|  |  |
| --- | --- |
| 실습보고서 | |
| 실습03: 로지스틱 회귀 | |
| 학번:2019146037 | 이름:홍석영 |

\*주의사항

- 양식 및 폰트 변경하지 않고 사용할 것

**실습 01**

|  |
| --- |
| 코드 (텍스트 형태로 copy할 것, 폰트크기 9pt) |
| import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  raw\_data = pd.read\_csv('logistic\_regression\_data.csv',names = ['n','x','y','z'])  number = np.asarray(raw\_data['n'].values.tolist())  x\_data = np.asarray(raw\_data['x'].values.tolist())  y\_data = np.asarray(raw\_data['y'].values.tolist())  NegPos\_data = np.asarray(raw\_data['z'].values.tolist())  Neg\_x = np.zeros(250)  Neg\_y = np.zeros(250)  Pos\_x = np.zeros(250)  Pos\_y = np.zeros(250)  a = 0  b = 0  for i in range(500):  if NegPos\_data[i] == 0:  Neg\_x[a] = x\_data[i]  Neg\_y[a] = y\_data[i]  a = a+1  else:  Pos\_x[b] = x\_data[i]  Pos\_y[b] = y\_data[i]  b = b+1  plt.plot(Neg\_x,Neg\_y,'ro',Pos\_x,Pos\_y,'bx')  plt.legend(['0','1'])  plt.xlabel('x')  plt.ylabel('y')  plt.grid(True)  plt.show() |

|  |
| --- |
| 그래프 (이미지 copy할 것) |
|  |

|  |
| --- |
| 설명 (본 실습 과제의 중요 이론 및 결과를 간략히 설명) |
| 엑셀(csv) 파일에 들어있는 첫번째 열 데이터(나이)와 두번째 열 데이터(키)를 ‘np.panda’ 를 사용하여 불러오고, 나이와 키를 ‘np.asarray’를 사용하여 각 열을 차례로 n데이터(number), x데이터(x\_data), y데이터(y\_data), z데이터(NegPos\_data)로 지정하였습니다.  z데이터(NegPos\_data)의 ‘0’, ‘1’ 결과에 따라 입력값을 나눠야 하기 때문에 ‘for’문 안에 조건문을 사용하여 ‘0’일때 x\_data를 Neg\_x, y\_data를 Neg\_y에 넣고 ‘1’인 경우 x\_data를 Pos\_x, y\_data를 Pos\_y 넣습니다.  ‘plot’을 사용하여 각 데이터의 위치를 빨간 o(클래스 0), 파란 x(클래스 1)로 그래프를 표현하였습니다.  이 과정을 통해 입력이 2개(x\_data, y\_data) 출력이 1개(NegPos\_data)인 데이터를 정리 하였습니다. |

**실습 02**

|  |
| --- |
| 코드 (텍스트 형태로 copy할 것, 폰트크기 9pt) |
| import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  raw\_data = pd.read\_csv('logistic\_regression\_data.csv',names = ['n','x','y','z'])  number = np.asarray(raw\_data['n'].values.tolist())  x\_data = np.asarray(raw\_data['x'].values.tolist())  y\_data = np.asarray(raw\_data['y'].values.tolist())  NegPos\_data = np.asarray(raw\_data['z'].values.tolist())  w0\_first = 0.1  w1\_first = 0.1  w2\_first = 0.1  LR = 0.0043  entropy = np.zeros(10000)  for i in range(10000):    Zn = w0\_first\*x\_data+w1\_first\*y\_data+w2\_first  Pn = 1/(1+np.exp(-Zn))    error = Pn - NegPos\_data    w0\_diff = np.mean(error\*x\_data)  w1\_diff = np.mean(error\*y\_data)  w2\_diff = np.mean(error)    w0\_first = w0\_first - LR\*w0\_diff  w1\_first = w1\_first - LR\*w1\_diff  w2\_first = w2\_first - LR\*w2\_diff    entropy[i] = -np.mean(NegPos\_data\*np.log(Pn)+(1-NegPos\_data)\*np.log(1-Pn))  t = np.linspace(0, 10000, 10000)  plt.plot(t,entropy,'b-')  plt.legend(['Mean cross entropy error'])  plt.xlabel('Repeat count')  plt.ylabel('Mean cross entropy error')  plt.grid(True)  plt.show() |

|  |
| --- |
| 그래프 |
|  |

|  |
| --- |
| 설명 (본 실습 과제의 중요 이론 및 결과를 간략히 설명) |
| 학습률(LR) = 0.0043 , 초기값(w0\_first, w1\_first,w2\_first) = 0.1, 반복 횟수 = 10000 으로 하여 경사하강법을 실행하였습니다.  error는 로지스틱 함수(Pn)와 데이터의 출력값(NegPos\_data)의 차이입니다.  w0\_diff은 w0를 미분하여 구한 기울기 값입니다.(error와 x\_data 곱의 평균)  w1\_diff은 w1을 미분하여 구한 기울기 값입니다.(error와 y\_data 곱의 평균)  w2\_diff은 w2을 미분하여 구한 기울기 값입니다.(error의 평균)  ‘w0\_first = w0\_first - LR\*w0\_diff’은 w0[t+1]을 구하는 과정입니다.  ‘w1\_first = w1\_first - LR\*w1\_diff’은 w1[t+1]을 구하는 과정입니다.  ‘w2\_first = w2\_first - LR\*w2\_diff’은 w2[t+1]을 구하는 과정입니다.  평균 교차엔트로피오차(entropy)는 ‘-np.mean(NegPos\_data\*np.log(Pn)+(1-NegPos\_data)\*np.log(1-Pn))’을 통해 구하였고 평균에 (-)가 붙는 이유는 로그 가능도가 매개변수에 대해 음의 볼록한 함수 임으로 부호를 바꾸어 양의 오목한 함수로 바꾸기 위해 붙습니다.  경사하강법의 반복 횟수가 증가할수록 평균 교차엔트로피 오차의 값은 점점 낮아지는 것을 확인할 수 있습니다. 이를 통해 반복 횟수와 평균 교차엔트로피 오차의 값은 반비례 관계임을 알 수 있고, 이는 반복횟수가 증가할 수록 모델 정확도는 점점 증가한다고 할 수 있습니다. |

**실습 03**

|  |
| --- |
| 코드 (텍스트 형태로 copy할 것, 폰트크기 9pt) |
| import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  raw\_data = pd.read\_csv('logistic\_regression\_data.csv',names = ['n','x','y','z'])  number = np.asarray(raw\_data['n'].values.tolist())  x\_data = np.asarray(raw\_data['x'].values.tolist())  y\_data = np.asarray(raw\_data['y'].values.tolist())  NegPos\_data = np.asarray(raw\_data['z'].values.tolist())  w0\_first = 0.1  w1\_first = 0.1  w2\_first = 0.1  de\_w0 = [0]\*10000  de\_w1 = [0]\*10000  de\_w2 = [0]\*10000  LR = 0.0043  for i in range(10000):    Zn = w0\_first\*x\_data+w1\_first\*y\_data+w2\_first  Pn = 1/(1+np.exp(-Zn))    error = Pn - NegPos\_data    w0\_diff = np.mean(error\*x\_data)  w1\_diff = np.mean(error\*y\_data)  w2\_diff = np.mean(error)    w0\_first = w0\_first - LR\*w0\_diff  de\_w0[i] = w0\_first    w1\_first = w1\_first - LR\*w1\_diff  de\_w1[i] = w1\_first    w2\_first = w2\_first - LR\*w2\_diff  de\_w2[i] = w2\_first  print(w0\_first)  print(w1\_first)  print(w2\_first)  t = np.linspace(0, 10000, 10000)  plt.plot(t,de\_w0,'bo',t,de\_w1,'go',t,de\_w2,'mo')  plt.legend(['w0','w1','w2'])  plt.xlabel('Repeat count')  plt.ylabel('Parameter')  plt.grid(True)  plt.show() |

|  |
| --- |
| 그래프 |
|  |

|  |
| --- |
| 최적 매개변수 및 설명 |
| 최적 매개변수는 [ = 0.60590427838293], [ = 0.624056433802464], [ = -3.4050415141501635] 입니다.  경사하강법의 초기값, 학습률, 반복 횟수 코드는 [실습 2]와 같은 코드를 사용하여 만들었습니다.  프로그램 실습 결과 반복 횟수가 증가할 수록 평균 교차엔트로피 오차의 값은 점점 감소하므로 매개변수 값은 최적 매개변수의 값에 수렴한다고 할 수 있습니다.  [실습 2]에서 추가된 코드로는 10000번을 반복하면서 변화되는 매개변수 값을 전부 확인하기 위하여 ‘de\_w0 = [0]\*10000, de\_w1 = [0]\*10000, de\_w2 = [0]\*10000’의 배열 리스트를 만든다음 for문 안에 한번씩 반복될 때 마다 w0,w1,w2의 값을 넣어줌으로써 해결하였습니다.  그 다음 ‘plot’을 이용하여 x축은 0 ~ 10000까지 같은 간격을 같는 10000개의 원소 t로 설정하였고(t = np.linspace(0,10000,10000)), y축은 ‘de\_w0’을 파란점(w0), ‘de\_w1’을 초록점(w1), ‘de\_w2’을 자홍점(w2)으로 그래프를 구현하였습니다. |

**실습 04**

|  |
| --- |
| 코드 (텍스트 형태로 copy할 것, 폰트크기 9pt) |
| import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  raw\_data = pd.read\_csv('logistic\_regression\_data.csv',names = ['n','x','y','z'])  number = np.asarray(raw\_data['n'].values.tolist())  x\_data = np.asarray(raw\_data['x'].values.tolist())  y\_data = np.asarray(raw\_data['y'].values.tolist())  NegPos\_data = np.asarray(raw\_data['z'].values.tolist())  w0\_first = 0.1  w1\_first = 0.1  w2\_first = 0.1  entropy = np.zeros(10000)  LR = 0.0043  for i in range(10000):    Zn = w0\_first\*x\_data+w1\_first\*y\_data+w2\_first  Pn = 1/(1+np.exp(-Zn))    error = Pn - NegPos\_data    w0\_diff = np.mean(error\*x\_data)  w1\_diff = np.mean(error\*y\_data)  w2\_diff = np.mean(error)    w0\_first = w0\_first - LR\*w0\_diff  w1\_first = w1\_first - LR\*w1\_diff  w2\_first = w2\_first - LR\*w2\_diff    entropy[i] = -np.mean(NegPos\_data\*np.log(Pn)+(1-NegPos\_data)\*np.log(1-Pn))    logistic\_regression = 1/(1+np.exp(-Zn))  y\_hat = np.zeros(500)  for i in range(500):  if logistic\_regression[i] < 0.5:  y\_hat[i] = 0  else:  y\_hat[i] = 1  conformity = 0  for i in range(500):  if y\_hat[i] == NegPos\_data[i]:  conformity = conformity + 1  else:  conformity = conformity  accuracy = (conformity/500)\*100  print(accuracy) |

|  |
| --- |
| 정확도 및 설명 |
| 정확도는 90.60000000000001% 입니다.  ‘if’문을 사용하여 로지스틱 회귀 모델(Pn=logistic\_regression)의 값(사후확률)이 0.5보다 작을 때 ‘0’으로 설정하고, 0.5보다 클 때 ‘1’로 설정하여 그 값들을 y\_hat에 대입합니다. 그리고, y\_hat과 데이터 출력값(NegPos\_data)와 비교하여 같으면 일치성 변수(conformity)의 값을 ‘1’ 증가 시키고 같지 않다면 증가시키지 않으면서 몇 개의 데이터가 일치하는지 찾습니다. 그 이후에 (일치하는 데이터 개수/출력 데이터 값의 개수)\*100을 하여 정확도(accuracy)를 구할 수 있습니다. |

**실습 05**

|  |
| --- |
| 코드 (텍스트 형태로 copy할 것, 폰트크기 9pt) |
| import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  raw\_data = pd.read\_csv('logistic\_regression\_data.csv',names = ['n','x','y','z'])  number = np.asarray(raw\_data['n'].values.tolist())  x\_data = np.asarray(raw\_data['x'].values.tolist())  y\_data = np.asarray(raw\_data['y'].values.tolist())  NegPos\_data = np.asarray(raw\_data['z'].values.tolist())  Neg\_x = np.zeros(250)  Neg\_y = np.zeros(250)  Pos\_x = np.zeros(250)  Pos\_y = np.zeros(250)  a = 0  b = 0  for i in range(500):  if NegPos\_data[i] == 0:  Neg\_x[a] = x\_data[i]  Neg\_y[a] = y\_data[i]  a = a+1  else:  Pos\_x[b] = x\_data[i]  Pos\_y[b] = y\_data[i]  b = b+1  w0\_first = 0.1  w1\_first = 0.1  w2\_first = 0.1  entropy = np.zeros(10000)  LR = 0.0043  for i in range(10000):    Zn = w0\_first\*x\_data+w1\_first\*y\_data+w2\_first  Pn = 1/(1+np.exp(-Zn))    error = Pn - NegPos\_data    w0\_diff = np.mean(error\*x\_data)  w1\_diff = np.mean(error\*y\_data)  w2\_diff = np.mean(error)    w0\_first = w0\_first - LR\*w0\_diff  w1\_first = w1\_first - LR\*w1\_diff  w2\_first = w2\_first - LR\*w2\_diff    entropy[i] = -np.mean(NegPos\_data\*np.log(Pn)+(1-NegPos\_data)\*np.log(1-Pn))    x1 = np.zeros(500)  x1 = -(w0\_first/w1\_first\*x\_data)-(w2\_first/w1\_first)  plt.plot(Neg\_x,Neg\_y,'ro',Pos\_x,Pos\_y,'bx',x\_data,x1,'k-')  plt.legend(['0','1','crystalline boundary'])  plt.xlabel('x')  plt.ylabel('y')  plt.grid(True)  plt.show() |

|  |
| --- |
| 그래프 |
|  |

|  |
| --- |
| 설명 |
| 모델의 결정경계는 [실습 3]의 코드에서 Zn의 값이 0이 될 때 결정경계라고 할 수 있습니다.  따라서 결정경계는 x1 = - 이며, 다시 말해 입력속성의 평면에서 기울기가 이고 절편이 인 직선입니다.  이 직선을 통해 입력속성들의 평면을 양성 영역과 음성 영역으로 분할 할 수 있습니다.  그래프를 보면 직선을 어떻게 옮겨도 모든 데이터를 정확하게 구분할 수 있는 직선을 찾을 수 없는데 이것이 로지스틱 회귀의 속성공간 선형분할 특성과 그에 따른 근복적인 한계입니다.  그래프는 ‘plot’을 이용하여 {x가 클래스 ‘0’인 x\_data(Neg\_x), y가 클래스 ‘0’인 y\_data(Neg\_y)를 빨간O(0)으로 표현하고}, {x가 클래스 ‘1’인 x\_data(Pos\_x), y가 클래스 ‘1’인 y\_data(Pos\_y)를 파란X(1)로 표현하고}, {x가 첫번째 입력 데이터(x\_data), y가 x1(= - )을 검은선(crystalline boundary)으로 표현하였습니다.} |