|  |  |
| --- | --- |
| 실습보고서 | |
| 실습02: 모델의 검증 | |
| 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('lin\_regression\_data\_03.csv',names = ['x','y'])  age\_data = np.asarray(raw\_data['x'].values.tolist())  height\_data = np.asarray(raw\_data['y'].values.tolist())  plt.plot(age\_data,height\_data,'ro')  plt.legend(['People Data'])  plt.xlabel('age')  plt.ylabel('height')  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 각각 x축 데이터 행렬, y축 데이터 행렬로 지정한다음 ‘plot’을 사용하여 각 데이터의 위치를 빨간 점(People Data) 그래프로 표현하였습니다.  age\_data는 x축의 나이 데이터를 저장한 배열, height\_data는 y축의 키 데이터를 저장한 배열입니다. |

**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('lin\_regression\_data\_03.csv',names = ['x','y'])  age\_data = np.asarray(raw\_data['x'].values.tolist())  height\_data = np.asarray(raw\_data['y'].values.tolist())  S\_age\_data = np.zeros(20)  S\_height\_data = np.zeros(20)  T\_age\_data = np.zeros(5)  T\_height\_data = np.zeros(5)  for s in range(20):  S\_age\_data[s] = age\_data[s]  S\_height\_data[s] = height\_data[s]    for t in range(5):  T\_age\_data[t] = age\_data[t+20]  T\_height\_data[t] = height\_data[t+20]  plt.plot(S\_age\_data,S\_height\_data,'ro',T\_age\_data,T\_height\_data,'go')  plt.legend(['S\_data','T\_data'])  plt.xlabel('age')  plt.ylabel('height')  plt.grid(True)  plt.show() |

|  |
| --- |
| 그래프 |
|  |

|  |
| --- |
| 설명 (본 실습 과제의 중요 이론 및 결과를 간략히 설명) |
| 훈련 집합 데이터를 넣을 모든 원소의 값이 0인 1행 20열인 배열을 np.zeros를 사용하여 만듭니다.  테스트 집합 또한 모든 원소의 값이 0인 1행 5열의 배열을 np.zeros를 사용하여 만듭니다.  1~20번의 훈련 집합(S)를 ‘for’문을 사용하여 age\_data값을 S\_age\_data에 넣고 height\_data 값을 S\_height\_data에 넣습니다.  21~25번의 테스트 집합(T)를 ‘for’문을 사용하여 age\_data값을 T\_age\_data에 넣고 height\_data 값을 T\_height\_data에 넣습니다.  빨간점(S\_data)은 x축을 S\_age\_data, y축을 S\_height\_data 로 설정한 그래프입니다.  초록점(T\_data)는 x축을 T\_age\_data, y축을 T\_height\_data 로 설정한 그래프입니다. |

**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('lin\_regression\_data\_01.csv',names = ['x','y'])  age\_data = np.asarray(raw\_data['x'].values.tolist())  height\_data = np.asarray(raw\_data['y'].values.tolist())  S\_age\_data = np.zeros(20)  S\_height\_data = np.zeros(20)  for s in range(20):  S\_age\_data[s] = age\_data[s]  S\_height\_data[s] = height\_data[s]  Fake\_GK = np.ones((20,1), int)  K = [6,7,8,9,10,11,12,13]  total\_G\_Par = [0]\*len(K)  for i in range(len(K)):  G\_Par = np.zeros(K[i])  Mk = np.zeros(K[i])  Guass = np.zeros((K[i],20))  Final\_Guass\_y\_hat = np.zeros(20)  dis = (max(S\_age\_data) - min(S\_age\_data))/(K[i]-1)  for k in range(K[i]):  Mk[k] = min(S\_age\_data) + dis\*k  Guass[k] = np.exp(-0.5\*((S\_age\_data-Mk[k])/(dis))\*\*2)  RGuass = np.concatenate([Guass.transpose(), Fake\_GK],1)  G\_Par = (np.linalg.inv(RGuass.transpose().dot(RGuass))).dot(RGuass.transpose()).dot(S\_height\_data.reshape(20,1))  total\_G\_Par[i] = G\_Par  print(total\_G\_Par) |

|  |
| --- |
| 최적해 및 설명 |
| K = 6일 때, 최적해는 차례대로 [-21.21663104], [6.60509189], [-15.17417262], [2.94197992], [-5.86254568],[-4.54210426], [30.84944245] 입니다.  K = 7일 때, 최적해는 차례대로 [0.58885485], [-1.36776413], [9.28927701], [-6.09485374], [15.95402324], [-6.16207823], [13.81140715], [10.61254433] 입니다.  K = 8일 때, 최적해는 차례대로 [-2.53752669], [2.2066432], [0.70434221], [4.89335392], [-2.61639022], [13.43979193], [-6.41463621], [12.54097603], [12.45935633] 입니다.  K = 9일 때, 최적해는 차례대로 [5.64553401], [0.36707437], [8.27052973], [2.78050889], [8.38492098], [1.95477338], [17.09270587], [-5.84120081], [19.71932465], [4.32779877] 입니다.  K = 10일 때, 최적해는 차례대로 [3.27199672], [2.24815767], [3.46222975], [6.58153714], [2.28400589], [7.79644729], [3.49152523], [13.34549728], [-4.23051581], [17.49065481], [6.09634345] 입니다.  K = 11일 때, 최적해는 차례대로 [26.12130346], [-1.28484369], [23.9463878], [5.17610676], [22.6483512], [4.87646772], [26.08170005], [4.1176131], [31.42901364], [-5.46792494], [39.0830706], [-17.46621308]  K = 12일 때, 최적해는 차례대로 [1.95268545e+01], [2.94346193e+00], [1.27611193e+01], [1.41014927e+01], [6.18312390e+00], [2.08302438e+01], [-1.92762038e-02], [3.07629598e+01], [-6.89767483e+00], [3.47548030e+01], [-8.53747582e+00], [3.53075162e+01], [-1.20676465e+01] 입니다.  K = 13일 때, 최적해는 차례대로 [10.83299196], [6.17058528], [2.24830875], [16.89606672], [-0.82604983], [17.08010607], [2.77240107], [12.50132558], [13.57161408], [0.90463839], [23.2927536], [-5.12375155], [26.63531322], [-3.95464478] 입니다.  Fake\_GK는 Wm을 표현하기 위해 필요한 더미 값(항상 1인 값)들을 나타내는 값입니다.  G\_Par은 매개변수의 값들을 나타냅니다. 매개변수는 (RGuass의 전치행렬과 RGuass값의 곱)을 역행렬 해준다음, 그 식을 RGuass 전치행렬과 곱해주고 그 다음 20행 1열인 S\_height\_data의 배열과 곱해주었습니다.  Mk는 k번째 가우스 함수의 평균을 나타냅니다.  Guass는 x를 S\_age\_data로 설정하여 표현한 기저 함수입니다.  RGuass는 기저 함수(Guass)와 더미 변수(Fake\_GK)를 합친 행렬로 매개변수를 구하기 위한 최종 기저함수 행렬을 의미합니다.  total\_G\_Par은 K가 6~13일 때의 모든 매개변수들을 저장하는 공간을 의미합니다. |

**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('lin\_regression\_data\_03.csv',names = ['x','y'])  age\_data = np.asarray(raw\_data['x'].values.tolist())  height\_data = np.asarray(raw\_data['y'].values.tolist())  S\_age\_data = np.zeros(20)  S\_height\_data = np.zeros(20)  T\_age\_data = np.zeros(5)  T\_height\_data = np.zeros(5)  for s in range(20):  S\_age\_data[s] = age\_data[s]  S\_height\_data[s] = height\_data[s]    for t in range(5):  T\_age\_data[t] = age\_data[t+20]  T\_height\_data[t] = height\_data[t+20]  t = np.linspace(min(S\_age\_data),max(S\_age\_data),20)  t1 = np.linspace(min(T\_age\_data),max(T\_age\_data),5)  Fake\_GK = np.ones((20,1), int)  Fake\_GK1 = np.ones((5,1), int)  K = [6,7,8,9,10,11,12,13]  MSE = np.zeros(8)  MSE1 = np.zeros(8)  for b in range(8):  G\_Par = np.zeros(K[b])  Mk = np.zeros(K[b])  Guass = np.zeros((K[b],20))    G\_Par1 = np.zeros(K[b])  Mk1 = np.zeros(K[b])  Guass2 = np.zeros((K[b],5))  dis = (max(S\_age\_data) - min(S\_age\_data))/(K[b]-1)    for k in range(K[b]):  Mk[k] = min(S\_age\_data) + dis\*k  Guass[k] = np.exp(-0.5\*((S\_age\_data-Mk[k])/(dis))\*\*2)    Mk1[k] = min(S\_age\_data) + dis\*k  Guass2[k] = np.exp(-0.5\*((T\_age\_data-Mk1[k])/(dis))\*\*2)  RGuass = np.concatenate([Guass.transpose(), Fake\_GK],1)  RGuass1 = np.concatenate([Guass2.transpose(), Fake\_GK1],1)  G\_Par = (np.linalg.inv(RGuass.transpose().dot(RGuass))).dot(RGuass.transpose()).dot(S\_height\_data.reshape(20,1))  Guass\_y\_hat = (G\_Par)\*(np.concatenate([Guass,Fake\_GK.transpose()]))  MSE[b] = sum((sum(Guass\_y\_hat) - S\_height\_data)\*\*2)/len(S\_height\_data)    Guass\_y\_hat1 = (G\_Par)\*(np.concatenate([Guass2,Fake\_GK1.transpose()]))  MSE1[b] = sum((sum(Guass\_y\_hat1) - T\_height\_data)\*\*2)/len(T\_height\_data)  plt.plot(K,MSE,'r-',K,MSE1,'g-')  plt.legend(['S\_data','T\_data'])  plt.xlabel('K')  plt.ylabel('MSE')  plt.grid(True)  plt.show() |

|  |
| --- |
| 그래프 |
|  |

|  |
| --- |
| 설명 (본 실습 과제의 중요 이론 및 결과를 간략히 설명) |
| 실습 3에서 구한 최적해를 통해 훈련데이터의 선형 기저함수 모델(Guass\_y\_hat)과 검증데이터의 선형 기저함수 모델(Guass\_y\_hat1)을 만들었습니다.  MSE는 훈련데이터의 평균제곱오차로 Guass\_y\_hat과 S\_height\_data의 차를 제곱한 값들의 평균으로 구하였습니다.  MSE1는 검증데이터의 평균제곱오차로 Guass\_y\_hat1과 T\_height\_data의 차를 제곱한 값들의 평균으로 구하였습니다.  빨간선(S\_data)은 x축을 K, y축을 MSE로 설정한 그래프입니다.  초록선(T\_data)는 x축을 K, y축을 MSE1로 설정한 그래프입니다.  위의 결과를 통해 K값이 증가하면서 어느 부분을 넘어가면 검증성능(T\_data)은 더 이상 개선되지 않고 오히려 악화되는 모습을 볼 수 있습니다. 이러한 현상을 ‘과적합’이라고 합니다. |

**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('lin\_regression\_data\_03.csv',names = ['x','y'])  age\_data = np.asarray(raw\_data['x'].values.tolist())  height\_data = np.asarray(raw\_data['y'].values.tolist())  Cross\_age\_data = np.zeros(5)  Cross\_height\_data = np.zeros(5)  Cross\_age\_data1 = np.zeros(5)  Cross\_height\_data1 = np.zeros(5)  Cross\_age\_data2 = np.zeros(5)  Cross\_height\_data2 = np.zeros(5)  Cross\_age\_data3= np.zeros(5)  Cross\_height\_data3 = np.zeros(5)  Cross\_age\_data4 = np.zeros(5)  Cross\_height\_data4 = np.zeros(5)  Cross\_age\_data = age\_data[0:5]  Cross\_height\_data = height\_data[0:5]  Cross\_age\_data1 = age\_data[5:10]  Cross\_height\_data1 = height\_data[5:10]  Cross\_age\_data2 = age\_data[10:15]  Cross\_height\_data2 = height\_data[10:15]  Cross\_age\_data3= age\_data[15:20]  Cross\_height\_data3 = height\_data[15:20]  Cross\_age\_data4 = age\_data[20:25]  Cross\_height\_data4 = height\_data[20:25]  plt.plot(Cross\_age\_data,Cross\_height\_data,'ro',Cross\_age\_data1,Cross\_height\_data1,'go',Cross\_age\_data2,Cross\_height\_data2,'bo',Cross\_age\_data3,Cross\_height\_data3,'ko',Cross\_age\_data4,Cross\_height\_data4,'mo')  plt.legend(['data 0','data 1','data 2','data 3','data 4'])  plt.xlabel('age')  plt.ylabel('height')  plt.grid(True)  plt.show() |

|  |
| --- |
| 그래프 |
|  |

|  |
| --- |
| 설명 (본 실습 과제의 중요 이론 및 결과를 간략히 설명) |
| np.zeros(5)를 사용하여 모든 원소가 ‘0’ 으로 구성된 1행 5열의 배열을 10개를 만듭니다.(Cross\_age\_data(0~4) 5개 + Cross\_height\_data(0~4) 5개)  슬라이싱을 통해 age\_data와 height\_data를 차례대로 5등분 하여 Cross\_age\_data(0~4)와 Cross\_height\_data(0~4)에 차례대로 넣습니다.  빨간점(data 0)은 x축을 Cross\_age\_data, y축을 Cross\_height\_data로 설정한 그래프입니다.  초록점(data 1)는 x축을 Cross\_age\_data1, y축을 Cross\_height\_data1로 설정한 그래프입니다.  파란점(data 2)은 x축을 Cross\_age\_data2, y축을 Cross\_height\_data2로 설정한 그래프입니다.  검은점(data 3)는 x축을 Cross\_age\_data3, y축을 Cross\_height\_data3로 설정한 그래프입니다.  자홍점(data 4)은 x축을 Cross\_age\_data4, y축을 Cross\_height\_data4로 설정한 그래프입니다. |

**실습 06**

|  |
| --- |
| 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('lin\_regression\_data\_03.csv',names = ['x','y'])  age\_data = np.asarray(raw\_data['x'].values.tolist())  height\_data = np.asarray(raw\_data['y'].values.tolist())  Cross\_age\_data = np.zeros(5)  Cross\_height\_data = np.zeros(5)  Cross\_age\_data1 = np.zeros(5)  Cross\_height\_data1 = np.zeros(5)  Cross\_age\_data2 = np.zeros(5)  Cross\_height\_data2 = np.zeros(5)  Cross\_age\_data3= np.zeros(5)  Cross\_height\_data3 = np.zeros(5)  Cross\_age\_data4 = np.zeros(5)  Cross\_height\_data4 = np.zeros(5)  Cross\_age\_data = age\_data[0:5]  Cross\_height\_data = height\_data[0:5]  Cross\_age\_data1 = age\_data[5:10]  Cross\_height\_data1 = height\_data[5:10]  Cross\_age\_data2 = age\_data[10:15]  Cross\_height\_data2 = height\_data[10:15]  Cross\_age\_data3= age\_data[15:20]  Cross\_height\_data3 = height\_data[15:20]  Cross\_age\_data4 = age\_data[20:25]  Cross\_height\_data4 = height\_data[20:25]  T\_Cross\_age\_data = np.concatenate([Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data3,Cross\_age\_data4])  T\_Cross\_age\_data1 = np.concatenate([Cross\_age\_data,Cross\_age\_data2,Cross\_age\_data3,Cross\_age\_data4])  T\_Cross\_age\_data2 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data3,Cross\_age\_data4])  T\_Cross\_age\_data3 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data4])  T\_Cross\_age\_data4 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data3])  T\_Cross\_age\_data5 = np.concatenate([T\_Cross\_age\_data,T\_Cross\_age\_data1,T\_Cross\_age\_data2,T\_Cross\_age\_data3,T\_Cross\_age\_data4])  T\_Cross\_height\_data = np.concatenate([Cross\_height\_data1,Cross\_height\_data2,Cross\_height\_data3,Cross\_height\_data4])  T\_Cross\_height\_data1 = np.concatenate([Cross\_height\_data,Cross\_height\_data2,Cross\_height\_data3,Cross\_height\_data4])  T\_Cross\_height\_data2 = np.concatenate([Cross\_height\_data,Cross\_height\_data1,Cross\_height\_data3,Cross\_height\_data4])  T\_Cross\_height\_data3 = np.concatenate([Cross\_height\_data,Cross\_height\_data1,Cross\_height\_data2,Cross\_height\_data4])  T\_Cross\_height\_data4 = np.concatenate([Cross\_height\_data,Cross\_height\_data1,Cross\_height\_data2,Cross\_height\_data3])  T\_Cross\_height\_data5 = np.concatenate([T\_Cross\_height\_data,T\_Cross\_height\_data1,T\_Cross\_height\_data2,T\_Cross\_height\_data3,T\_Cross\_height\_data4])  S\_Cross\_age\_data = Cross\_age\_data  S\_Cross\_age\_data1 = Cross\_age\_data1  S\_Cross\_age\_data2 = Cross\_age\_data2  S\_Cross\_age\_data3 = Cross\_age\_data3  S\_Cross\_age\_data4 = Cross\_age\_data4  S\_Cross\_age\_data5 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data3,Cross\_age\_data4])  S\_Cross\_height\_data = Cross\_height\_data  S\_Cross\_height\_data1 = Cross\_height\_data1  S\_Cross\_height\_data2 = Cross\_height\_data2  S\_Cross\_height\_data3 = Cross\_height\_data3  S\_Cross\_height\_data4 = Cross\_height\_data4  S\_Cross\_height\_data5 = np.concatenate([S\_Cross\_height\_data,S\_Cross\_height\_data1,S\_Cross\_height\_data2,S\_Cross\_height\_data3,S\_Cross\_height\_data4])  Total\_MSE = np.zeros(5)  Total\_G\_Par = np.zeros((5,10,1))  for i in range(5):  Fake\_GK = np.ones((20,1), int)  Fake\_GK1 = np.ones((5,1), int)  MSE = np.zeros(1)  MSE1 = np.zeros(1)  K = [9]  for b in range(1):  G\_Par = np.zeros(K[b])  Mk = np.zeros(K[b])  Guass = np.zeros((K[b],20))    G\_Par1 = np.zeros(K[b])  Mk1 = np.zeros(K[b])  Guass2 = np.zeros((K[b],5))  dis = (max(T\_Cross\_age\_data5[i\*20:(i+1)\*20]) - min(T\_Cross\_age\_data5[i\*20:(i+1)\*20]))/(K[b]-1)    for k in range(K[b]):  Mk[k] = min(T\_Cross\_age\_data5[i\*20:(i+1)\*20]) + dis\*k  Guass[k] = np.exp(-0.5\*((T\_Cross\_age\_data5[i\*20:(i+1)\*20]-Mk[k])/(dis))\*\*2)    Mk1[k] = min(T\_Cross\_age\_data5[i\*20:(i+1)\*20]) + dis\*k  Guass2[k] = np.exp(-0.5\*((S\_Cross\_age\_data5[i\*5:(i+1)\*5]-Mk1[k])/(dis))\*\*2)  RGuass = np.concatenate([Guass.transpose(), Fake\_GK],1)  RGuass1 = np.concatenate([Guass2.transpose(), Fake\_GK1],1)  G\_Par = (np.linalg.inv(RGuass.transpose().dot(RGuass))).dot(RGuass.transpose()).dot(T\_Cross\_height\_data5[i\*20:(i+1)\*20].reshape(20,1))  Guass\_y\_hat = (G\_Par)\*(np.concatenate([Guass,Fake\_GK.transpose()]))  MSE[b] = sum((sum(Guass\_y\_hat) - T\_Cross\_height\_data5[i\*20:(i+1)\*20])\*\*2)/len(T\_Cross\_height\_data5[i\*20:(i+1)\*20])    Guass\_y\_hat1 = (G\_Par)\*(np.concatenate([Guass2,Fake\_GK1.transpose()]))  MSE1[b] = sum((sum(Guass\_y\_hat1) - S\_Cross\_height\_data5[i\*5:(i+1)\*5])\*\*2)/len(S\_Cross\_height\_data5[i\*5:(i+1)\*5])    Total\_MSE[i] = MSE1[b]-MSE[b]  Total\_G\_Par[i] = G\_Par  print(Total\_MSE)  print(Total\_G\_Par) |

|  |
| --- |
| 매개변수, 일반화오차 및 설명 |
| T\_Cross\_age\_data(0~4)까지는 5겹 교차검증에 필요한 훈련집합이며 T\_Cross\_age\_data5는 T\_Cross\_age\_data(0~4)까지의 모든 T\_Cross\_age\_data를 넣은 최종 배열입니다.(훈련 집합의 나이 데이터)  T\_Cross\_height\_data(0~4)과 T\_Cross\_height\_data5 도 위의 설명과 동일 합니다.(훈련집합의 키 데이터)  S\_Cross\_age\_data(0~4)까지는 5겹 교차검증에 필요한 검증집합이며, S\_Cross\_age\_data5는 S\_Cross\_age\_data(0~4)까지의 모든 S\_Cross\_age\_data를 넣은 최종 배열입니다.(검증 집합의 나이 데이터)  S\_Cross\_height\_data(0~4)과 S\_Cross\_height\_data5 도 위의 설명과 동일 합니다.(검증집합의 키 데이터)  회귀 시스템의 경우 예측 성공과 실패를 정수로 표현할 수 없으므로, 평균제곱오차를 이용하여 성능을 측정합니다.  일반화 성능 = ‘검증 집합 데이터에 대해 측정한 평균제곱오차’ 입니다.  훈련 성능 = ‘훈련 집합 데이터에 대해 측정한 평균 제곱오차’ 입니다.  일반화 오차 = |일반화 성능 – 훈련 성능| 입니다.  평균제곱오차는 저번 실습시간 가우시안 함수를 이용하여 구하였습니다. (MSE = (y\_hat – y)^2의 평균)  학습과 검증 1의 매개변수는 [-23.27773784], [8.46036298], [-22.82681943], [5.92008981], [-13.70476057], [-1.92170106], [-4.4006918], [-4.09937563], [-6.4250132], [33.96748624] 입니다. 일반화 오차는 [1.62973769e+00] 입니다.  학습과 검증 2의 매개변수는 [-8.9599264], [6.05043946], [-9.33234678], [5.83480925], [-3.33087033], [-0.31618698], [4.52856969], [-1.88903723], [5.00044914], [19.22266449] 입니다. 일반화 오차는 [4.59959430e-01] 입니다.  학습과 검증 3의 매개변수는 [-42.45075814], [4.27465609], [-36.68523632], [1.22808137], [-30.16935684], [-1.69319305], [-27.44140133], [2.73127103], [-31.63690077], [57.87225316] 입니다. 일반화 오차는 [1.11251389e+00] 입니다.  학습과 검증 4의 매개변수는 [38.594979], [-10.0175891], [41.86303039], [-11.20861564], [46.91526877], [-10.57723163], [46.04739466], [-9.52523718], [46.23936352], [-23.04483409] 입니다. 일반화 오차는 [1.53298382e+02] 입니다.  학습과 검증 5의 매개변수는 [-21.89483269], [7.31964336], [-21.0321158], [5.87378214], [-14.76465843], [1.47883791], [-8.65651539], [0.10144946], [-7.92520385], [33.05474] 입니다. 일반화 오차는 [2.58378697e-02] 입니다. |

**실습 07**

|  |
| --- |
| 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('lin\_regression\_data\_03.csv',names = ['x','y'])  age\_data = np.asarray(raw\_data['x'].values.tolist())  height\_data = np.asarray(raw\_data['y'].values.tolist())  Cross\_age\_data = np.zeros(5)  Cross\_height\_data = np.zeros(5)  Cross\_age\_data1 = np.zeros(5)  Cross\_height\_data1 = np.zeros(5)  Cross\_age\_data2 = np.zeros(5)  Cross\_height\_data2 = np.zeros(5)  Cross\_age\_data3= np.zeros(5)  Cross\_height\_data3 = np.zeros(5)  Cross\_age\_data4 = np.zeros(5)  Cross\_height\_data4 = np.zeros(5)  Cross\_age\_data = age\_data[0:5]  Cross\_height\_data = height\_data[0:5]  Cross\_age\_data1 = age\_data[5:10]  Cross\_height\_data1 = height\_data[5:10]  Cross\_age\_data2 = age\_data[10:15]  Cross\_height\_data2 = height\_data[10:15]  Cross\_age\_data3= age\_data[15:20]  Cross\_height\_data3 = height\_data[15:20]  Cross\_age\_data4 = age\_data[20:25]  Cross\_height\_data4 = height\_data[20:25]  T\_Cross\_age\_data = np.concatenate([Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data3,Cross\_age\_data4])  T\_Cross\_age\_data1 = np.concatenate([Cross\_age\_data,Cross\_age\_data2,Cross\_age\_data3,Cross\_age\_data4])  T\_Cross\_age\_data2 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data3,Cross\_age\_data4])  T\_Cross\_age\_data3 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data4])  T\_Cross\_age\_data4 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data3])  T\_Cross\_age\_data5 = np.concatenate([T\_Cross\_age\_data,T\_Cross\_age\_data1,T\_Cross\_age\_data2,T\_Cross\_age\_data3,T\_Cross\_age\_data4])  T\_Cross\_height\_data = np.concatenate([Cross\_height\_data1,Cross\_height\_data2,Cross\_height\_data3,Cross\_height\_data4])  T\_Cross\_height\_data1 = np.concatenate([Cross\_height\_data,Cross\_height\_data2,Cross\_height\_data3,Cross\_height\_data4])  T\_Cross\_height\_data2 = np.concatenate([Cross\_height\_data,Cross\_height\_data1,Cross\_height\_data3,Cross\_height\_data4])  T\_Cross\_height\_data3 = np.concatenate([Cross\_height\_data,Cross\_height\_data1,Cross\_height\_data2,Cross\_height\_data4])  T\_Cross\_height\_data4 = np.concatenate([Cross\_height\_data,Cross\_height\_data1,Cross\_height\_data2,Cross\_height\_data3])  T\_Cross\_height\_data5 = np.concatenate([T\_Cross\_height\_data,T\_Cross\_height\_data1,T\_Cross\_height\_data2,T\_Cross\_height\_data3,T\_Cross\_height\_data4])  S\_Cross\_age\_data = Cross\_age\_data  S\_Cross\_age\_data1 = Cross\_age\_data1  S\_Cross\_age\_data2 = Cross\_age\_data2  S\_Cross\_age\_data3 = Cross\_age\_data3  S\_Cross\_age\_data4 = Cross\_age\_data4  S\_Cross\_age\_data5 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data3,Cross\_age\_data4])  S\_Cross\_height\_data = Cross\_height\_data  S\_Cross\_height\_data1 = Cross\_height\_data1  S\_Cross\_height\_data2 = Cross\_height\_data2  S\_Cross\_height\_data3 = Cross\_height\_data3  S\_Cross\_height\_data4 = Cross\_height\_data4  S\_Cross\_height\_data5 = np.concatenate([S\_Cross\_height\_data,S\_Cross\_height\_data1,S\_Cross\_height\_data2,S\_Cross\_height\_data3,S\_Cross\_height\_data4])  Total\_MSE = np.zeros(5)  Total\_G\_Par = np.zeros((5,10,1))  Total\_Guass\_y\_hat = np.zeros((5,10,20))  t = np.linspace(min(age\_data),max(age\_data),20)  t1 = np.linspace(min(age\_data),max(age\_data),5)  for i in range(5):  Fake\_GK = np.ones((20,1), int)  Fake\_GK1 = np.ones((5,1), int)  K = [9]  MSE = np.zeros(1)  MSE1 = np.zeros(1)  for b in range(1):  G\_Par = np.zeros(K[b])  Mk = np.zeros(K[b])  Guass = np.zeros((K[b],20))  Guass1 = np.zeros((K[b],20))    G\_Par1 = np.zeros(K[b])  Mk1 = np.zeros(K[b])  Guass2 = np.zeros((K[b],5))  Guass3 = np.zeros((K[b],5))  dis = (max(T\_Cross\_age\_data5[i\*20:(i+1)\*20]) - min(T\_Cross\_age\_data5[i\*20:(i+1)\*20]))/(K[b]-1)  for k in range(K[b]):  Mk[k] = min(T\_Cross\_age\_data5[i\*20:(i+1)\*20]) + dis\*k  Guass[k] = np.exp(-0.5\*((T\_Cross\_age\_data5[i\*20:(i+1)\*20]-Mk[k])/(dis))\*\*2)  Guass1[k] = np.exp(-0.5\*((t-Mk[k])/(dis))\*\*2)    Mk1[k] = min(T\_Cross\_age\_data5[i\*20:(i+1)\*20]) + dis\*k  Guass2[k] = np.exp(-0.5\*((S\_Cross\_age\_data5[i\*5:(i+1)\*5]-Mk1[k])/(dis))\*\*2)  Guass3[k] = np.exp(-0.5\*((t1-Mk1[k])/(dis))\*\*2)  RGuass = np.concatenate([Guass.transpose(), Fake\_GK],1)  RGuass1 = np.concatenate([Guass2.transpose(), Fake\_GK1],1)  G\_Par = (np.linalg.inv(RGuass.transpose().dot(RGuass))).dot(RGuass.transpose()).dot(T\_Cross\_height\_data5[i\*20:(i+1)\*20].reshape(20,1))  Guass\_y\_hat = (G\_Par)\*(np.concatenate([Guass1,Fake\_GK.transpose()]))  MSE[b] = sum((sum(Guass\_y\_hat) - T\_Cross\_height\_data5[i\*20:(i+1)\*20])\*\*2)/len(T\_Cross\_height\_data5[i\*20:(i+1)\*20])    Guass\_y\_hat1 = (G\_Par)\*(np.concatenate([Guass2,Fake\_GK1.transpose()]))  MSE1[b] = sum((sum(Guass\_y\_hat1) - S\_Cross\_height\_data5[i\*5:(i+1)\*5])\*\*2)/len(S\_Cross\_height\_data5[i\*5:(i+1)\*5])    Total\_Guass\_y\_hat[i] = Guass\_y\_hat  Total\_MSE[i] = MSE1[b]-MSE[b]  Total\_G\_Par[i] = G\_Par  plt.figure(1)  plt.plot(T\_Cross\_age\_data,T\_Cross\_height\_data,'ro',S\_Cross\_age\_data,S\_Cross\_height\_data,'ko',t,sum(Total\_Guass\_y\_hat[0]),'b-')  plt.legend(['Traning data','Validation data','Linear Basis Function Model'])  plt.xlabel('age')  plt.ylabel('height')  plt.grid(True)  plt.show()  plt.figure(2)  plt.plot(T\_Cross\_age\_data1,T\_Cross\_height\_data1,'ro',S\_Cross\_age\_data1,S\_Cross\_height\_data1,'ko',t,sum(Total\_Guass\_y\_hat[1]),'b-')  plt.legend(['Traning data','Validation data','Linear Basis Function Model'])  plt.xlabel('age')  plt.ylabel('height')  plt.grid(True)  plt.show()  plt.figure(3)  plt.plot(T\_Cross\_age\_data2,T\_Cross\_height\_data2,'ro',S\_Cross\_age\_data2,S\_Cross\_height\_data2,'ko',t,sum(Total\_Guass\_y\_hat[2]),'b-')  plt.legend(['Traning data','Validation data','Linear Basis Function Model'])  plt.xlabel('age')  plt.ylabel('height')  plt.grid(True)  plt.show()  plt.figure(4)  plt.plot(T\_Cross\_age\_data3,T\_Cross\_height\_data3,'ro',S\_Cross\_age\_data3,S\_Cross\_height\_data3,'ko',t,sum(Total\_Guass\_y\_hat[3]),'b-')  plt.legend(['Traning data','Validation data','Linear Basis Function Model'])  plt.xlabel('age')  plt.ylabel('height')  plt.grid(True)  plt.show()  plt.figure(5)  plt.plot(T\_Cross\_age\_data4,T\_Cross\_height\_data4,'ro',S\_Cross\_age\_data4,S\_Cross\_height\_data4,'ko',t,sum(Total\_Guass\_y\_hat[4]),'b-')  plt.legend(['Traning data','Validation data','Linear Basis Function Model'])  plt.xlabel('age')  plt.ylabel('height')  plt.grid(True)  plt.show() |

|  |
| --- |
| 그래프 |
|  |

|  |
| --- |
| 설명 (본 실습 과제의 중요 이론 및 결과를 간략히 설명) |
| 앞선 실습에서 만든 훈련데이터의 선형 기저함수 모델(Guass\_y\_hat)의 x값엔 ‘t = age\_data값의 최소 ~ age\_data값의 최대를 균일하게 20개로 나눈 값’을 검증데이터의 선형 기저함수 모델(Guass\_y\_hat1)의 x값엔 ‘t1 = age\_data값의 최소 ~ age\_data값의 최대를 균일하게 5개로 나눈 값’을 대입하여 최종 선형 기저함수 모델을 만들었습니다.  빨간점(Training data)은 x축을 T\_Cross\_age\_data(0~4), y축을 T\_Cross\_height\_data(0~4)로 설정한 그래프입니다.  검은점(Validation data)는 x축을 S\_Cross\_age\_data(0~4), y축을 S\_Cross\_height\_data(0~4)로 설정한 그래프입니다.  파란선(Linear Basis Function Model)은 x축을 t, y축을 선형 기저함수 회귀 모델(sum(Total\_Guass\_y\_hat[0~4]))로 설정한 그래프입니다. |

**실습 08**

|  |
| --- |
| 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('lin\_regression\_data\_03.csv',names = ['x','y'])  age\_data = np.asarray(raw\_data['x'].values.tolist())  height\_data = np.asarray(raw\_data['y'].values.tolist())  Cross\_age\_data = np.zeros(5)  Cross\_height\_data = np.zeros(5)  Cross\_age\_data1 = np.zeros(5)  Cross\_height\_data1 = np.zeros(5)  Cross\_age\_data2 = np.zeros(5)  Cross\_height\_data2 = np.zeros(5)  Cross\_age\_data3= np.zeros(5)  Cross\_height\_data3 = np.zeros(5)  Cross\_age\_data4 = np.zeros(5)  Cross\_height\_data4 = np.zeros(5)  Cross\_age\_data = age\_data[0:5]  Cross\_height\_data = height\_data[0:5]  Cross\_age\_data1 = age\_data[5:10]  Cross\_height\_data1 = height\_data[5:10]  Cross\_age\_data2 = age\_data[10:15]  Cross\_height\_data2 = height\_data[10:15]  Cross\_age\_data3= age\_data[15:20]  Cross\_height\_data3 = height\_data[15:20]  Cross\_age\_data4 = age\_data[20:25]  Cross\_height\_data4 = height\_data[20:25]  T\_Cross\_age\_data = np.concatenate([Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data3,Cross\_age\_data4])  T\_Cross\_age\_data1 = np.concatenate([Cross\_age\_data,Cross\_age\_data2,Cross\_age\_data3,Cross\_age\_data4])  T\_Cross\_age\_data2 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data3,Cross\_age\_data4])  T\_Cross\_age\_data3 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data4])  T\_Cross\_age\_data4 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data3])  T\_Cross\_age\_data5 = np.concatenate([T\_Cross\_age\_data,T\_Cross\_age\_data1,T\_Cross\_age\_data2,T\_Cross\_age\_data3,T\_Cross\_age\_data4])  T\_Cross\_height\_data = np.concatenate([Cross\_height\_data1,Cross\_height\_data2,Cross\_height\_data3,Cross\_height\_data4])  T\_Cross\_height\_data1 = np.concatenate([Cross\_height\_data,Cross\_height\_data2,Cross\_height\_data3,Cross\_height\_data4])  T\_Cross\_height\_data2 = np.concatenate([Cross\_height\_data,Cross\_height\_data1,Cross\_height\_data3,Cross\_height\_data4])  T\_Cross\_height\_data3 = np.concatenate([Cross\_height\_data,Cross\_height\_data1,Cross\_height\_data2,Cross\_height\_data4])  T\_Cross\_height\_data4 = np.concatenate([Cross\_height\_data,Cross\_height\_data1,Cross\_height\_data2,Cross\_height\_data3])  T\_Cross\_height\_data5 = np.concatenate([T\_Cross\_height\_data,T\_Cross\_height\_data1,T\_Cross\_height\_data2,T\_Cross\_height\_data3,T\_Cross\_height\_data4])  S\_Cross\_age\_data = Cross\_age\_data  S\_Cross\_age\_data1 = Cross\_age\_data1  S\_Cross\_age\_data2 = Cross\_age\_data2  S\_Cross\_age\_data3 = Cross\_age\_data3  S\_Cross\_age\_data4 = Cross\_age\_data4  S\_Cross\_age\_data5 = np.concatenate([Cross\_age\_data,Cross\_age\_data1,Cross\_age\_data2,Cross\_age\_data3,Cross\_age\_data4])  S\_Cross\_height\_data = Cross\_height\_data  S\_Cross\_height\_data1 = Cross\_height\_data1  S\_Cross\_height\_data2 = Cross\_height\_data2  S\_Cross\_height\_data3 = Cross\_height\_data3  S\_Cross\_height\_data4 = Cross\_height\_data4  S\_Cross\_height\_data5 = np.concatenate([S\_Cross\_height\_data,S\_Cross\_height\_data1,S\_Cross\_height\_data2,S\_Cross\_height\_data3,S\_Cross\_height\_data4])  Total\_MSE = np.zeros(5)  Total\_G\_Par = np.zeros((5,10,1))  for i in range(5):  Fake\_GK = np.ones((20,1), int)  Fake\_GK1 = np.ones((5,1), int)  K = [9]  MSE = np.zeros(1)  MSE1 = np.zeros(1)  for b in range(1):  G\_Par = np.zeros(K[b])  Mk = np.zeros(K[b])  Guass = np.zeros((K[b],20))    G\_Par1 = np.zeros(K[b])  Mk1 = np.zeros(K[b])  Guass2 = np.zeros((K[b],5))  dis = (max(T\_Cross\_age\_data5[i\*20:(i+1)\*20]) - min(T\_Cross\_age\_data5[i\*20:(i+1)\*20]))/(K[b]-1)  for k in range(K[b]):  Mk[k] = min(T\_Cross\_age\_data5[i\*20:(i+1)\*20]) + dis\*k  Guass[k] = np.exp(-0.5\*((T\_Cross\_age\_data5[i\*20:(i+1)\*20]-Mk[k])/(dis))\*\*2)    Mk1[k] = min(T\_Cross\_age\_data5[i\*20:(i+1)\*20]) + dis\*k  Guass2[k] = np.exp(-0.5\*((S\_Cross\_age\_data5[i\*5:(i+1)\*5]-Mk1[k])/(dis))\*\*2)  RGuass = np.concatenate([Guass.transpose(), Fake\_GK],1)  RGuass1 = np.concatenate([Guass2.transpose(), Fake\_GK1],1)  G\_Par = (np.linalg.inv(RGuass.transpose().dot(RGuass))).dot(RGuass.transpose()).dot(T\_Cross\_height\_data5[i\*20:(i+1)\*20].reshape(20,1))  Guass\_y\_hat = (G\_Par)\*(np.concatenate([Guass,Fake\_GK.transpose()]))  MSE[b] = sum((sum(Guass\_y\_hat) - T\_Cross\_height\_data5[i\*20:(i+1)\*20])\*\*2)/len(T\_Cross\_height\_data5[i\*20:(i+1)\*20])    Guass\_y\_hat1 = (G\_Par)\*(np.concatenate([Guass2,Fake\_GK1.transpose()]))  MSE1[b] = sum((sum(Guass\_y\_hat1) - S\_Cross\_height\_data5[i\*5:(i+1)\*5])\*\*2)/len(S\_Cross\_height\_data5[i\*5:(i+1)\*5])    Total\_MSE[i] = MSE1[b]-MSE[b]  Total\_G\_Par[i] = G\_Par  # 최종 일반화 오차 구하는 식  print(np.mean(Total\_MSE))  # 최종 매개변수 평균 구하는 식  Average\_G\_Par = np.zeros(10)  for i in range(10):  Average\_G\_Par[i] = np.mean(Total\_G\_Par[:,i])  print(Average\_G\_Par) |

|  |
| --- |
| 매개변수, 일반화오차 및 설명 |
| ‘for’문을 사용하여 5개의 G\_par 배열 값을 Total\_G\_Par[i]에 넣고, 5개의 일반화 오차값(MSE1-MSE) 값을 Total\_MSE[i]에 넣었습니다.  최종 매개변수는 모든 매개변수들의 평균을 통해서 구했습니다.(np.mean 사용)  최종 일반화 오차는 모든 일반화 오차들의 평균을 통해서 구했습니다.(np.mean 사용)  최종 매개변수는 [-11.59765521], [3.21750256], [-9.60269759], [1.52962939], [-3.01087548], [-2.60589496], [2.01547116], [-2.53618591], [1.05053897], [24.21446196] 입니다.  최종 일반화 오차는 [31.305286240684342] 입니다.  실습 3에서  K = 9일 때, 최적해는 차례대로 [5.64553401], [0.36707437], [8.27052973], [2.78050889], [8.38492098], [1.95477338], [17.09270587], [-5.84120081], [19.71932465], [4.32779877] 입니다.  일반화 오차는 [0.02583786969777374]입니다.  5겹 교차검증과 가우스 함수를 이용한 선형 기저함수 모델의 결과는 이론상으론 비슷해야 하지만, k겹 교차검증을 할 때, 데이터 집합, 훈련 집합, 검증 집합이 동일한 통계적 특성을 갖고 데이터 집합을 분리하여야 제대로된 결과가 나온다는 사실을 알 수 있습니다. 즉, 홀드아웃의 성능은 데이터 집합을 어떻게 분리 하느냐에 많은 영향을 받습니다. |