|  |  |
| --- | --- |
| 실습보고서 | |
| 실습04: 인공신경망 | |
| 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('NN\_2d\_data.csv',names = ['x','y','z'])  middle = np.asarray(raw\_data['x'].values.tolist())  final = np.asarray(raw\_data['y'].values.tolist())  grade = np.asarray(raw\_data['z'].values.tolist())  a = 0  b = 0  c = 0  A\_grade\_middle = np.zeros(112)  A\_grade\_final = np.zeros(112)  B\_grade\_middle = np.zeros(263)  B\_grade\_final = np.zeros(263)  C\_grade\_middle = np.zeros(125)  C\_grade\_final = np.zeros(125)  for i in range(500):  if grade[i] == 'A':  A\_grade\_middle[a] = middle[i]  A\_grade\_final[a] = final[i]  a = a+1  elif grade[i] == 'B':  B\_grade\_middle[b] = middle[i]  B\_grade\_final[b] = final[i]  b = b+1  else:  C\_grade\_middle[c] = middle[i]  C\_grade\_final[c] = final[i]  c = c+1  plt.plot(A\_grade\_middle,A\_grade\_final,'ro',B\_grade\_middle,B\_grade\_final,'bx',C\_grade\_middle,C\_grade\_final,'ks')  plt.legend(['A','B','C'])  plt.xlabel('MIDDLE')  plt.ylabel('FINAL')  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(중간고사)와 두번째 열 데이터(기말고사)와 세번째 열 데이터(학점)을 ‘np.panda’ 를 사용하여 불러오고, 중간고사와 기말고사와 학점을 ‘np.asarray’를 사용하여 차례로 x데이터(middle), y데이터(final), z데이터(grade)로 지정하였습니다.  z데이터(grade)의 ‘A’, ‘B’, ‘C’ 결과에 따라 입력값(중간고사 + 기말고사)를 나누기 위해 ‘for’문 안에 조건문을 사용하여 ‘A’일때의 중간고사(middle) 값을 A\_grade\_middle, 기말고사(final) 값을 A\_grade\_final에 넣고 ‘B’일때의 중간고사(middle) 값을 B\_grade\_middle, 기말고사(final) 값을 B\_grade\_final에 넣고 ‘C’일때의 중간고사(middle) 값을 C\_grade\_middle, 기말고사(final) 값을 C\_grade\_final에 넣습니다.  ‘plot’을 사용하여 각 데이터의 위치를 빨간 o(A학점), 파란 x(B학점), 검정 ㅁ(C학점)으로 그래프를 표현하였습니다. |

**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('NN\_2d\_data.csv',names = ['x','y','z'])  middle = np.asarray(raw\_data['x'].values.tolist())  final = np.asarray(raw\_data['y'].values.tolist())  grade = np.asarray(raw\_data['z'].values.tolist())  # One-hot 인코딩 입력 데이터  x = np.stack([middle,final]).T  # One-hot 인코딩 출력 데이터  y0 = np.zeros(500)  y1 = np.zeros(500)  y2 = np.zeros(500)  a = 0  for i in range(500):  if grade[i] == 'A':  y0[a] = 1  a = a+1  elif grade[i] == 'B':  y1[a] = 1  a = a+1  else:  y2[a] = 1  a = a+1  y = np.stack([y0,y1,y2]).T |

|  |
| --- |
| 설명 (본 실습 과제의 중요 이론 및 결과를 간략히 설명) |
| One-hot 인코딩 입력 데이터는 middle과 final을 ‘np.stack’과 Transpose를 이용하여 [500,2]의 입력 데이터를 생성하였습니다.  One-hot 인코딩 출력 데이터는 ‘for’문과 ‘조건문’을 사용하여 학점 ‘A’의 index와 동일한 y0의 index에 1을 저장하고, 학점 ‘B’의 index와 동일한 y1의 index에 1을 저장하고 학점 ‘C’의 index와 동일한 y2의 indexx에 1을 저장하고, ‘np.stack’과 Transpose를 이용하여 [500,3]의 출력 데이터를 생성하였습니다. |

**실습 03**

|  |
| --- |
| 인공신경망 구조, 평균제곱오차 및 설명 |
| <인공신경망 구조>      <평균제곱오차>    평균제곱오차는 n번째 데이터의 평균 제곱오차는 Q개의 출력의 평균제곱오차의 합입니다. 그리고 총 N개의 훈련데이터에 대한 평균제곱오차는 N개의 평균제곱오차의 평균입니다. |

**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('NN\_2d\_data.csv',names = ['x','y','z'])  middle = np.asarray(raw\_data['x'].values.tolist())  final = np.asarray(raw\_data['y'].values.tolist())  grade = np.asarray(raw\_data['z'].values.tolist())  y0 = np.zeros(500)  y1 = np.zeros(500)  y2 = np.zeros(500)  a = 0  for i in range(500):  if grade[i] == 'A':  y0[a] = 1  a = a+1  elif grade[i] == 'B':  y1[a] = 1  a = a+1  else:  y2[a] = 1  a = a+1  y\_qn = np.stack([y0,y1,y2])  v\_total = np.array([[-1.12, -0.05, 0.09],[0.81, -0.10, 0.15],[0.49, 9.84, -7.96]])  w\_total = np.array([[-0.40, 0.02, 0.32],[-6.13, 3.58, 4.24],[1.15, 4.80, -6.16],[0.94, -5.93, -1.11]])  x2 = np.ones(500)  x\_mn = np.stack([middle,final,x2])  LR = 0.0000001  epoch = 100  b = np.zeros([4,500])  b\_dummy = np.ones(1)  v\_diff = np.zeros([3,3])  for z in range(epoch):  for n in range(500):  alpha = np.dot(np.array(v\_total).T,x\_mn[:,n])  f\_a = 1 / (1 + np.exp(-alpha))  b = np.hstack([f\_a,b\_dummy])    beta = np.dot(np.array(w\_total).T, b)  y\_hat = 1 / (1 + np.exp(-beta))  error = y\_hat - y\_qn[:,n]    e = 2\*error\*y\_hat\*(1-y\_hat)  w\_diff = e\*f\_a    w\_00 = e[0]\*w\_total[0,0] + e[1]\*w\_total[0,1] + e[2]\*w\_total[0,2]  w\_10 = e[0]\*w\_total[1,0] + e[1]\*w\_total[1,1] + e[2]\*w\_total[1,2]  w\_20 = e[0]\*w\_total[2,0] + e[1]\*w\_total[2,1] + e[2]\*w\_total[2,2]    w\_4 = np.hstack([w\_00,w\_10,w\_20])    for m in range(2):  for l in range(3):  v\_diff[m][l] = f\_a[l]\*(1-f\_a[l])\*x\_mn[m,n]\*w\_4[l]    w\_total = w\_total - LR\*w\_diff  v\_total = v\_total - LR\*v\_diff  print(v\_total)  print(w\_total) |

|  |
| --- |
| 매개변수 및 설명 |
| 학습률(LR) = 0.0000001, N개의 개별 데이터에 대해 학습을 진행하는 내부 루프(Epoch) = 10000 으로 하여 오차역전파 알고리즘을 실행한 결과 최적매개변수는 아래와 같습니다.  V = [-1.11970471 -0.05772331 0.0948062 ] W = [ -0.39744977 0.02309383 0.31839417]  [ 0.81044282 -0.09921743 0.15051879] [-6.12744977 3.58309383 4.23839417]  [ 0.49 9.84 -7.96 ] [ 1.15255023 4.80309383 -6.16160583]  [ 0.94255023 -5.92690617 -1.11160583]    오차 역전파 알고리즘에서 은닉층 노드의 입력 벡터(alpha) = , 은닉층 노드의 출력 벡터(b) =  출력층 입력 벡터(beta) = , 출력층 출력 벡터(y\_hat) = 로 Forward의 데이터 부분을 생성하였습니다.  예측 오차(e) = 로 구하고, alpha의 시그모이드 함수를 적용한 값(f\_a)과 곱하여(w\_diff) 매개변수 w(w\_total)을 업데이트 하였습니다.  v(v\_total)는 (v\_diff)의 식을 이용하여 업데이트 하였습니다. |

**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('NN\_2d\_data.csv',names = ['x','y','z'])  middle = np.asarray(raw\_data['x'].values.tolist())  final = np.asarray(raw\_data['y'].values.tolist())  grade = np.asarray(raw\_data['z'].values.tolist())  y0 = np.zeros(500)  y1 = np.zeros(500)  y2 = np.zeros(500)  a = 0  for i in range(500):  if grade[i] == 'A':  y0[a] = 1  a = a+1  elif grade[i] == 'B':  y1[a] = 1  a = a+1  else:  y2[a] = 1  a = a+1  y\_qn = np.stack([y0,y1,y2])  v\_total = np.array([[-1.12, -0.05, 0.09],[0.81, -0.10, 0.15],[0.49, 9.84, -7.96]])  w\_total = np.array([[-0.40, 0.02, 0.32],[-6.13, 3.58, 4.24],[1.15, 4.80, -6.16],[0.94, -5.93, -1.11]])  x2 = np.ones(500)  x\_mn = np.stack([middle,final,x2])  LR = 0.0000001  epoch = [0]\*100  for h in range(100):  epoch[h] = h+1  max\_number = np.zeros(500)  b = np.zeros([4,500])  b\_dummy = np.ones(1)  y\_hat\_nq = np.zeros([500,3])  v\_diff = np.zeros([3,3])  total\_mse = np.zeros(epoch[-1])  accuracy = np.zeros(epoch[-1])  mse = np.zeros(500)  for o in range(len(epoch)):  for z in range(epoch[o]):  for n in range(500):  alpha = np.dot(np.array(v\_total).T,x\_mn[:,n])  f\_a = 1 / (1 + np.exp(-alpha))  b = np.hstack([f\_a,b\_dummy])    beta = np.dot(np.array(w\_total).T, b)  y\_hat = 1 / (1 + np.exp(-beta))  error = y\_hat - y\_qn[:,n]    e = 2\*error\*y\_hat\*(1-y\_hat)  w\_diff = e\*f\_a    w\_00 = e[0]\*w\_total[0,0] + e[1]\*w\_total[0,1] + e[2]\*w\_total[0,2]  w\_10 = e[0]\*w\_total[1,0] + e[1]\*w\_total[1,1] + e[2]\*w\_total[1,2]  w\_20 = e[0]\*w\_total[2,0] + e[1]\*w\_total[2,1] + e[2]\*w\_total[2,2]    w\_4 = np.hstack([w\_00,w\_10,w\_20])    for m in range(2):  for l in range(3):  v\_diff[m][l] = f\_a[l]\*(1-f\_a[l])\*x\_mn[m,n]\*w\_4[l]    w\_total = w\_total - LR\*w\_diff  v\_total = v\_total - LR\*v\_diff    compare\_y\_hat = np.zeros([3,500])  y\_hat\_nq[n] = y\_hat  mse[n] = sum((y\_hat - y\_qn[:,n])\*(y\_hat - y\_qn[:,n]))    total\_mse[o] = np.mean(mse)    conformity = 0  for n in range(500):  max\_number = y\_hat\_nq[n,0]  a = 0  if y\_hat\_nq[n,1] > max\_number:  max\_number = y\_hat\_nq[n,1]  a = 1  if y\_hat\_nq[n,2] > max\_number:  max\_number = y\_hat\_nq[n,2]  a = 2    compare\_y\_hat[a,n] = 1    if np.all(compare\_y\_hat[:,n] == y\_qn[:,n]):  conformity = conformity + 1  else:  conformity = conformity  accuracy[o] = (conformity/500)\*100  plt.subplot(2,1,1)  plt.plot(epoch,accuracy,'k-o')  plt.legend(['accuracy'])  plt.xlabel('lteration')  plt.ylabel('Accuracy')  plt.grid(True)  plt.subplot(2,1,2)  plt.plot(epoch,total\_mse,'k-o')  plt.legend(['mse'])  plt.xlabel('lteration')  plt.ylabel('MSE')  plt.grid(True)  plt.show() |

|  |
| --- |
| 그래프 |
|  |

|  |
| --- |
| 설명 |
| Epoch를 [0~100]까지 리스트를 만들고, ‘for’문을 사용하여 반복 횟수에 따른 예측 정확도 및 평균 제곱오차를 정확도array(accuracy)와 평균 제곱오차array(total\_mse)에 저장하였습니다.  평균 제곱오차는 의 수식을 이용하여 만들었습니다.    예측 정확도는 출력층 출력 벡터(y\_hat)에서 각 행마다 0~2의 열의 확률 값을 비교하여 가장 높은 열의 index에 1을 넣고, 실제 출력(y\_qn)과 비교하여 정확도를 구하였습니다.  실습 결과 epoch가 증가할수록 평균 제곱오차의 값이 감소하고, 예측 정확도는 증가함을 확인할 수 있었습니다.  Epoch가 100일 때의 예측 정확도 값은 예측 정확도 98.2 / 평균 제곱오차 값은 0.108936 의 결과가 나왔습니다. |

**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('NN\_2d\_data.csv',names = ['x','y','z'])  middle = np.asarray(raw\_data['x'].values.tolist())  final = np.asarray(raw\_data['y'].values.tolist())  grade = np.asarray(raw\_data['z'].values.tolist())  a = 0  b = 0  c = 0  A\_grade\_middle = np.zeros(112)  A\_grade\_final = np.zeros(112)  B\_grade\_middle = np.zeros(263)  B\_grade\_final = np.zeros(263)  C\_grade\_middle = np.zeros(125)  C\_grade\_final = np.zeros(125)  for i in range(500):  if grade[i] == 'A':  A\_grade\_middle[a] = middle[i]  A\_grade\_final[a] = final[i]  a = a+1  elif grade[i] == 'B':  B\_grade\_middle[b] = middle[i]  B\_grade\_final[b] = final[i]  b = b+1  else:  C\_grade\_middle[c] = middle[i]  C\_grade\_final[c] = final[i]  c = c+1  y0 = np.zeros(500)  y1 = np.zeros(500)  y2 = np.zeros(500)  a = 0  for i in range(500):  if grade[i] == 'A':  y0[a] = 1  a = a+1  elif grade[i] == 'B':  y1[a] = 1  a = a+1  else:  y2[a] = 1  a = a+1  y\_qn = np.stack([y0,y1,y2])  v\_total = np.array([[-1.12, -0.05, 0.09],[0.81, -0.10, 0.15],[0.49, 9.84, -7.96]])  w\_total = np.array([[-0.40, 0.02, 0.32],[-6.13, 3.58, 4.24],[1.15, 4.80, -6.16],[0.94, -5.93, -1.11]])  x2 = np.ones(500)  x\_mn = np.stack([middle,final,x2])  LR = 0.0000001  epoch = 10000  max\_number = np.zeros(500)  b = np.zeros([4,500])  b\_dummy = np.ones(1)  y\_hat\_nq = np.zeros([500,3])  v\_diff = np.zeros([3,3])  for z in range(epoch):  for n in range(500):  alpha = np.dot(np.array(v\_total).T,x\_mn[:,n])  f\_a = 1 / (1 + np.exp(-alpha))  b = np.hstack([f\_a,b\_dummy])    beta = np.dot(np.array(w\_total).T, b)  y\_hat = 1 / (1 + np.exp(-beta))  error = y\_hat - y\_qn[:,n]    e = 2\*error\*y\_hat\*(1-y\_hat)  w\_diff = e\*f\_a    w\_00 = e[0]\*w\_total[0,0] + e[1]\*w\_total[0,1] + e[2]\*w\_total[0,2]  w\_10 = e[0]\*w\_total[1,0] + e[1]\*w\_total[1,1] + e[2]\*w\_total[1,2]  w\_20 = e[0]\*w\_total[2,0] + e[1]\*w\_total[2,1] + e[2]\*w\_total[2,2]    w\_4 = np.hstack([w\_00,w\_10,w\_20])    for m in range(2):  for l in range(3):  v\_diff[m][l] = f\_a[l]\*(1-f\_a[l])\*x\_mn[m,n]\*w\_4[l]    w\_total = w\_total - LR\*w\_diff  v\_total = v\_total - LR\*v\_diff    compare\_y\_hat = np.zeros([3,500])  y\_hat\_nq[n] = y\_hat  test\_x = np.zeros(100)  test\_x0 = np.zeros(10000)  test\_x1 = np.zeros(10000)  test\_x = np.linspace(0, 100, 100)    l = 0  for k in range(100):  for q in range(100):  test\_x0[l] = test\_x[k]  test\_x1[l] = test\_x[q]  l = l+1  test\_x2 = np.ones(10000)  total\_test\_xx = np.stack([test\_x0,test\_x1])  total\_test\_x = np.stack([test\_x0,test\_x1,test\_x2])  max\_index = np.zeros(10000)  w = 0  u = 0  e = 0  for i in range(10000):  test\_alpha = np.dot(np.array(v\_total).T,total\_test\_x[:,i])  test\_f\_a = 1 / (1 + np.exp(-test\_alpha))  test\_b = np.hstack([test\_f\_a,b\_dummy])    test\_beta = np.dot(np.array(w\_total).T, test\_b)  test\_y\_hat = 1 / (1 + np.exp(-test\_beta))    max\_index[i] = np.argmax(test\_y\_hat)    if max\_index[i] == 0:  w += 1  elif max\_index[i] == 1:  u += 1  else:  e += 1  test\_x0\_0 = np.zeros(w)  test\_x1\_0 = np.zeros(w)  test\_x0\_1 = np.zeros(u)  test\_x1\_1 = np.zeros(u)  test\_x0\_2 = np.zeros(e)  test\_x1\_2 = np.zeros(e)  w = 0  u = 0  e = 0  for i in range(10000):    if max\_index[i] == 0:  test\_x0\_0[w] = total\_test\_xx[0,i]  test\_x1\_0[w] = total\_test\_xx[1,i]  w += 1  elif max\_index[i] == 1:  test\_x0\_1[u] = total\_test\_xx[0,i]  test\_x1\_1[u] = total\_test\_xx[1,i]  u += 1  else:  test\_x0\_2[e] = total\_test\_xx[0,i]  test\_x1\_2[e] = total\_test\_xx[1,i]  e += 1    plt.scatter(test\_x0\_0,test\_x1\_0)  plt.scatter(test\_x0\_1,test\_x1\_1)  plt.scatter(test\_x0\_2,test\_x1\_2)  plt.plot(A\_grade\_middle,A\_grade\_final,'ro',B\_grade\_middle,B\_grade\_final,'bx',C\_grade\_middle,C\_grade\_final,'ks')  plt.xlabel('MIDDLE')  plt.ylabel('FINAL')  plt.grid(True) |

|  |
| --- |
| 그래프 |
|  |

|  |
| --- |
| 설명 |
| 실습 06은 Epoch를 10000번 반복하여 얻은 오차 역전파 모델에 x축과 y축을 0과 100 사이의 일정한 간격으로 100개씩 나누어 만든 10,000개의 좌표 값을 넣어 결정 경계를 확인해보고 주어진 데이터 값들이 잘 분류가 되었는지 그래프로 확인해보는 실습입니다.  x축과 y축을 0과 100 사이의 일정한 간격으로 100개씩 나누어 만든 10,000개의 좌표 값은 ‘np.linspace(0,100,100)’을 값들을 test\_x의 array에 저장한다음, ‘이중 for문’을 사용하여 입력속성 x0인 x축 좌표를 만들고 x1의 입력속성이 될 y축 좌표를 만들고 ‘np.ones(10000)’을 이용하여 입력 속성 x2(더미노드)를 만든 후, np.stack을 이용하여 One-hot 인코딩 입력 데이터를 생성하였습니다.  위의 입력 데이터를 오차 역전파 모델에 적용하여 출력층 출력 벡터(test\_y\_hat)을 얻고, 출력층 출력 벡터의 각 행마다 0~2의 열의 확률 값을 비교하여 가장 높은 열의 index에 1을 넣어 최종 출력 데이터를 생성하였습니다.  그 다음, 최종 출력 데이터에서 y0값이 1일 때의 입력 데이터(test\_x0\_0, test\_x1\_0) / y1 값이 1일 때의 입력 데이터(test\_x0\_1, test\_x1\_1) / y2값이 1일 때의 입력 데이터(test\_x0\_2, test\_x1\_2)를 나누어 ‘plt.scatter’을 사용하여 그래프를 그렸습니다. 그 결과, x축과 y축을 0과 100 사이의 일정한 간격으로 100개씩 나누어 만든 10,000개의 좌표 값을 넣어 결정 경계를 확인할 수 있었습니다. |