**EIEN 443 Artificial Neural Networks and Deep Learning (2021 Spring)**

Homework #2 (Due: Apr 12. 2021)

Name: \_\_\_\_\_\_\_\_\_박정수\_\_\_\_\_\_\_\_\_\_ Student ID: \_\_\_\_\_\_2016270431\_\_\_\_\_\_\_\_

**Q1 (1 point)**

|  |
| --- |
| import numpy as np  import pandas as pd  from tensorflow.keras import models, layers  from tensorflow.keras.callbacks import EarlyStopping  import matplotlib.pyplot as plt  a = pd.read\_csv('/home/oms315/Desktop/HW2.txt')  a.columns = ['id', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape',  'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',  'Normal Nucleoli', 'Mitoses', 'Class']  a.drop(['id'], inplace=True, axis=1)  a['Class'] = a['Class'].map(lambda k: 1 if k == 4 else 0)  x = np.array(a.drop(['Class'], axis=1))  y = np.array(a['Class'])  test\_data = x[:100].astype('float64')  test\_targets = y[:100].astype('float64')  val\_data = x[100:200].astype('float64')  val\_targets = y[100:200].astype('float64')  train\_data = x[200:].astype('float64')  train\_targets = y[200:].astype('float64')  mean = train\_data.mean(axis=0)  train\_data -= mean  std = train\_data.std(axis=0)  train\_data /= std  test\_data -= mean  test\_data /= std  val\_data -= mean  val\_data /= std  model = models.Sequential()  model.add(layers.Dense(10, activation='relu', input\_shape=(9,)))  model.add(layers.Dense(10, activation='relu'))  model.add(layers.Dense(1, activation='sigmoid'))  model.compile(optimizer='rmsprop', loss='binary\_crossentropy', metrics=['accuracy'])  history = model.fit(train\_data, train\_targets, epochs=200, batch\_size=10, validation\_data=(val\_data, val\_targets), callbacks=[EarlyStopping(monitor='val\_loss', patience=2)])  result = model.evaluate(test\_data, test\_targets)  print(result)  print(model.predict(test\_data))  print(test\_targets)  def plot\_loss(h, title="loss"):  plt.plot(h.history['loss'])  plt.plot(h.history['val\_loss'])  plt.title(title)  plt.ylabel('Loss')  plt.xlabel('Epoch')  plt.legend(['Training', 'Validation'], loc=0)  plt.show()  plot\_loss(history) |

**Q2 (1 point)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Trial #1 | Trial #2 | Trial #3 | Trial #4 | Trial #5 |
| Training  Loss | 0.0621 | 0.0443 | 0.0674 | 0.0603 | 0.0683 |
| Training  Accuracy | 0.9794 | 0.9817 | 0.9755 | 0.9774 | 0.9758 |
| Test  Loss | 0.1325 | 0.1383 | 0.1337 | 0.1255 | 0.1389 |
| Test  Accuracy | 0.9402 | 0.9203 | 0.9401 | 0.9608 | 0.9701 |

**Your answer:**

**일관성 있다. 각각의 Trial에 대한 Loss와Accuracy는 큰 변화가 없기 때문이다.**

**Q3 (1 point)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | None | Relu | Sigmoid | tanh |
| Training  Loss | 0.0643±0.0026 | 0.0588±  0.0084 | 0.0733±  0.0032 | 0.0707±  0.0027 |
| Training  Accuracy | 0.9743±  0.0021 | 0.9778±  0.0032 | 0.9765±  0.0014 | 0.9761±  0.0026 |
| Test  Loss | 0.1491±  0.0147 | 0.1387±  0.0101 | 0.1704±  0.0202 | 0.1592±  0.0123 |
| Test  Accuracy | 0.9329±  0.0105 | 0.9412±  0.0167 | 0.933±  0.0067 | 0.9299±  0.0081 |

**Your answer:**

**Activation function으로 Relu를 사용했을 때가 가장 좋다. 그 이유는 Training Loss, Test Loss가 가장 작고 Training Accuracy, Test Accuracy가 가장 크기 때문이다.**

**Q4 (2 points)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 2 | 5 | 10 | 20 | 50 |
| Training  Loss | 0.0794±  0.0084878 | 0.0632±  0.00351 | 0.0623±  0.006506 | 0.0605±  0.003997 | 0.0516±  0.002681 |
| Training  Accuracy | 0.9767±  0.00091815 | 0.9805±  0.005008 | 0.9747±  0.004221 | 0.9764±  0.003185 | 0.9805±  0.001828 |
| Test  Loss | 0.188±  0.045728 | 0.1608±  0.035835 | 0.1279±  0.016396 | 0.1327±  0.019025 | 0.1336±  0.011389 |
| Test  Accuracy | 0.9159±  0.033569 | 0.9279±  0.019225 | 0.9319±  0.008361 | 0.9339±  0.011384 | 0.9379±  0.008343 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 100 | 200 | 500 | 1000 | 2000 |
| Training  Loss | 0.0508±  0.001358 | 0.0442±  0.002605 | 0.0376±  0.001539 | 0.0312±  0.00745 | 0.0378±  0.003303 |
| Training  Accuracy | 0.9797±  0.001709 | 0.9826±  0.001878 | 0.9851±  0.000929 | 0.9892±  0.002688 | 0.9851±  0.002283 |
| Test  Loss | 0.1287±  0.020887 | 0.1264±  0.009558 | 0.1253±  0.007467 | 0.1853±  0.049369 | 0.2426±  0.048159 |
| Test  Accuracy | 0.9356±  0.011354 | 0.9379±  0.008343 | 0.9380±  0.004439 | 0.916±  0.008944 | 0.9058±  0.013535 |

**Your answer:**

**히든 뉴런의 개수가 500인 경우가 Best Case다. 그 이유는 히든 뉴런의 개수가 500개보다 작은 경우보다 Loss가 작고 Accuracy가 높다. 또한 히든 뉴런의 개수가 1000,2000개인 경우는 Test Loss가 증가하였고 Accuracy는 감소하였다. 위와 같은 이유로 볼 때 히든 뉴런의 개수가 500인 경우가 Loss를 줄이고 Accuracy를 증가시키는 최선의 경우이다.**

**Q E1 (0.5 points)**

**Your answer:**

**히든 뉴런의 개수는 특정개수로 정해져있지 않기 때문에 최적의 개수를 찾아야한다. 따라서 모든 경우를 전부 시도해보는 것이 아니라 대략적으로 Best Case가 나오는 경우를 찾은 후 그것을 기반으로 추가적인 Search를 진행해야한다.**