

Softmax Regression

Multinomial Classification

Multinomial Classification

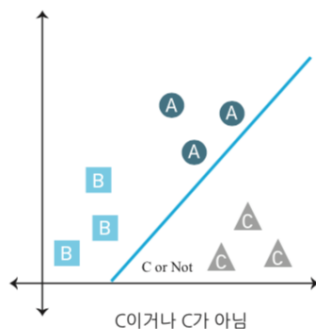
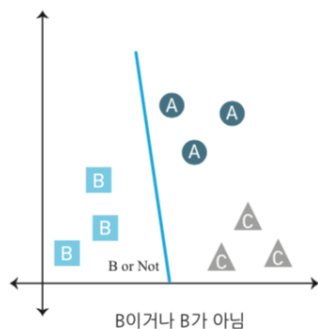
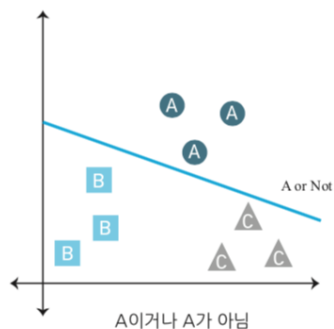
- Multinomial Classification

- Binary Classification은 2가지로 결과를 분류하는 것이지만 Multinomial Classification은 3가지 이상으로 결과를 분류
- 기본 개념

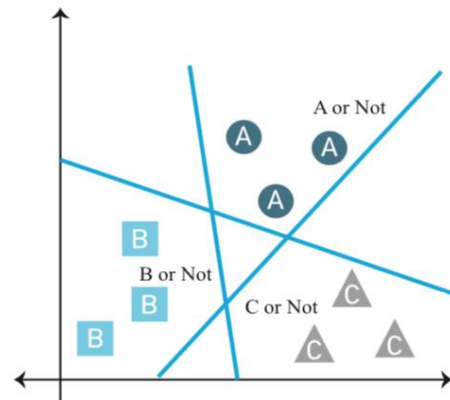
- Binary Classification을 이용한 Multinomial Classification : A, B, C 분류

- A 이거나 A가 아님
- B 이거나 B가 아님
- C 이거나 C가 아님

- A,B,C 를 분류 하는 Binary Classification을 함께 사용한다면 3가지를 분류하는 Multinomial Classification이 가능



함



Multinomial Classification Softmax Regression

Multinomial Classification Softmax Regression 실습

- 학습에 필요한 모듈 선언

```
1 #####
2 # [학습에 필요한 모듈 선언]
3 #####
4 import tensorflow as tf
5 import numpy as np
6 from numpy.random import multivariate_normal, permutation
7 import pandas as pd
8 from matplotlib import pyplot as plt
9 # seaborn import error 발생시 pip install seaborn 로 설치
10 import seaborn as sns
11
```

- tensorflow, numpy, matplotlib, pandas 라이브러리 사용
- seaborn 라이브러리는 그래프를 그리는 라이브러리로 학습 데이터를 시각화 함

Multinomial Classification Softmax Regression 실습

- 환경설정

```
12 #####
13 # [환경설정]
14 #####
15 # 학습 데이터 수 선언
16 # y = class1 인 클래스
17 Y_class1 = 200
18 # y = class2 인 클래스
19 Y_class2 = 200
20 # y = class3 인 클래스
21 Y_class3 = 200
22 # 학습 데이터(훈련/검증/테스트) 비율
23 trainDataRate = 0.7
24 validationDataRate = 0.1
25 # 학습률
26 learningRate = 0.01
27 # 총 학습 횟수
28 totalStep = 10001
29
```

- 학습 데이터는 3가지 종류의 상태를 가지도록 하며 총 600개의 데이터를 직접 생성
- 학습률과 총 학습 횟수를 지정, 학습 데이터를 훈련, 검증, 테스트 데이터로 분리할 비율 선언

Multinomial Classification Softmax Regression 실습

- 빌드단계 - Step1) 학습 데이터 준비 : 데이터 생성

- 학습데이터는 독립변수 5개, 종속변수 3개 (라벨 3개)를 직접 생성
- 분류 되는 모델을 만들기 위하여 어느정도 군집성을 가지는 데이터를 생성하여 사용
- 학습데이터는 multivariate_normal() 이용하여 다변수 정규분포에서 난수를 생성
- 첫번째 라벨의 실제 값을 'class1'로 지정

```
30 #####
31 # [빌드단계]
32 # Step 1) 학습 데이터 준비
33 #####
34 # 시드 설정 : 항상 같은 난수를 생성하기 위하여 수동으로 설정
35 np.random.seed(321)
36
37 ### (1) 학습 데이터 생성
38 # y = class1 인 학습데이터 생성
39 # 데이터 수
40 dataNumber_y1 = Y_class1
41 # 데이터가 평균
42 mu_y1 = [1, 1, 1, 1, 1]
43 # 데이터 분산된 정도
44 variance_y1 = 4
45 # 난수 생성
46 data_y1 = multivariate_normal(mu_y1, np.eye(5) * variance_y1, dataNumber_y1)
47 df_y1 = pd.DataFrame(data_y1, columns=['x1', 'x2', 'x3', 'x4', 'x5'])
48 df_y1['y'] = 'class1'
49
```

Multinomial Classification Softmax Regression 실습

- **빌드단계 - Step1) 학습 데이터 준비 : 데이터 생성**
 - 첫번째 라벨 'class1'의 데이터를 생성하는 것과 동일한 방법으로 'class2', 'class3' 라벨의 학습데이터를 생성

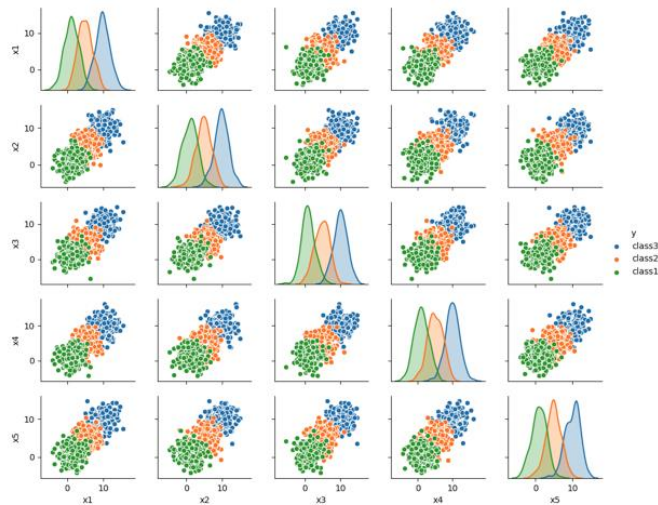
```
50 # y = class2 인 학습데이터 생성
51 # 데이터 수
52 dataNumber_y2 = Y_class2
53 # 데이터가 평균
54 mu_y2 = [5, 5, 5, 5, 5]
55 # 데이터 분산된 정도
56 variance_y2 = 4
57 # 난수 생성
58 data_y2 = multivariate_normal(mu_y2, np.eye(5) * variance_y2, dataNumber_y2)
59 df_y2 = pd.DataFrame(data_y2, columns=['x1', 'x2', 'x3', 'x4', 'x5'])
60 df_y2['y'] = 'class2'
61
62 # y = class3 인 학습데이터 생성
63 # 데이터 수
64 dataNumber_y3 = Y_class3
65 # 데이터가 평균
66 mu_y3 = [10,10,10,10,10]
67 # 데이터 분산된 정도
68 variance_y3 = 4
69 # 난수 생성
70 data_y3 = multivariate_normal(mu_y3, np.eye(5) * variance_y3, dataNumber_y3)
71 df_y3 = pd.DataFrame(data_y3, columns=['x1', 'x2', 'x3', 'x4', 'x5'])
72 df_y3['y'] = 'class3'
73
```


Multinomial Classification Softmax Regression 실습

- 빌드단계 - Step1) 학습 데이터 준비 : 데이터 생성

```
74 # 생성한 데이터를 하나의 DataFrame 으로 합치기
75 df = pd.concat([df_y1, df_y2, df_y3], ignore_index = True)
76 # 순서에 상관없이 데이터 정렬
77 df_totalTrainData = df.reindex(permutation(df.index)).reset_index(drop=True)
78
79 # 학습 데이터 확인
80 print("==== Data =====")
81 print(df_totalTrainData.head())
82 print(df_totalTrainData.tail())
83 # 학습데이터 shape 확인
84 print("df_totalTrainData Shape : {}".format(df_totalTrainData.shape))
85
86 # 학습데이터 전체 그래프 확인
87 sns.pairplot(df_totalTrainData, hue="y", height=2)
88 plt.show()
89
```

- 3가지 라벨을 가진 학습데이터를 하나의 Dataframe으로 합침
- 직접 생성한 학습데이터를 seaborn 라이브러리를 이용하여 확인
- 정규 분포를 가지는 랜덤한 데이터를 생성하였기 때문에 학습 모델의 정확도는 어느정도 높은 값을 가질 것으로 예측



```
==== Data =====
   x1      x2      x3      x4      x5      y
0  8.129395 10.992341  8.104283 10.875429  9.732628 class3
1 10.030596  8.026777 10.414516  9.301360  8.819167 class3
2 13.194423  8.518136  8.749465 13.518546  7.343280 class3
3  6.367265  6.449413  4.547888  7.859430  2.725347 class2
4 11.547493 10.979799  9.752420 13.578842 11.388635 class3
...
595 5.083993  5.046393  3.870268  3.937826 1.947004 class2
596 3.060972 -2.289712 -1.257480  3.877043  3.532501 class1
597 5.246079  2.124267  7.474473  7.799916  3.584603 class2
598 8.334964 12.220503  8.737706 10.636006  6.780932 class3
599 4.887244  3.902188  4.692803  6.634168  3.967666 class2
df_totalTrainData Shape : (600, 6)
```

Multinomial Classification Softmax Regression 실습

- 빌드단계 - Step1) 학습 데이터 준비 : 데이터 전처리

```
90  ### (2) 범주형 데이터 y컬럼 데이터 맵핑 선언
91  # y 컬럼 문자열 데이터를 리스트 형태로 변환
92  y_mapping = {
93      "class1" : [1.0, 0.0, 0.0],
94      "class2" : [0.0, 1.0, 0.0],
95      "class3" : [0.0, 0.0, 1.0]
96  }
97  df_totalTrainData['y'] = df_totalTrainData['y'].map(y_mapping)
98
99
100 print("==== after mapping =====>")
101 print(df_totalTrainData.head())
102 print(df_totalTrainData.tail())
103
```

- 데이터 전처리 - 범주형 데이터 맵핑

- 문자열 데이터로 되어 있는 라벨 데이터를 정수형 데이터로 대체
- 학습데이터의 실제값을 배열로 구성된 데이터의 라벨로 맵핑

```
==== Data =====>
      x1      x2      x3      x4      x5      y
0  8.129395  10.992341  8.104283  10.875429  9.732628  class3
1  10.030596  8.826777  10.414516  9.301360  8.819167  class3
2  13.194423  8.518136  8.749465  13.518546  7.343280  class3
3   6.367265  6.449413  4.547888  7.859430  2.725347  class2
4  11.547493  10.979799  9.752420  13.578842  11.388635  class3
      x1      x2      x3      x4      x5      y
595  5.083993  5.046393  3.870268  3.937826  1.947004  class2
596  3.060972 -2.289712 -1.257480  3.877043  3.532501  class1
597  5.246079  2.124267  7.474473  7.799916  3.584603  class2
598  8.334964  12.220503  8.737706  10.636006  6.780932  class3
599  4.887244  3.902188  4.692803  6.634168  3.967666  class2
df_totalTrainData Shape : (600, 6)
```

```
==== after mapping =====>
      x1      x2      x3      x4      x5      y
0  8.129395  10.992341  8.104283  10.875429  9.732628  [0.0, 0.0, 1.0]
1  10.030596  8.826777  10.414516  9.301360  8.819167  [0.0, 0.0, 1.0]
2  13.194423  8.518136  8.749465  13.518546  7.343280  [0.0, 0.0, 1.0]
3   6.367265  6.449413  4.547888  7.859430  2.725347  [0.0, 1.0, 0.0]
4  11.547493  10.979799  9.752420  13.578842  11.388635  [0.0, 0.0, 1.0]
      x1      x2      x3      x4      x5      y
595  5.083993  5.046393  3.870268  3.937826  1.947004  [0.0, 1.0, 0.0]
596  3.060972 -2.289712 -1.257480  3.877043  3.532501  [1.0, 0.0, 0.0]
597  5.246079  2.124267  7.474473  7.799916  3.584603  [0.0, 1.0, 0.0]
598  8.334964  12.220503  8.737706  10.636006  6.780932  [0.0, 0.0, 1.0]
599  4.887244  3.902188  4.692803  6.634168  3.967666  [0.0, 1.0, 0.0]
```

Multinomial Classification Softmax Regression 실습

- 빌드단계 - Step1) 학습 데이터 준비

- 결과데이터를 리스트로 반환
- 학습데이터는 훈련, 검증, 테스트 데이터를 7:1:2의 비율로 나눔
- 학습데이터 600개 중 훈련 데이터 420개, 검증 데이터 60개, 테스트 데이터 120개로 분리

```
104 ### (3) 훈련, 검증, 테스트 데이터 나누기
105 # 결과데이터 리스트로 변환
106 resultColumnName = ['y']
107 yLabelList = ['class1', 'class2', 'class3']
108 yList = df_totalTrainData.as_matrix(resultColumnName)
109 result_dataList = np.array([element1 for element3 in yList
110                             for element2 in element3
111                             for element1 in element2]).reshape(len(yList), 3)
112
113 # 학습데이터 리스트로 변환
114 featureColumnName = ['x1', 'x2', 'x3', 'x4', 'x5']
115 feature_dataList = df_totalTrainData.as_matrix(featureColumnName)
116 # trainDataRate, validationDataRate 비율로 데이터 나눔
117 trainDataNumber = round(len(feature_dataList) * trainDataRate)
118 validationDataNumber = round(len(feature_dataList) * validationDataRate)
119 # 훈련 데이터 선언
120 xTrainDataList = feature_dataList[:trainDataNumber]
121 yTrainDataList = result_dataList[:trainDataNumber]
122 # 검증 데이터 선언
123 xValidationDataList = feature_dataList[trainDataNumber:trainDataNumber+validationDataNumber]
124 yValidationDataList = result_dataList[trainDataNumber:trainDataNumber+validationDataNumber]
125 # 테스트 데이터 선언
126 xTestDataList = feature_dataList[trainDataNumber+validationDataNumber:]
127 yTestDataList = result_dataList[trainDataNumber+validationDataNumber:]
128
129 print("[TrainData Size]\nx : {}, y : {}".format(len(xTrainDataList),
130                                                  len(yTrainDataList)))
131 print("[ValidationData Size]\nx : {}, y : {}".format(len(xValidationDataList),
132                                                      len(yValidationDataList)))
133 print("[TestData Size]\nx : {}, y : {}".format(len(xTestDataList),
134                                                  len(yTestDataList)))
135
```

Multinomial Classification Softmax Regression 실습

- 빌드단계 - Step2) 모델 생성을 위한 변수 초기화
 - feature_num은 학습데이터의 독립변수 X의 갯수
 - result_num은 종속변수Y(라벨의 종류)의 종류 갯수
 - 학습 데이터 입력공간 X, Y를 placeholder 로 선언
 - W와 b를 값을 저장할 변수를 Variable로 선언

```
136 #####
137 # [빌드단계]
138 # Step 2) 모델 생성을 위한 변수 초기화
139 #####
140 # feature 로 사용할 데이터 갯수
141 feature_num = len(featureColumnName)
142 # result 로 사용할 종류 갯수
143 result_num = len(yLabelList)
144
145 # 학습데이터(x : feature)가 들어갈 플레이스 홀더 선언
146 X = tf.placeholder(tf.float32, shape=[None, feature_num])
147 # 학습데이터(y : result)가 들어갈 플레이스 홀더 선언
148 Y = tf.placeholder(tf.float32, shape=[None, result_num])
149
150 # Weight 변수 선언
151 W = tf.Variable(tf.zeros([feature_num, result_num]))
152 # Bias 변수 선언
153 b = tf.Variable(tf.zeros([result_num]))
154
```

Multinomial Classification Softmax Regression 실습

- 빌드단계 - Step3) 학습 모델 그래프 구성

```
155 #####
156 # [빌드단계]
157 # Step 3) 학습 모델 그래프 구성
158 #####
159 # 3-1) 학습데이터를 대표 하는 가설 그래프 선언
160 hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)
161
162 # 3-2) 비용함수(오차함수, 손실함수) 선언
163 costFunction = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
164
165 # 3-3) 비용함수의 값이 최소가 되도록 하는 최적화함수 선언
166 optimizer = tf.train.GradientDescentOptimizer(learning_rate=learningRate)
167 train = optimizer.minimize(costFunction)
168
```

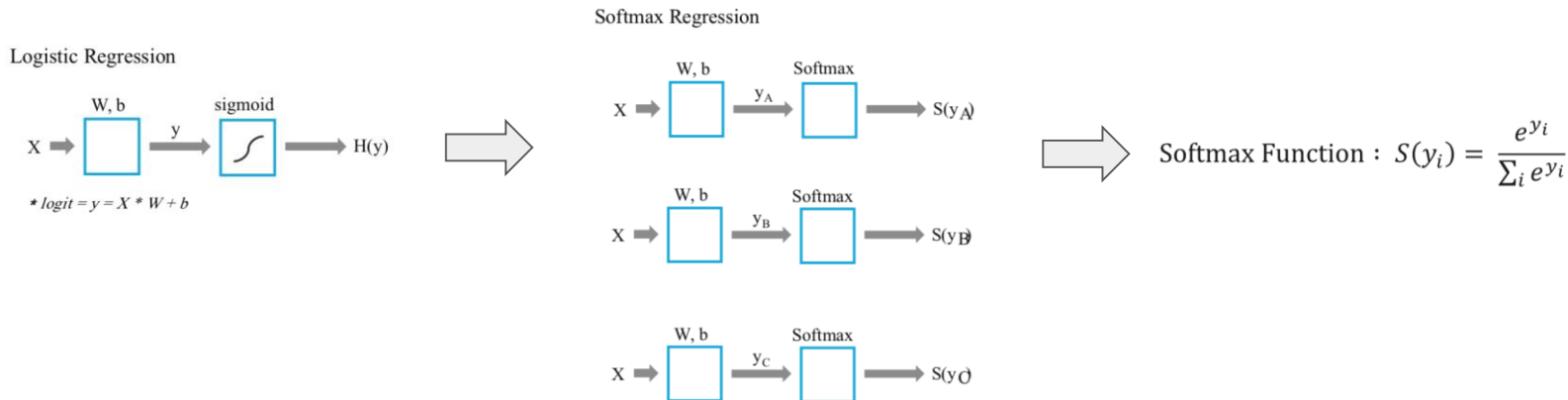
- 학습 데이터의 특성을 대표하는 **가설 수식 작성**
- 가설 수식에 학습데이터 x 의 값을 입력한 결과값(예측값)과 실제값의 오차를 계산하는 **비용함수(오차함수, 손실 함수) 선언**
- 비용함수의 값이 최소가 될 수 있도록 W, b의 최적값을 찾는 **최적화 함수 선언**

Multinomial Classification Softmax Regression 실습 - 가설 선언

- 가설 선언
 - 목적 : Logistic Regression을 출력값의 개수 만큼 사용하여 출력값을 Softmax Function을 이용하여 확률값으로 변경
 - 가설 수식 생성

1) Logistic Regression 가설 수식을 출력값 n개 만큼 사용하여 Matrix Multiplication 형태로 Logit 수식 표현

2) Logit 수식의 값을 Softmax Function에 입력하여 결과를 확률 값으로 변경




Multinomial Classification Softmax Regression 실습 - 가설 수식 정리 과정

- n개의 분류를 위한 Softmax Regression 가설 수식 정리 과정
 - Linear Regression의 Logit(log(odd))수식

$$\text{hypothesis : } P = \frac{1}{1+e^{-(W \times X + b)}}$$
$$\text{logit} = y = X * W + b = \begin{bmatrix} b, & W_1, & W_2, & \dots, & W_n \end{bmatrix} \times \begin{bmatrix} 1 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = [b * 1 + W_1 X_1 + W_2 X_2 + \dots + W_n X_n]$$

- Softmax Regression Logit 수식의 Matrix Multiplication 형태로 변환(n = 3 일때)

$$\begin{aligned} & \begin{bmatrix} b, & W_{A1}, & W_{A2}, & \dots, & W_{An} \end{bmatrix} \times \begin{bmatrix} 1 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = [b * 1 + W_{A1} X_1 + W_{A2} X_2 + \dots + W_{An} X_n] \\ & \begin{bmatrix} b, & W_{B1}, & W_{B2}, & \dots, & W_{Bn} \end{bmatrix} \times \begin{bmatrix} 1 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = [b * 1 + W_{B1} X_1 + W_{B2} X_2 + \dots + W_{Bn} X_n] \\ & \begin{bmatrix} b, & W_{C1}, & W_{C2}, & \dots, & W_{Cn} \end{bmatrix} \times \begin{bmatrix} 1 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = [b * 1 + W_{C1} X_1 + W_{C2} X_2 + \dots + W_{Cn} X_n] \end{aligned}$$

$$\begin{bmatrix} b, & W_{A1}, & W_{A2}, & \dots, & W_{An} \\ b, & W_{B1}, & W_{B2}, & \dots, & W_{Bn} \\ b, & W_{C1}, & W_{C2}, & \dots, & W_{Cn} \end{bmatrix} \times \begin{bmatrix} 1 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} b * 1 + W_{A1} X_1 + W_{A2} X_2 + \dots + W_{An} X_n \\ b * 1 + W_{B1} X_1 + W_{B2} X_2 + \dots + W_{Bn} X_n \\ b * 1 + W_{C1} X_1 + W_{C2} X_2 + \dots + W_{Cn} X_n \end{bmatrix}$$

Multinomial Classification Softmax Regression 실습 - 가설 수식 정리 과정

- n개의 분류를 위한 Softmax Regression 가설 수식 정리 과정
 - Logit 수식을 Softmax Function 에 입력하여 확률값을 출력

$$\text{Softmax Function : } S(y_i) = \frac{e^{y_i}}{\sum_i e^{y_i}}$$

$$\text{logit} = y = X * W + b = \begin{bmatrix} b, W_{A1}, W_{A2}, \dots, W_{An} \\ b, W_{B1}, W_{B2}, \dots, W_{Bn} \\ b, W_{C1}, W_{C2}, \dots, W_{Cn} \end{bmatrix} \times \begin{bmatrix} 1 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} b * 1 + W_{A1}X_1 + W_{A2}X_2 + \dots + W_{An}X_n \\ b * 1 + W_{B1}X_1 + W_{B2}X_2 + \dots + W_{Bn}X_n \\ b * 1 + W_{C1}X_1 + W_{C2}X_2 + \dots + W_{Cn}X_n \end{bmatrix}$$

- 코드로 표현

```
hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)
```


Multinomial Classification Softmax Regression 실습 - Softmax Function

- Softmax Function 수식 정리
- (1) 결괏값(라벨) K개 일때 Odd수식

$$\begin{array}{l} \text{분자: } i \text{ 번째 케이스 확률} \\ \text{분모: 모든 케이스 확률} \end{array} \quad \frac{P(C_i|X)}{P(C_k|X)} = e^{(t_i)} \quad \begin{array}{l} t = \log\left(\frac{y}{1-y}\right) = \text{Logit}(y) \\ e^t = \frac{y}{1-y} \end{array}$$

- (2) 양변에 1번째 부터 k-1번째 까지 더함

$$\sum_{i=1}^{k-1} \frac{P(C_i|X)}{P(C_k|X)} = \sum_{i=1}^{k-1} e^{(t_i)}$$

- (3) 모든 경우의 수 확률의 합은 1이기때문에 1 ~ K-1번째 결괏값의 합은 1 에서 K번째 확률을 빼는 것과 동일
좌변 분자의 수식을 다음과 같이 변경

$$\frac{1 - P(C_k|X)}{P(C_k|X)} = \sum_{i=1}^{k-1} e^{(t_i)}$$

Multinomial Classification Softmax Regression 실습 - Softmax Function

- Softmax Function 수식 정리

(4) 좌변의 분자와 분모를 뒤집어 $P(C_k|X)$ 으로 정리

$$\frac{1 - P(C_k|X)}{P(C_k|X)} = \sum_{i=1}^{k-1} e^{(t_i)} \implies \frac{P(C_k|X)}{1 - P(C_k|X)} = \frac{1}{\sum_{i=1}^{k-1} e^{(t_i)}}$$

=> 좌변의 분모를 양변에 곱함

$$P(C_k|X) = \frac{1 - P(C_k|X)}{\sum_{i=1}^{k-1} e^{(t_i)}}$$

=> 우변의 분모를 양변에 곱함

$$P(C_k|X) * \sum_{i=1}^{k-1} e^{(t_i)} = 1 - P(C_k|X)$$

=> $P(C_k|X)$ 를 좌변으로 넘겨서 정리

$$P(C_k|X) \left(\sum_{i=1}^{k-1} e^{(t_i)} + 1 \right) = 1$$

=> $P(C_k|X)$ 만 남겨서 정리

결과값(라벨)이 K개 일때 Odd 수식 $P(C_k|X) = \frac{1}{\sum_{i=1}^{k-1} e^{(t_i)} + 1}$

Multinomial Classification Softmax Regression 실습 - Softmax Function

- Softmax Function 수식 정리
- (5) 결괏값 K개에 대한 Odd수식을 변형

$$\frac{P(C_i|X)}{P(C_k|X)} = e^{(t_i)} \Rightarrow P(C_k|X) = \frac{P(C_i|X)}{e^{(t_i)}}$$

(6) 최종적으로 i번째의 결괏값(라벨)을 구하기 위하여 (4)에서 정리한 결괏값(라벨)이 K개일때 Odd 수식에 (5)에서 변형한 수식을 대입하여 $P(C_i|X)$ 로 정리

$$P(C_k|X) = \frac{P(C_i|X)}{e^{(t_i)}} \xrightarrow{\text{대입}} P(C_k|X) = \frac{1}{\sum_{i=1}^{k-1} e^{(t^i)} + 1} \Rightarrow \frac{P(C_i|X)}{e^{(t_i)}} = \frac{1}{\sum_{i=1}^{k-1} e^{(t^i)} + 1}$$

=> $P(C_i|X)$ 만 남기고 정리

$$\text{i 번째 결괏값(라벨) 수식} \quad P(C_i|X) = \frac{e^{(t_i)}}{\sum_{i=1}^{k-1} e^{(t^i)} + 1}$$

Multinomial Classification Softmax Regression 실습 - Softmax Function

- Softmax Function 수식 정리
(7) i 번째 결괏값(라벨) 수식의 분모 1을 K번째 결괏값의 확률로 표현하여 수식 정리

$$P(C_i|X) = \frac{e^{(t_i)}}{\sum_{i=1}^{k-1} e^{(t^i)} + \boxed{1}} \Longleftarrow 1 = \frac{P(C_k|X)}{P(C_k|X)} = e^{(t_k)}$$

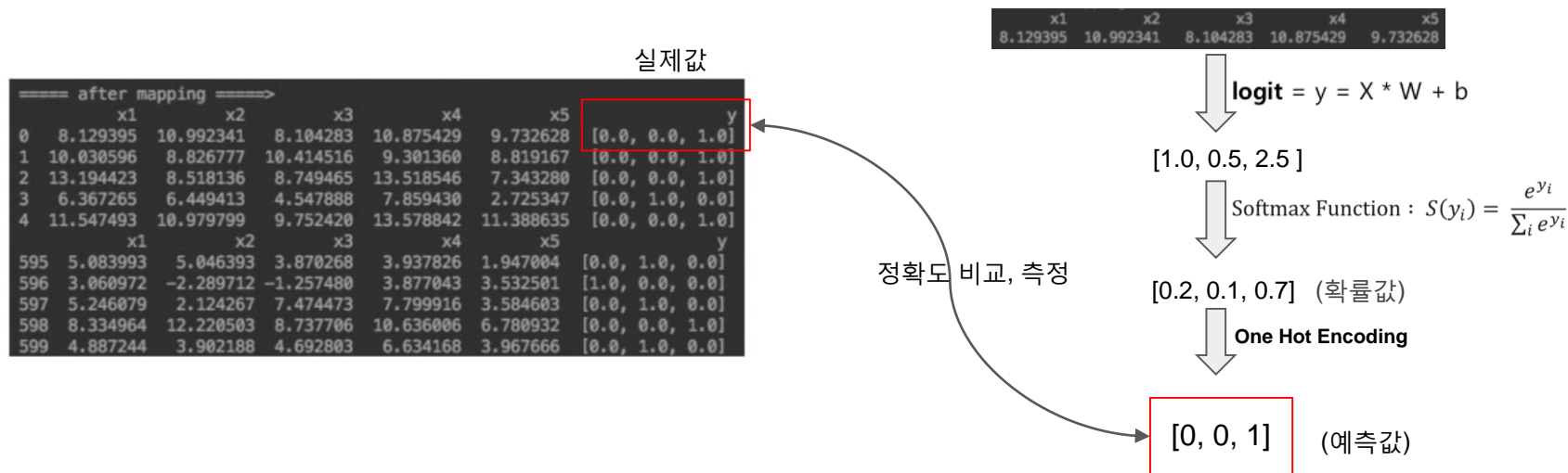
$$P(C_i|X) = \frac{e^{(t_i)}}{\sum_{i=1}^{k-1} e^{(t^i)} + e^{(t_k)}}$$

- (8) 정리된 수식의 분모는 1부터 K-1 번째의 확률값의 합에 K번째 확률을 더하는 수식이기 때문에 분모의 의미는 1부터 K번째 까지의 모든 확률을 더한 수식으로 정리 함

$$P(C_i|X) = \frac{e^{(t_i)}}{\sum_{i=1}^k e^{(t^i)}} \Longrightarrow \boxed{\text{Softmax Function : } S(y_i) = \frac{e^{y_i}}{\sum_i e^{y_i}}}$$

Multinomial Classification Softmax Regression 실습 - One Hot Encoding

- One Hot Encoding
 - 가장 큰 확률을 1로 표현하고 나머지는 0으로 표현
 - 예를들어 [0.2, 0.1, 0.7] 의 값을 가진 결과를 One Hot Encoding을 적용하면 [0, 0, 1] 이 됨
- Softmax Function을 통하여 나온 확률 예측값을 One Hot Encoding 적용하여 값을 변경하고 학습데이터의 실제값(종속변수)과 비교하여 예측 정확도를 측정



Multinomial Classification Softmax Regression 실습 - 비용함수 선언

- 비용함수(오차함수, 손실함수) 선언
 - Softmax Regression의 비용함수는 Cross Entropy(크로스엔트로피)를 사용
 - i 번째 예측값의 Cross Entropy 수식

$$\text{Cross Entropy} : D(S, L) = - \sum_i L_i \log(S_i)$$

S : Softmax Function으로 계산된 예측값
 L : 실제 학습 데이터 값(라벨값)
 D : S 와 L 의 차이(거리, Distance)

- Cross Entropy를 이용한 실제값, 예측값 케이스 정리(K=3일때)

$$\text{Cross Entropy} : D(S, L) = \sum_i L_i * (-\log(S_i))$$

L : 실제값 (라벨값)

$$A = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, C = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- L = A 일 때 S = B로 예측하면 → 예측 실패(비용 함수 값이 커짐)

$$S = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, L_i * (-\log(S_i)) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \odot -\log(s) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \odot \begin{bmatrix} \infty \\ 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 1 * \infty \\ 0 * 0 \\ 0 * \infty \end{bmatrix} = \infty$$

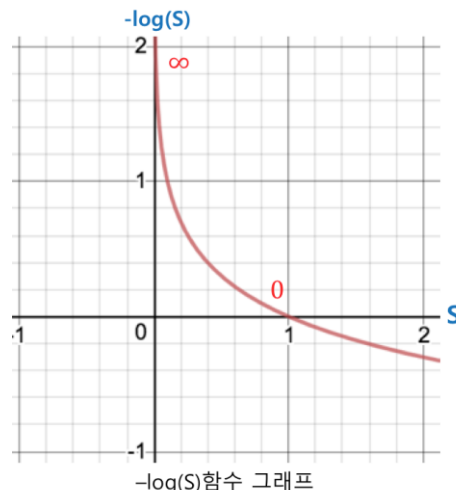
S : 예측값

- L = A 일 때 S = C로 예측하면 → 예측 실패(비용 함수 값이 커짐)

$$S = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, L_i * (-\log(S_i)) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \odot -\log(s) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \odot \begin{bmatrix} \infty \\ \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 1 * \infty \\ 0 * \infty \\ 0 * 0 \end{bmatrix} = \infty$$

- L = A 일 때 S = A로 예측하면 → 예측 성공(비용 함수 값이 작아짐)

$$S = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, L_i * (-\log(S_i)) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \odot -\log(s) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \odot \begin{bmatrix} 0 \\ \infty \\ \infty \end{bmatrix} = \begin{bmatrix} 1 * 0 \\ 0 * \infty \\ 0 * \infty \end{bmatrix} = 0$$



Multinomial Classification Softmax Regression 실습 - 비용함수 선언

- 비용함수(오차함수, 손실함수) 선언
 - N개의 결괏값에 대한 비용함수 수식 일반화

$$costFunction = \frac{1}{N} \sum_i D(S(WX_i + b), L_i)$$

$$Cross\ Entropy : D(S, L) = - \sum_i L_i \log(S_i)$$

S : Softmax Function으로 계산된 예측값
L : 실제 학습 데이터 값(라벨값)
D : S와 L의 차이(거리, Distance)

- 코드로 표현

```
costFunction = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
```

Multinomial Classification Softmax Regression 실습 - 최적화 함수 선언

- 최적화 함수 선언
 - 최적화 알고리즘 Gradient descent 사용

$$W := W - \alpha \frac{\partial}{\partial W} cost(w) \quad \alpha : \text{학습률} \quad \text{비용 함수} - \text{cost}(w) \quad costFunction = \frac{1}{N} \sum_i D(S(WX_i + b), L_i)$$

$$Cross\ Entropy : D(S, L) = - \sum_i L_i \log(S_i)$$

S : Softmax Function으로 계산된 예측값
L : 실제 학습 데이터의 값(라벨값)
D : S와 L의 차이(Distance)

$$W := W - \alpha \frac{\partial}{\partial W} \left(\frac{1}{N} \sum_i D(S(WX_i + b), L_i) \right)$$

- 코드로 표현

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learningRate)
train = optimizer.minimize(costFunction)
```


Multinomial Classification Softmax Regression 실습

- 실행단계 - 학습 모델 그래프 실행

```
169 #####
170 # [실행단계]
171 # 학습 모델 그래프를 실행
172 #####
173 # 실행을 위한 세션 선언
174 sess = tf.Session()
175 # 최적화 과정을 통하여 구해질 변수 W,b 초기화
176 sess.run(tf.global_variables_initializer())
177
178 # 예측값, 정확도 수식 선언
179 predicted = tf.equal(tf.argmax(hypothesis, axis=1), tf.argmax(Y, axis=1))
180 accuracy = tf.reduce_mean(tf.cast(predicted, tf.float32))
181
182 # 학습, 검증 정확도를 저장할 리스트 선언
183 train_accuracy = list()
184 validation_accuracy = list()
185
```

- Session 변수(sess)를 선언
- 최적화 과정에서 계산되는 변수(W, b)의 초기화
- 예측값, 정확도를 구하기 위한 수식 선언(One Hot Encoding이용)
- 모델 학습 결과 확인을 위한 리스트 선언

Multinomial Classification Softmax Regression 실습

- 실행단계 - 학습 모델 그래프 실행

- totalStep 만큼 모델 학습
- 학습을 하면서 중간 결과를 저장하고 출력함
- 학습이 완료된 모델의 W, b 변수의 값을 출력
- 훈련 데이터를 이용하여 모델 학습
- 정확도 결과 확인 그래프 출력

- 학습 조건

- 학습 데이터수 : 600개
 - 훈련 데이터 : 420개
 - 검증 데이터 : 60개
 - 테스트 데이터 : 120개
- 최적화 함수 : Gradient descent
- 학습률 : 0.01
- 학습 횟수 10,001회

```
186 print("-----")
187 print("Train(Optimization) Start ")
188 for step in range(totalStep):
189     # X, Y에 학습데이터 입력하여 비용함수, W, b, accuracy, train을 실행
190     cost_val, W_val, b_val, acc_val, _ = sess.run([costFunction, W, b, accuracy, train],
191                                                    feed_dict={X: xTrainDataList,
192                                                            Y: yTrainDataList})
193     train_accuracy.append(acc_val)
194
195     if step % 1000 == 0:
196         print("step : {}. cost : {}, accuracy : {}".format(step,
197                                                             cost_val,
198                                                             acc_val))
199
200     if step == totalStep-1 :
201         print("W : {}\nb:{}".format(W_val, b_val))
202
203     # matplotlib 를 이용하여 결과를 시각화
204     # 정확도 결과 확인 그래프
205     plt.plot(range(len(train_accuracy)),
206             train_accuracy,
207             linewidth=2,
208             label='Training')
209     plt.legend()
210     plt.title("Train Accuracy Result")
211     plt.show()
212
213     print("Train Finished")
```

Multinomial Classification Softmax Regression 실습

- 실행단계 - 학습 모델 그래프 실행

- 검증 데이터 60개를 이용하여 훈련데이터로 학습된 모델을 재학습하여 결과 확인
- 학습 조건은 동일한 상태에서 학습 데이터 수만 변경됨
- 정확도 결과 확인 그래프 출력

```
214 print("-----")
215 print("Validation Start")
216 for step in range(totalStep):
217     # X, Y에 테스트데이터 입력하여 비용함수, W, b, accuracy, train을 실행
218     cost_val_v, W_val_v, b_val_v, acc_val_v, _ = sess.run([costFunction, W, b, accuracy, train],
219                                                         feed_dict={X: xValidationDataList,
220                                                         Y: yValidationDataList})
221     validation_accuracy.append(acc_val_v)
222
223     if step % 1000 == 0:
224         print("step : {}. cost : {}, accuracy : {}".format(step,
225                                                         cost_val_v,
226                                                         acc_val_v))
227
228     if step == totalStep-1:
229         print("W : {}\nb:{}".format(W_val_v, b_val_v))
230
231
232 # matplotlib 를 이용하여 결과를 시각화
233 # 정확도 결과 확인 그래프
234 plt.plot(range(len(train_accuracy)),
235         train_accuracy,
236         linewidth=2,
237         label='Training')
238 plt.plot(range(len(validation_accuracy)),
239         validation_accuracy,
240         linewidth=2,
241         label='Validation')
242 plt.legend()
243 plt.title("Train and Validation Accuracy Result")
244 plt.show()
245
246 print("Validation Finished")
```

Multinomial Classification Softmax Regression 실습

- 실행단계 - 학습 모델 그래프 실행

```
247 print("-----")
248 print("[Test Result]")
249 # 최적화가 끝난 학습 모델 테스트
250 h_val, p_val, a_val = sess.run([hypothesis, predicted, accuracy],
251                                feed_dict={X: xTestDataList,
252                                            Y: yTestDataList})
253 print("\nHypothesis : {} \nPrediction : {} \nAccuracy : {}".format(h_val,p_val,a_val))
254 print("-----")
255
256 #세션종료
257 sess.close()
```

- 테스트 데이터 120개를 이용하여 학습을 완료한 모델의 결과를 확인함

Multinomial Classification Softmax Regression 실습

- 결과 확인

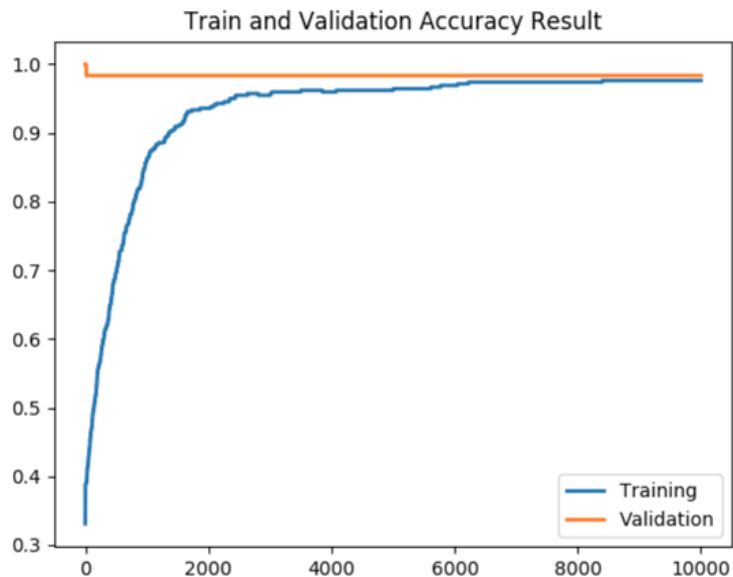
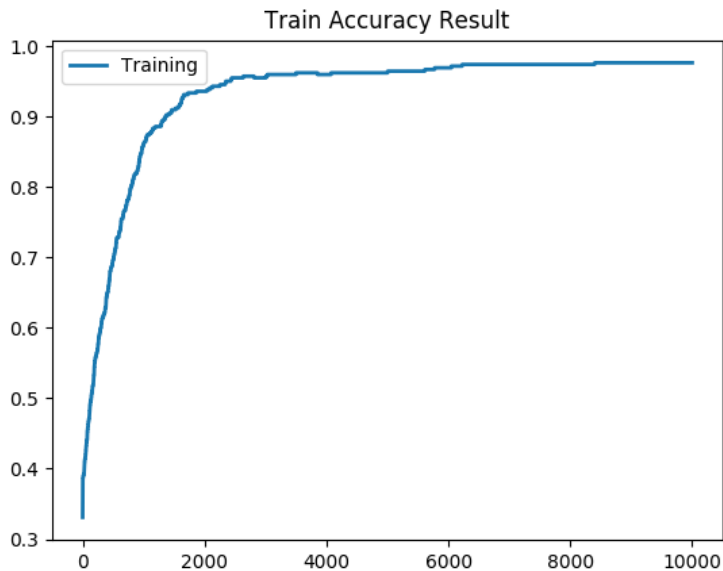
- 훈련 데이터 학습 정확도 : 97.61%
- 검증 데이터 학습 정확도 : 98.33%

```
Train(Optimization) Start
step : 0. cost : 1.098612666130066, accuracy : 0.33095237612724304
step : 1000. cost : 0.5333108901977539, accuracy : 0.8619047403335571
step : 2000. cost : 0.4115321934223175, accuracy : 0.9357143044471741
step : 3000. cost : 0.34549203515052795, accuracy : 0.95476192233594666
step : 4000. cost : 0.30216193199157715, accuracy : 0.9595237970352173
step : 5000. cost : 0.2709597051143646, accuracy : 0.961904764175415
step : 6000. cost : 0.24720239639282227, accuracy : 0.9690476059913635
step : 7000. cost : 0.22841595113277435, accuracy : 0.973809540271759
step : 8000. cost : 0.21314206719398499, accuracy : 0.973809540271759
step : 9000. cost : 0.2004544734954834, accuracy : 0.976190447807312
step : 10000. cost : 0.18973249197006226, accuracy : 0.976190447807312
W : [[-0.21759647  0.01524368  0.20235062]
      [-0.20482178  0.03833586  0.1664812 ]
      [-0.27348518  0.04942771  0.22405553]
      [-0.2634968   0.06629159  0.19720232]
      [-0.29177386  0.04855135  0.24321814]]
b : [ 4.607681  0.56815624 -5.1758313 ]
Train Finished
```

```
Validation Start
step : 0. cost : 0.1819770783185959, accuracy : 1.0
step : 1000. cost : 0.15691377222537994, accuracy : 0.9833333492279053
step : 2000. cost : 0.14913706481456757, accuracy : 0.9833333492279053
step : 3000. cost : 0.1423015594482422, accuracy : 0.9833333492279053
step : 4000. cost : 0.13621042668819427, accuracy : 0.9833333492279053
step : 5000. cost : 0.13074064254760742, accuracy : 0.9833333492279053
step : 6000. cost : 0.12579694390296936, accuracy : 0.9833333492279053
step : 7000. cost : 0.12130295485258102, accuracy : 0.9833333492279053
step : 8000. cost : 0.11719690263271332, accuracy : 0.9833333492279053
step : 9000. cost : 0.11342794448137283, accuracy : 0.9833333492279053
step : 10000. cost : 0.10995392501354218, accuracy : 0.9833333492279053
W : [[-0.28798214  0.08255772  0.2054329 ]
      [-0.28869995  0.14267388  0.14601989]
      [-0.2692163   -0.04666025  0.31586772]
      [-0.30705068  -0.01053992  0.3175857 ]
      [-0.48737264  0.10172389  0.3856425 ]]
b : [ 6.1371307  0.7769367 -6.914061 ]
Validation Finished
```

Multinomial Classification Softmax Regression 실습

- 결과 확인
 - 훈련, 검증 데이터의 학습 정확도 그래프



Multinomial Classification Softmax Regression 실습

- 결과 확인
 - 테스트 데이터를 이용하여 학습 모델 결과를 확인
 - 가설수식의 결과값과 예측결과, 정확도를 출력함
 - 테스트 정확도 : 98%

[Test Result]

```
Hypothesis : [[1.90917682e-03 6.57581806e-01 3.40508997e-01]
[1.19474521e-06 8.05432051e-02 9.19455588e-01]
[9.91785407e-01 8.20888113e-03 5.82922894e-06]
[1.36894300e-06 2.29809389e-01 7.70189285e-01]
[9.94809151e-01 5.18845720e-03 2.41954353e-06]
[3.83183369e-08 5.29373027e-02 9.47062731e-01]
[9.98996913e-01 1.00302976e-03 1.54608529e-07]
```

중간 생략

```
[1.84504949e-02 6.67178333e-01 3.14371139e-01]
[6.93267282e-07 1.23803474e-01 8.76195848e-01]
[2.65607629e-02 8.64640415e-01 1.08798847e-01]]
Prediction : [ True True True True True True True True True True True True
 True True True True True True True True True True True True True
 True True True True True True True True True True True True True
 True True True False True True True True True True True True True
 True True True True True True True True True True True True True
 True True True True True True True True True True True True True
 True True False True True True True True True True True True True
 True True True True True True True True True True True True True
 True True True True True True True True True True True True True]
Accuracy : 0.9833333492279053
```

Iris Softmax Regression

Iris Softmax Regression 개요

- Iris (붓꽃) 학습 데이터
 - 꽃잎의 각 부분의 너비와 길이등을 측정한 데이터
 - 데이터 컬럼

sepalLength (꽃받침 길이)	sepalWidth (꽃받침 너비)	petalLength (꽃잎 길이)	petalWidth (꽃잎 너비)	species (꽃 종류)
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa

- 결괏값(라벨) 데이터 종류 3가지



Iris-setosa

Iris-versicolor

Iris-virginica

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
```

중간 생략

```
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
```

중간 생략

```
5.7,2.8,4.1,1.3,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica
```

Iris 학습 데이터 세트

- 데이터 전처리 과정을 통하여 학습을 할 수 있는 데이터로 변환해야함
- 최종 학습 목표는 꽃잎 데이터를 이용하여 꽃의 종류를 분류 하는 학습 모델 생성

Iris Softmax Regression 실습

- 학습에 필요한 모듈 선언

```
1 #####
2 # [학습에 필요한 모듈 선언]
3 #####
4 import tensorflow as tf
5 import pandas as pd
6 from sklearn.utils import shuffle
7 import matplotlib.pyplot as plt
8 import seaborn as sns
9 import os
10 # requests import error 발생시 pip install requests 로 설치
11 import requests
12
```

- tensorflow, numpy, matplotlib, pandas, sklearn, seaborn 라이브러리 사용
- requests 라이브러리를 이용하여 Iris 데이터를 다운로드 함

Iris Softmax Regression 실습

- 환경설정

- 학습 데이터를 훈련, 테스트 데이터로 분리할 비율 선언
- 학습률, 학습 횟수, 데이터 섞기 여부 선언
- 학습 데이터를 다운 받기 위한 변수 선언

```
13 #####
14 # [환경설정]
15 #####
16 # 학습 데이터(훈련/테스트) 비율
17 trainDataRate = 0.7
18 # 학습률
19 learningRate = 0.01
20 # 총 학습 횟수
21 totalStep = 10001
22 # 데이터 섞기
23 shuffleOn = True
24 # 학습 데이터 파일명 지정
25 fileName = "IrisData.csv"
26 # 학습 데이터 경로 지정
27 currentFolderPath = os.getcwd()
28 dataSetFolderPath = os.path.join(currentFolderPath, 'dataset')
29 dataSetFilePath = os.path.join(dataSetFolderPath, fileName)
30
```

Iris Softmax Regression 실습

- 빌드단계 - Step1) 학습 데이터 준비 : 데이터 읽기

```
31 #####
32 # [빌드단계]
33 # Step 1) 학습 데이터 준비
34 #####
35 ### (1) 데이터 읽어오기
36 # 해당 경로에 학습 데이터가 없으면 다운로드
37 if os.path.exists(datasetFilePath) is not True:
38     print("#===== Download Iris Data =====#")
39     # iris 데이터 셋 다운로드
40     url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
41     req = requests.get(url, allow_redirects=True)
42     # 학습데이터 저장
43     open(datasetFilePath, "wb").write(req.content)
44     print("#===== Download Completed =====#")
45
```

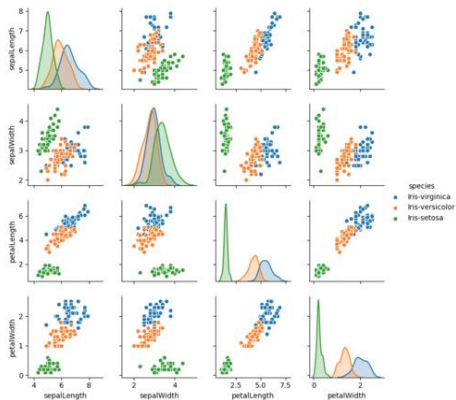
- 환경 설정에서 선언한 학습데이터가 저장되는 위치에 데이터가 없으면 데이터를 다운로드 함

Iris Softmax Regression 실습

- 빌드단계 - Step1) 학습 데이터 준비 : 데이터 읽기

- pandas 를 이용하여 학습 데이터 읽어오며
읽어 올때 데이터를 섞기 여부에 따라
shuffle을 함
- 읽어 드린 학습데이터 확인

```
==== Data =====
  sepalLength sepalWidth petalLength petalWidth species
116      6.5        3.0        5.5        1.8  Iris-virginica
 77      6.7        3.0        5.0        1.7  Iris-versicolor
 64      5.6        2.9        3.6        1.3  Iris-versicolor
 90      5.5        2.6        4.4        1.2  Iris-versicolor
122      7.7        2.8        6.7        2.0  Iris-virginica
 99      5.7        2.8        4.1        1.3  Iris-versicolor
 35      5.0        3.2        1.2        0.2  Iris-setosa
 15      5.7        4.4        1.5        0.4  Iris-setosa
 36      5.5        3.5        1.2        0.2  Iris-setosa
 18      5.7        3.8        1.7        0.3  Iris-setosa
Shape : (150, 5)
Species :
Iris-virginica    50
Iris-versicolor    50
Iris-setosa       50
Name: species, dtype: int64
```



```
46 # pandas를 이용하여 CSV 파일 데이터 읽기
47 allColumnName = ["sepalLength", "sepalWidth", "petalLength", "petalWidth", "species"]
48 # column이 없는 데이터라서 파일을 읽어올때 header 를 생성하지 않고 column을 추가
49 if shuffleOn:
50     df = shuffle(pd.read_csv(datasetFilePath, header=None, names=allColumnName))
51 else:
52     df = pd.read_csv(datasetFilePath, header=None, names=allColumnName)
53
54 # 학습 데이터 확인
55 print("==== Data =====")
56 print(df.head())
57 print(df.tail())
58 # 학습 데이터 shape 확인
59 print("Shape : {}".format(df.shape))
60 # 학습 데이터 결과 갯수 확인
61 print("Species : \n{}".format(df["species"].value_counts()))
62
63
64 # 학습 데이터 전체 그래프
65 sns.pairplot(df, hue="species", height = 2)
66 plt.show()
67
```

Iris Softmax Regression 실습

- 빌드단계 - Step1) 학습 데이터 준비 : 데이터 전처리

```
68 ### (2) 범주형 데이터 맵핑 선언
69 # species 를 3가지 종류로 나눈 dataframe 으로 변환
70 df_one_hot_encoded = pd.get_dummies(df)
71
72 print("==== after mapping =====>")
73 print(df_one_hot_encoded.head())
74 print(df_one_hot_encoded.tail())
75
```

- 데이터 전처리 - 범주형 데이터 맵핑 선언

- specise(꽃 종류) 컬럼의 라벨 데이터를 실수형 데이터로 맵핑
- get_dummies()를 이용하여 꽃의 종류 당 하나의 컬럼나뉘서 총 3개의 컬럼으로 구성
- 해당 꽃의 컬럼의 데이터가 라벨과 같을 경우 1, 아닐경우 0 으로 표현

```
==== Data =====>
      sepallLength  sepalWidth  petalLength  petalWidth  species
116             6.5         3.0         5.5         1.8  Iris-virginica
77              6.7         3.0         5.0         1.7  Iris-versicolor
64              5.6         2.9         3.6         1.3  Iris-versicolor
90              5.5         2.6         4.4         1.2  Iris-versicolor
122             7.7         2.8         6.7         2.0  Iris-virginica
```

맵핑 전

```
==== after mapping =====>
      sepallLength  sepalWidth  petalLength  petalWidth  species_Iris-setosa  \
116             6.5         3.0         5.5         1.8                0
77              6.7         3.0         5.0         1.7                0
64              5.6         2.9         3.6         1.3                0
90              5.5         2.6         4.4         1.2                0
122             7.7         2.8         6.7         2.0                0

      species_Iris-versicolor  species_Iris-virginica
116                        0                      1
77                         1                      0
64                         1                      0
90                         1                      0
122                        0                      1
```

맵핑 후

Iris Softmax Regression 실습

- 빌드단계 - Step1) 학습 데이터 준비

- 훈련 데이터와 테스트 데이터를 7:3의 비율로 설정
- pandas의 sample(), drop(), filter() 함수를 이용하여 데이터 추출
- pandas Dataframe DOC : <https://pandas.pydata.org/pandas-docs/stable/reference/frame.html>

```
76 ### (3) 훈련, 테스트 데이터 나누기
77 # 학습 데이터 리스트로 변환
78 # 훈련 데이터를 정해진 비율만큼 추출
79 df_trainData = df_one_hot_encoded.sample(frac=trainDataRate)
80
81 # 훈련 데이터를 제거한 나머지 데이터를 테스트 데이터로 지정
82 df_testData = df_one_hot_encoded.drop(df_trainData.index)
83
84 # 학습데이터와 결과데이터의 컬럼 선언
85 featureColumnName = ['sepalLength', 'sepalWidth', 'petalLength', 'petalWidth']
86 resultColumnName = ['species_Iris-setosa', 'species_Iris-versicolor', 'species_Iris-virginica']
87 # 학습데이터 선언
88 xTrainDataList = df_trainData.filter(featureColumnName)
89 yTrainDataList = df_trainData.filter(resultColumnName)
90 # 테스트 데이터 선언
91 xTestDataList = df_testData.filter(featureColumnName)
92 yTestDataList = df_testData.filter(resultColumnName)
93
94 print("[TrainData Size] x : {}, y :{}".format(len(xTrainDataList),
95                                              len(yTrainDataList)))
96 print("[TestData Size] x : {}, y :{}".format(len(xTestDataList),
97                                              len(yTestDataList)))
98
```

Iris Softmax Regression 실습

- 빌드단계 - Step2) 모델 생성을 위한 변수 초기화

```
99 #####
100 # [빌드단계]
101 # Step 2) 모델 생성을 위한 변수 초기화
102 #####
103 # feature 로 사용할 데이터 갯수
104 feature_num = len(featureColumnName)
105 # result 로 사용할 종류 갯수
106 result_num = len(resultColumnName)
107
108 # 학습데이터가 들어갈 플레이스 홀더 선언
109 X = tf.placeholder(tf.float32, shape=[None, feature_num])
110 # 학습데이터가 들어갈 플레이스 홀더 선언
111 Y = tf.placeholder(tf.float32, shape=[None, result_num])
112
113 # Weight 변수 선언
114 W = tf.Variable(tf.zeros([feature_num, result_num]))
115 # Bias 변수 선언
116 b = tf.Variable(tf.zeros([result_num]))
117
```

- feature_num은 학습데이터의 꽃의 특성 데이터 4개
- result_num은 꽃의 종류 3가지
- 학습 데이터 입력공간 X, Y를 placeholder 로 선언
- W와 b를 값을 저장할 변수를 Variable로 선언

Iris Softmax Regression 실습

- 빌드단계 - Step3) 학습 모델 그래프 구성

```
118 #####
119 # [빌드단계]
120 # Step 3) 학습 모델 그래프 구성
121 #####
122 # 3-1) 학습데이터를 대표 하는 가설 그래프 선언
123 hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)
124
125 # 3-2) 비용함수(오차함수, 손실함수) 선언
126 costFunction = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
127
128 # 3-3) 비용함수의 값이 최소가 되도록 하는 최적화함수 선언
129 optimizer = tf.train.GradientDescentOptimizer(learning_rate=learningRate)
130 train = optimizer.minimize(costFunction)
131
```

- 학습 데이터의 특성을 대표하는 **가설 수식 작성**
- 가설 수식에 학습데이터 x 의 값을 입력한 결과값(예측값)과 실제값의 오차를 계산하는 **비용함수(오차함수, 손실 함수) 선언**
- 비용함수의 값이 최소가 될 수 있도록 W, b의 최적값을 찾는 **최적화 함수 선언**

Iris Softmax Regression 실습

- 실행단계 - 학습 모델 그래프 실행

```
132 #####
133 # [실행단계]
134 # 학습 모델 그래프를 실행
135 #####
136 # 실행을 위한 세션 선언
137 sess = tf.Session()
138 # 최적화 과정을 통하여 구해질 변수 W, b 초기화
139 sess.run(tf.global_variables_initializer())
140
141 # 예측값, 정확도 수식 선언
142 predicted = tf.equal(tf.argmax(hypothesis, axis=1), tf.argmax(Y, axis=1))
143 accuracy = tf.reduce_mean(tf.cast(predicted, tf.float32))
144
145 # 학습, 테스트 정확도를 저장할 리스트 선언
146 train_accuracy = list()
147
```

- Session 변수(sess)를 선언
- 최적화 과정에서 계산되는 변수(W, b)의 초기화
- 예측값, 정확도를 구하기 위한 수식 선언(One Hot Encoding이용)
- 모델 학습 결과 확인을 위한 리스트 선언

Iris Softmax Regression 실습

- 실행단계 - 학습 모델 그래프 실행
 - totalStep 만큼 모델 학습
 - 학습을 하면서 중간 결과를 저장하고 출력함
 - 학습이 완료된 모델의 W, b 변수의 값을 출력
 - 훈련 데이터를 이용하여 모델 학습
 - 정확도 결과 확인 그래프 출력
- 학습 조건
 - 학습 데이터수 : 150개
 - 최적화 함수 : Gradient descent
 - 학습률 : 0.01
 - 학습 횟수 10,001회

```
148 print("-----")
149 print("Train(Optimization) Start ")
150
151 for step in range(totalStep):
152     # X, Y에 학습데이터 입력하여 비용함수, W, b, accuracy, train을 실행
153     cost_val, W_val, b_val, acc_val, _ = sess.run([costFunction, W, b, accuracy, train],
154                                                    feed_dict={X: xTrainDataList,
155                                                            Y: yTrainDataList})
156
157     train_accuracy.append(acc_val)
158
159     if step % 1000 == 0:
160         print("step : {}, cost : {}, accuracy : {}".format(step,
161                                                                cost_val,
162                                                                acc_val))
163
164     if step == totalStep-1 :
165         print("W : {}\nb:{}".format(W_val, b_val))
166
167     # matplotlib 를 이용하여 결과를 시각화
168     # 정확도 결과 확인 그래프
169     plt.plot(range(len(train_accuracy)),
170              train_accuracy,
171              linewidth=2,
172              label='Training')
173
174     plt.legend()
175     plt.title("Train Accuracy Result")
176     plt.show()
177
178 print("Train Finished")
```

Iris Softmax Regression 실습

- 실행단계 - 학습 모델 그래프 실행

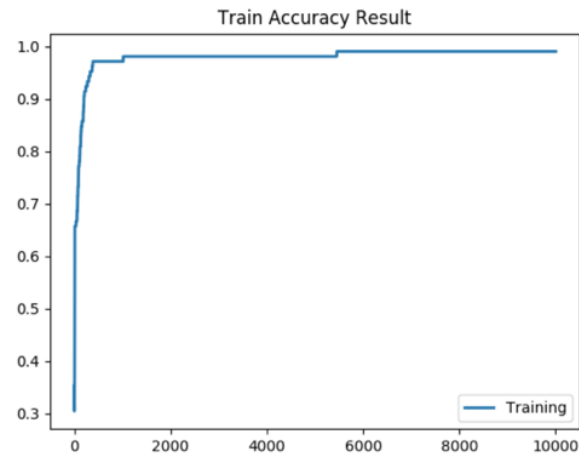
```
177 print("-----")
178 print("[Test Result]")
179 # 최적화가 끝난 학습 모델 테스트
180 h_val, p_val, a_val = sess.run([hypothesis, predicted, accuracy],
181                                feed_dict={X: xTestDataList,
182                                            Y: yTestDataList})
183 print("\nHypothesis : {} \nPrediction : {} \nAccuracy : {}".format(h_val,p_val,a_val))
184 print("-----")
185
186 #세션종료
187 sess.close()
```

- 테스트 데이터를 이용하여 학습을 완료한 모델의 결과를 확인함

Iris Softmax Regression 실습

- 결과 확인
 - 훈련 데이터 학습 정확도 : 99%

```
Train(Optimization) Start
step : 0. cost : 1.0986120700836182, accuracy : 0.3523809611797333
step : 1000. cost : 0.3588559329509735, accuracy : 0.9714285731315613
step : 2000. cost : 0.2723434865474701, accuracy : 0.9809523820877075
step : 3000. cost : 0.2252206951379776, accuracy : 0.9809523820877075
step : 4000. cost : 0.19521349668502808, accuracy : 0.9809523820877075
step : 5000. cost : 0.1743469089269638, accuracy : 0.9809523820877075
step : 6000. cost : 0.15893900394439697, accuracy : 0.9904761910438538
step : 7000. cost : 0.147054985165596, accuracy : 0.9904761910438538
step : 8000. cost : 0.13758142292499542, accuracy : 0.9904761910438538
step : 9000. cost : 0.12983213365077972, accuracy : 0.9904761910438538
step : 10000. cost : 0.12336087226867676, accuracy : 0.9904761910438538
W : [[ 0.90809953  0.6953396 -1.6034315 ]
      [ 2.0654657 -0.14330748 -1.9221547 ]
      [-2.8290837 -0.13638029  2.9654355 ]
      [-1.2990803 -1.1405295  2.4396093 ]]
b:[ 0.4441536  0.6295711 -1.0737243]
Train Finished
```



Iris Softmax Regression 실습

- 결과 확인
 - 테스트 데이터를 이용하여 학습 모델 결과를 확인
 - 가설수식의 결과값과 예측결과, 정확도를 출력함
 - 테스트 정확도 : 97%

[Test Result]

```
Hypothesis : [[1.55100890e-03 3.76837194e-01 6.21611774e-01]
[2.30081932e-05 3.18000056e-02 9.68177021e-01]
[2.39924975e-02 9.51164842e-01 2.48426218e-02]
[6.90674642e-05 1.16742425e-01 8.83188486e-01]
[2.86685216e-04 1.59592390e-01 8.40120971e-01]
[1.35160444e-05 8.20332766e-02 9.17953193e-01]
[2.05565058e-02 9.52330828e-01 2.71126367e-02]
[1.07099554e-02 9.69590485e-01 1.96995996e-02]
[6.61427015e-03 9.51024532e-01 4.23611216e-02]
[3.05886730e-03 9.15141106e-01 8.17999914e-02]
[9.92067814e-01 7.93213397e-03 3.64008845e-09]
[1.97639763e-02 9.25044894e-01 5.51911667e-02]
```

중간 생략

```
[1.21808887e-06 4.04792419e-03 9.95950818e-01]
[7.01073324e-04 2.73824573e-01 7.25474298e-01]
[1.45584764e-02 9.57696497e-01 2.77450513e-02]]
Prediction : [ True True True True True True True True True True True True
 True True True True True True True True True False True True True
 True True True True True True True True True True True True True
 True True True True True True True True True True]
Accuracy : 0.9777777791023254
```