|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 211019 |
| 교육장소 | 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| 1. 로지스틱 회귀(Logistic regression)   이름은 회귀이지만 분류 모델. 선형 회귀와 동일하게 선형 방정식을 학습함.  예 )  캡처.JPG   * 여기서 a, b, c, d, e는 가중치 혹은 계수임. Z는 어떤 값도 가능하지만 확률이 되려면 0~1(또는 0~100%)사이 값이 되어야 함. Z가 아주 큰 음수일 때 0이 되고 z가 아주 큰 양수일 때 1이 되도록 바꾸는 방법이 바로 로지스틱 함수(시그모이드 함수).   캡처.JPG   * 왼쪽의 함수식을 이용하면 오른쪽 같은 그래프가 만들어진다. Z가 무한하게 큰 음수일 경우 0에 가까워지고 z가 무한하게 큰 양수일 경우 1에 가까워지며 z가 0이 될 때는 0.5가 됨. 그러므로 0~1 사이 값을 0~100%까지 확률로 해석할 수 있음. * 넘파이를 사용하면 이 그래프를 간단히 그릴 수 있음. -5와 5 사이에 0.1 간격으로 배열 z를 만들고 z위치마다 시그모이드 함수를 계산함. 지수함수 계산은 np.exp()함수를 사용함.   예 )  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt fish = pd.read\_csv('https://bit.ly/fish\_csv\_data')  fish.head()  print(fish.shape)  fish['Species'].value\_counts()  fish\_input = fish[["Weight","Length","Diagonal",'Height','Width']].to\_numpy()  # fish\_input[:5]  fish\_target = fish['Species'].to\_numpy()  fish\_target[:10]  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target = train\_test\_split(  fish\_input, fish\_target, stratify=fish\_target)  from sklearn.preprocessing import StandardScaler  ss = StandardScaler()  ss.fit(train\_input)  train\_scaled = ss.transform(train\_input)  test\_scaled = ss.transform(test\_input)  z = np.arange(-5, 5, 0.1) # 세타, 곡선모양에 영향  prob\_y = 1/(1 + np.exp(-z))  plt.plot(z, prob\_y)  plt.axhline(1, linestyle = "--", color = 'r')  plt.axhline(0, linestyle = "--", color = 'r')  plt.axhline(0.5, linestyle = "--", color = 'b')  plt.show()  # Boolean index  bream\_smelt\_index = (train\_target == "Bream") | (train\_target == "Smelt")  # bream이랑 smelt만 True  bream\_smelt\_index  train\_target  train\_bream\_smelt = train\_scaled[bream\_smelt\_index]  target\_bream\_smelt = train\_target[bream\_smelt\_index]  print(np.shape(train\_scaled))  print(np.shape(train\_bream\_smelt))  from sklearn.linear\_model import LogisticRegression  lr = LogisticRegression()  lr.fit(train\_bream\_smelt, target\_bream\_smelt)  lr.score(train\_bream\_smelt, target\_bream\_smelt)  print(train\_bream\_smelt[:5])  print(lr.predict\_proba(train\_bream\_smelt[:5]))  print(lr.predict(train\_bream\_smelt[:5]))  print("Bream = 0, Smelt = 1")  print(lr.coef\_, lr.intercept\_)  # p(x) = 1/ (1 + e^-(-2.1-0.44x1-0.61x2,,, -0.74x5))  # C=1 / lambda  lr = LogisticRegression(C = 20, max\_iter = 1000)  lr.fit(train\_scaled, train\_target)  print(lr.score(train\_scaled, train\_target))  print(lr.score(test\_scaled, test\_target))  print(lr.predict(test\_scaled[:10]))  test\_target[:10]  lr.predict\_proba(test\_scaled[:5]).round(3)   1. 경사 하강법(Gradient descent)   앞서 훈련한 모델을 버리지 않고 새로운 데이터에 대해서만 조금씩 더 훈련하는 훈련방식을 점진적 학습 또는 온라인 학습이라고 부르며, 대표적인 점진적 학습 알고리즘은 확률적 경사 하강법(Stochastic Gradient Descent).   * 확률적 경사 하강법   전체 샘플을 사용하지 않고 딱 하나의 샘플을 훈련 세트에서 랜덤하게 골라 가장 가파른 기울기울 찾는 것. 훈련 세트에서 랜덤하게 하나씩 샘플을 선택, 경사를 조금씩 내려간다.  예) import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  fish = pd.read\_csv('https://bit.ly/fish\_csv\_data')  fish\_input = fish[['Weight','Length','Diagonal','Height','Width']].to\_numpy()  fish\_target = fish['Species'].to\_numpy()  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target = train\_test\_split(  fish\_input, fish\_target, stratify= fish\_target)  # 표 준 화  from sklearn.preprocessing import StandardScaler  ss = StandardScaler()  ss.fit(train\_input)  train\_scaled = ss.transform(train\_input)  test\_scaled = ss.transform(test\_input)  from sklearn.linear\_model import SGDClassifier  sc = SGDClassifier(loss = "log", max\_iter= 100)  # max\_iter : 경사하강법에서 100번까지 연산한다.  #기울기 보면서 플러스 마이너스 100번까지만 하면 대충 근사해가 나올거야  sc.fit(train\_scaled, train\_target)  print(sc.score(train\_scaled, train\_target))  print(sc.score(test\_scaled, test\_target))  ## Partial fit  sc.partial\_fit(train\_scaled, train\_target)  print(sc.score(train\_scaled, train\_target))  print(sc.score(test\_scaled, test\_target))  classes = np.unique(train\_target)  classes  sc = SGDClassifier(loss = 'log')  train\_score = []  test\_score = []  for i in range(0, 300):  sc.partial\_fit(train\_scaled, train\_target, classes = classes)  train\_score.append(sc.score(train\_scaled, train\_target))  test\_score.append(sc.score(test\_scaled, test\_target))  plt.plot(train\_score, label = 'Train')  plt.plot(test\_score, label = 'Test')  plt.ylim(0.5, 1.0)  plt.legend()  plt.show() | |