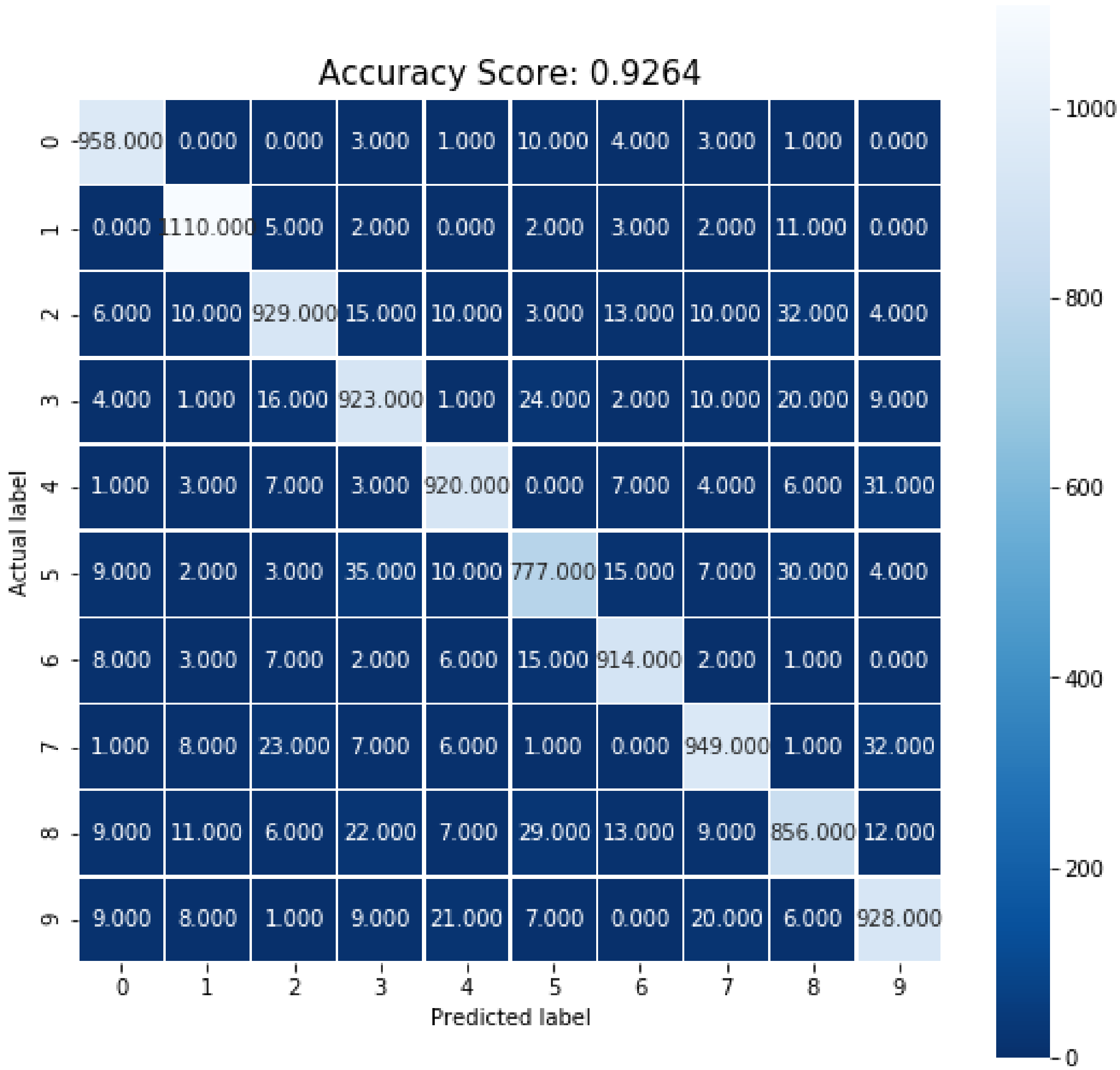
COEN 240 Machine Learning

Homework #3

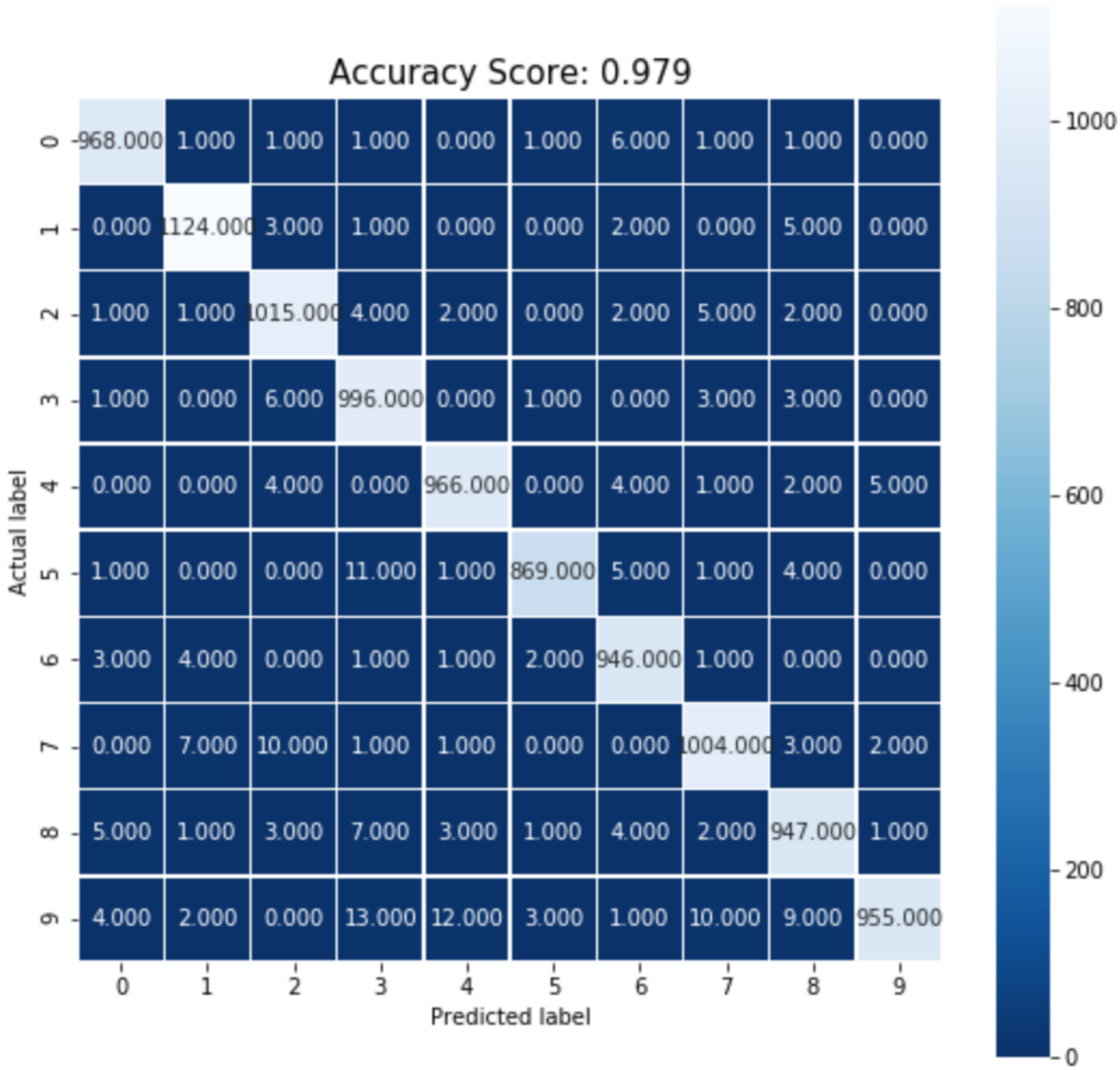
Name: Alex Cherekdjian **Student ID:** 00001083236

Problem 1



Comment: The logistic regression confusion matrix shows a fairly accurate recognition score of around 92% with a fairly consistent distribution of miss labeled predictions.

Problem 2



Comment: The neural network confusion matrix shows a much better recognition accuracy than the logistic regression of around 97% and a much smaller distribution of miss labeled predictions.

Problem 3

3a

$$\delta_k = \frac{\partial E_n}{\partial a_k} \qquad E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - t_{nk})^2$$

$$y_{nk} = y_k(x_n) \qquad y_k = \frac{1}{1+e^{-a_k}} = \text{sigmoid}$$

$$\delta_k = \frac{\partial E_n}{\partial y_{nk}}\frac{\partial y_{nk}}{\partial a_k} = \sum_{k=1}^{K}(y_{nk} - t_{nk})\frac{\partial y_{nk}}{\partial a_k}$$

$$\frac{\partial y_{nk}}{\partial a_k} = \sigma(a_k)(1-\sigma(a_k)) \qquad \boxed{\delta_k = \sum_{k=1}^{K}(y_{nk} - t_{nk})\,\sigma(a_k)(1-\sigma(a_k))}$$

3.b.

$$\delta_j = \frac{\partial E_n}{\partial a_j}$$

$$= \sum_k \underbrace{\frac{\partial E_n}{\partial a_k}}_{\delta_{a_k}} \frac{\partial a_k}{\partial a_j} = \delta_j = h'(a_j)\sum_k w_{kj}\delta_k$$

$$\frac{\partial a_k}{\partial a_j} = w_{kj}\,h'(a_j)$$

$$h(a_j) = \tanh(a_j) = \frac{e^{a_j} - e^{-a_j}}{e^{a_j} + e^{-a_j}}$$

$$h'(a_j) = 1 - h^2(a_j) = 1 - \tanh^2(a_j)$$

$$\boxed{\delta_j = 1 - \tanh^2(a_j)\sum_{k=1}^{K} w_{kj}(y_{nk} - t_{nk})\,\sigma(a_k)(1-\sigma(a_k))}$$

Attachment

Problem 1 Code (in zip file):

```
import tensorflow as tf
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

# uploading data from mnist dataset
mnist = tf.keras.datasets.mnist
(x_traino, y_train),(x_testo, y_test) = mnist.load_data()

# reshaping matrices and init the regression solver
x_train = np.reshape(x_traino,(60000,28*28))
x_test = np.reshape(x_testo,(10000,28*28))
x_train, x_test = x_train / 255.0, x_test / 255.0
logreg = LogisticRegression(solver='saga', multi_class='multinomial',max_iter
= 100,verbose=2)

# train the algorithm
logreg.fit(x_train, y_train)

# predict all values from dataset
predictions = logreg.predict(x_test)

# create confusion matrix and accuracy score
predictions_cm = confusion_matrix(y_test, predictions)
recognition_accuracy_rate = accuracy_score(y_test, predictions)

# plot using seaborn to make it look nice
plt.figure(figsize=(9,9))
sns.heatmap(predictions_cm, annot=True, fmt=".3f", linewidths=.5, square =
True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(recognition_accuracy_rate)
plt.title(all_sample_title, size = 15)
```

Problem 2 Code (in zip file):

```
import tensorflow as tf
import numpy as np
import seaborn as sns
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

# uploading data from mnist dataset
mnist = tf.keras.datasets.mnist
(x_traino, y_train),(x_testo, y_test) = mnist.load_data()

# reshaping matrices and init the regression solver
x_train = np.reshape(x_traino,(60000,28*28))
x_test = np.reshape(x_testo,(10000,28*28))
x_train, x_test = x_train / 255.0, x_test / 255.0

# fix random seed for reproducibility
numpy.random.seed(7)

# create the model
# model.add(Flatten(input_shape(28,28))
model = Sequential()
model.add(Dense(512, input_dim=784, activation='relu')) # input 784
#model.add(Dense(512, activation='relu')) # hidden 512, reLu
model.add(Dense(10, activation='softmax')) # output 10 nodes, soft max

# compile the model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# fit the model
model.fit(x_train, y_train, epochs=5, batch_size=64, verbose=2)

# calculate predictions
predictions_net = model.predict(x_test)

# finding values from prob percentages of each category
prediction = np.argmax(predictions_net, axis=1)

# create confusion matrix and accuracy score
```

```
predictions_cm = confusion_matrix(y_test, prediction)
recognition_accuracy_rate_net = accuracy_score(y_test, prediction)

# plot using seaborn to make it look nice
plt.figure(figsize=(9,9))
sns.heatmap(predictions_cm, annot=True, fmt=".3f", linewidths=.5, square =
True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score:
{0}'.format(recognition_accuracy_rate_net)
plt.title(all_sample_title, size = 15)
```