0.904859154614281] [0.09454816449248672, 0.9054518355075133] [0.09396535854708968, 0.9060346414529103] </pre> |
|--|---|

문제 2의 경우, 목표값이었던 $[0.0, 1.0]$ 에 근접하려면 적어도 epoch = 100의 학습이 필요했다. Epoch를 늘리면 더 목표치에 가까워진다. 결과적으로 목표값에 매우 근접함을 확인하였다. 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]
[7.9999999997367, 39.99999999985945]
[7.9999999999472, 39.9999999997186]
[7.9999999999895, 39.9999999999446]
[7.9999999999998, 39.9999999999886]
[7.9999999999995, 39.9999999999997]
[7.9999999999998, 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]
[0.5141369095193018, 0.4858630904806982]
[0.48997015667395205, 0.5100298433260481]
[0.46696912191626355, 0.5330308780837366]
[0.4452505757689585, 0.5547494242310416]
[0.4248720313731654, 0.5751279686268346]
[0.40584250210811035, 0.5941574978918897]
[0.3881342493928367, 0.6118657506071633]
[0.3716937978123171, 0.628306202187683]
[0.3564512861760681, 0.6435487138239319]
[0.3423278388116104, 0.6576721611883897]
[0.3292410304318099, 0.6707589695681901]
[0.3171087138119923, 0.6828912861880077]
[0.30585154329874326, 0.6941484567012567]
[0.2953945154360422, 0.7046054845639577]
[0.2856678015044761, 0.7143321984955239]
[0.276607090360548, 0.7233929096394519]
[0.26815360654895004, 0.73184639345105]
[0.26025392361288235, 0.7397460763871176]
[0.25285965704742286, 0.7471403429525771]
[0.2459270946442907, 0.7540729053557094]
[0.23941680254549422, 0.7605831974545059]
[0.23329323154593415, 0.7667067684540658]
[0.22752433863251037, 0.7724756613674897]
[0.22208123226265536, 0.7779187677373447]
[0.21693784558007184, 0.7830621544199282]
[0.21207063898111253, 0.7879293610188874]
[0.11419021647738882, 0.8858097835226112]
[0.1132260646424308, 0.8867739353575692]
[0.11228328592523742, 0.8877167140747625]
[0.1113611370835183, 0.8886388629164816]
[0.1104589095427838, 0.8895410904572162]
[0.10957592738899022, 0.8904240726110099]
[0.1087115454989149, 0.8912884545010851]
[0.10786514779740713, 0.8921348522025929]
[0.10703614563162146, 0.8929638543683786]
[0.10622397625320631, 0.8937760237467937]
[0.10542810140020399, 0.894571898599796]
[0.10464800597112638, 0.8953519940288737]
[0.1038831967843108, 0.8961168032156892]
[0.10313320141624123, 0.8968667985837587]
[0.10239756711304648, 0.8976024328869535]
[0.10167585976986468, 0.8983241402301353]
[0.10096766297319772, 0.8990323370268023]
[0.10027257710177358, 0.8997274228982264]
[0.0995902184817957, 0.9004097815182043]
[0.0989202185927826, 0.9010797814072173]
[0.09826222332050381, 0.901737766794962]
[0.09761589225378717, 0.9023841077462128]
[0.09698089802222383, 0.9030191019777762]
[0.096356932567202382, 0.9036430743279762]
[0.0957436720774838, 0.9042563279225162]
[0.09514084538571899, 0.904859154614281]
[0.09454816449248672, 0.9054518355075133]
[0.09396535854708968, 0.9060346414529103]

```

문제 4의 경우, epoch = 100의 학습이 있어야 목표값이었던 [0.0, 1.0]에 근접했다. Epoch를 늘리면 더

목표치에 가까워진다. 결과적으로 목표값에 매우 근접함을 확인하였다. 직접 계산하여 w 를 3번 업데이트 해주었는데 프로그래밍 결과와 동일한 결과를 얻었다.

Problem 5)

```

Problem 5
[0.4766391688241321, 0.5233608311758678] [0.09644841015761473, 0.9035515898423853]
[0.4592704848500297, 0.5407295151499704] [0.09599434873498335, 0.9040056512650168]
[0.4427011020599115, 0.5572988979400886] [0.09554608578501293, 0.9044539142149871]
[0.42695010147612705, 0.5730498985238731] [0.09510350315975985, 0.9048964968402402]
[0.4120194277173524, 0.5879805722826477] [0.09466648598197758, 0.9053335140180224]
[0.3978973604652305, 0.6021026395347695] [0.09423492253146483, 0.9057650774685352]
[0.38456179273209234, 0.6154382072679077] [0.09380870413614156, 0.9061912958638585]
[0.37198314206946637, 0.6280168579305337] [0.09338772506762466, 0.9066122749323753]
[0.3601268142613614, 0.6398731857386386] [0.09297188244108766, 0.9070281175589123]
[0.3489552060862537, 0.6510447939137463] [0.09256107611920102, 0.9074389238807989]
[0.33842927612154783, 0.6615707238784522] [0.09215520861995938, 0.9078447913800407]
[0.32850973527056937, 0.6714902647294305] [0.09175418502821363, 0.9082458149717865]
[0.3191579174137956, 0.6808420825862045] [0.0913579129107349, 0.9086420870892651]
[0.3103363903871992, 0.6896636096128009] [0.09096630223464652, 0.9090336977653535]
[0.30200936227648295, 0.6979906377235171] [0.09057926528906933, 0.9094207347109307]
[0.2941429305021196, 0.7058570694978805] [0.09019671660983294, 0.909803283390167]
[0.28670521307761304, 0.713294786922387] [0.08981857290711362, 0.9101814270928864]
[0.2796663937364194, 0.7203336062635806] [0.08944475299586721, 0.9105552470041328]
[0.27299870582598557, 0.7270012941740145] [0.08907517772893073, 0.9109248222710693]
[0.2666763741320159, 0.7333236258679842] [0.08870976993267464, 0.9112902300673253]
[0.26067552911435005, 0.7393244708856499] [0.08834845434509211, 0.9116515456549079]
[0.2549741043081568, 0.7450258956918432] [0.0879911575562186, 0.9120088424437814]
[0.24955172473382778, 0.7504482752661723] [0.08763780795077898, 0.912362192049221]
[0.24438959192334628, 0.7556104080766538] [0.08728833565296634, 0.9127116643470337]
[0.23947036947763892, 0.760529630522361] [0.08694267247325978, 0.9130573275267402]
[0.23477807180344065, 0.7652219281965594] [0.08660075185719389, 0.913399248142806]
[0.23029795774361936, 0.7697020422563806] [0.08626250883599626, 0.9137374911640037]

```

문제 5의 경우 학습 속도가 느려서, $\text{epoch} = 150$ 으로 학습을 시켰을 때 목표치였던 $[0.0, 1.0]$ 에 도달함을 확인하였다. Epoch를 늘리면 목표치에 더욱 가까워졌다. 직접 계산하여 w 를 3번 업데이트 해주었는데 프로그래밍 결과와 동일한 결과를 얻었다.

Problem 6)

```

Problem 6
[0.539286074115832, 0.46071392588416815] [0.4886103185463793, 0.5113896814536207]
[0.44287917177835934, 0.5571208282216407] [0.4021675278520071, 0.5978324721479928]
[0.36621948541534827, 0.6337805145846518] [0.33460847979555597, 0.6653915202044439]
[0.3068456438371835, 0.6931543561628166] [0.2824440421458899, 0.715559578541101]
[0.26095193798693594, 0.739048062013064] [0.24196678008819428, 0.7580332199118057]
[0.22513826486148036, 0.7748617351385196] [0.21016579104869068, 0.7898342089513093]
[0.1967934336499922, 0.8032065663500078] [0.1848041566877932, 0.8151958433122067]
[0.17401414195709627, 0.8259858580429037] [0.16426763263956845, 0.8357323673604315]
[0.15543243220999994, 0.8445675677900001] [0.14739606749436748, 0.8526039325056325]
[0.14006256374167797, 0.859937436258322] [0.13334975636195776, 0.8666502436380423]
[0.12718706003215047, 0.8728129399678496] [0.12151362099043052, 0.8784863790095695]
[0.11627678709279694, 0.883723212907203] [0.11143083981487865, 0.8885691601851213]
[0.10693594152019495, 0.8930640584798051] [0.10275725943136196, 0.8972427405686381]
[0.09886423467820935, 0.9011357653217906] [0.09522997059386064, 0.9047700294061394]
[0.09183071920636565, 0.9081692807936342] [0.08864544877508826, 0.9113545512249117]

```

문제 6은 epoch = 30일 때의 결과를 출력한 것이다. 30회의 학습만으로 목표치였던 [0.0, 1.0]에 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] [0.1517985257228039, 0.848201474277196]
[0.44192531389351497, 0.558074686106485] [0.14597079302688054, 0.8540292069731195]
[0.41022706014277976, 0.5897729398572203] [0.14053280949523472, 0.8594671905047653]
[0.3814331978855707, 0.6185668021144293] [0.13544915491930562, 0.8645508450806944]
[0.35536011194740175, 0.6446398880525983] [0.13068833401543836, 0.8693116659845617]
[0.3317875068364586, 0.6682124931635415] [0.1262227961966914, 0.8737777203803309]
[0.3104829915098865, 0.6895170084901137] [0.12202592414222707, 0.877974075857773]
[0.29121773272973145, 0.7087822672702685] [0.11807682964711054, 0.8819231703528895]
[0.27377558193730867, 0.7262244180626913] [0.11435486819433657, 0.8856451318056635]
[0.2579577441676463, 0.7420422558323537] [0.11084194524043403, 0.8891580547595659]
[0.243584570511645, 0.756415429488355] [0.1075217599213875, 0.8924782400786125]
[0.2304955978978478, 0.7695044021021523] [0.10437959694133236, 0.8956204030586676]
[0.21854859402157706, 0.781451405978423] [0.10140214556755153, 0.8985978544324484]
[0.20761809715477123, 0.7923819028452287] [0.09857734189916936, 0.9014226581008307]
[0.19759375478408173, 0.8024062452159183] [0.09589423114609771, 0.9041057688539023]
[0.18837864130123563, 0.8116213586987643] [0.09334284713888949, 0.9066571528611105]
[0.17988765517190788, 0.820112344828092] [0.09091410670099946, 0.9090858932990005]
[0.1720460460376914, 0.8279539539623086] [0.08859971686318373, 0.9114002831368163]
[0.16478809188776686, 0.8352119081122331] [0.08639209319477026, 0.9136079068052297]
[0.15805592876133684, 0.8419440712386632] [0.08428428777643772, 0.9157157122235623]
[0.1517985257228039, 0.848201474277196] [0.0822699255509361, 0.9177300744490638]
[0.14597079302688054, 0.8540292069731195] [0.08034314796781432, 0.9196568520321856]
[0.14053280949523472, 0.8594671905047653] [0.07849856299072912, 0.9215014370092709]
[0.13544915491930562, 0.8645508450806944] [0.07673120066554634, 0.9232687993344537]
[0.13068833401543836, 0.8693116659845617] [0.07503647355779498, 0.9249635264422051]
[0.1262227961966914, 0.8737777203803309] [0.07341014146210144, 0.9265898585378987]
[0.12202592414222707, 0.877974075857773] [0.07184827986654924, 0.9281517201334508]
[0.11807682964711054, 0.8819231703528895] [0.07034725172360042, 0.9296527482763995]
[0.11435486819433657, 0.8856451318056635] [0.06890368213805963, 0.9310963178619404]
[0.11084194524043403, 0.8891580547595659] [0.06751443563306284, 0.9324855643669371]
[0.1075217599213875, 0.8924782400786125] [0.06591443563306284, 0.9339747114619371]
[0.10437959694133236, 0.8956204030586676] [0.06431443563306284, 0.9354638625529321]
Press any key to continue

```

문제 7은 epoch = 50일 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 w를 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.