Spring 2024: NN&DeepLearning_Spring25_Assignment4

Assignment-4

NAME: Lakkireddy Sriram Reddy

STUDENT ID:700758340

Github link: https://github.com/sriram7040/Neural-network-and-deep-learning.git

Video link: https://drive.google.com/file/d/19czoi-HSAB0d54GGLZI-AC2JaOaTwgeZ/view?usp=drive_link

1. Use the use case in the class:

a. Add more Dense layers to the existing code and check how the accuracy changes.

2. Change the data source to Breast Cancer dataset * available in the source code folder and make required changes. Report accuracy of the model.

3. Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below). from sklearn.preprocessing import StandardScaler sc = StandardScaler() Breast Cancer dataset is designated to predict if a patient has Malignant (M) or Benign = B cancer

---

1.1 Add more Dense layers to the existing code and check how the accuracy changes.

[6]
```
import pandas as pd
import tensorflow as tf
data = pd.read_csv('diabetes.csv')
```

[13]
```
path_to_csv= 'diabetes.csv'
```

[15]

Generated code may be subject to a license | Adavellisahaja/Python
```
import keras
import pandas
from keras.models import Sequential
from keras.layers import Dense, Activation

# load dataset
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

dataset = pd.read_csv(path_to_csv, header=None).values

X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                    test_size=0.25, random_state=87)
```

```
dataset = pd.read_csv(path_to_csv, header=None).values

X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                     initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test))
```

```
Model: "sequential_5"
```

Layer (type)	Output Shape	Param #
dense_12 (Dense)	(None, 20)	180
dense_13 (Dense)	(None, 1)	21

```
 Total params: 605 (2.37 KB)
 Trainable params: 201 (804.00 B)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 404 (1.58 KB)
None
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - acc: 0.7158 - loss: 0.6210
[0.6654766201972961, 0.6666666865348816]
```

1.2 Change the data source to Breast Cancer dataset

```
[16] path_to_csv= 'breastcancer.csv'
```

```
from re import X
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

#load dataset
cancer_data = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer
my_first_nn.add(Dense(30, activation='relu')) # hidden layer2
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                     initial_epoch=0)
```

```python
np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer
my_first_nn.add(Dense(30, activation='relu')) # hidden layer2
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                      initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test,Y_test))
```

```
14/14                    0s 3ms/step - acc: 0.9462 - loss: 0.1266
Model: "sequential_6"
```

Layer (type)	Output Shape	Param #
dense_14 (Dense)	(None, 20)	620
dense_15 (Dense)	(None, 30)	630
dense_16 (Dense)	(None, 1)	31

```
 Total params: 3,845 (15.02 KB)
 Trainable params: 1,281 (5.00 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2,564 (10.02 KB)
None
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - acc: 0.8812 - loss: 0.2781
[0.23744958639144897, 0.9090909361839294]
```

1.3 Normalize the data before feeding the data

```python
from re import X
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load dataset
cancer_data = load_breast_cancer()
X = cancer_data.data
y = cancer_data.target

# Normalize the data
sc = StandardScaler()
X_scaled = sc.fit_transform(X)

# Split dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=87)

# Set random seed for reproducibility
np.random.seed(155)
```

```python
# Set random seed for reproducibility
np.random.seed(155)

# Create the neural network model
my_first_nn = Sequential()
my_first_nn.add(Dense(20, input_dim=30, activation='relu'))  # Hidden layer
my_first_nn.add(Dense(1, activation='sigmoid'))  # Output layer

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

# Train the model
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, initial_epoch=0, verbose=1)

# Print model summary
print(my_first_nn.summary())

# Evaluate the model on the test set
loss, accuracy = my_first_nn.evaluate(X_test, Y_test)
print(f"Neural Network Model Accuracy: {accuracy:.4f}")
```

Model: "sequential_7"

Layer (type)	Output Shape	Param #
dense_17 (Dense)	(None, 20)	620
dense_18 (Dense)	(None, 1)	21

```
 Total params: 1,925 (7.52 KB)
 Trainable params: 641 (2.50 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 1,284 (5.02 KB)
None
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9659 - loss: 0.2223
Neural Network Model Accuracy: 0.9720
```

2.Use Image Classification on the hand written digits data set (mnist)

1. Plot the loss and accuracy for both training data and validation data using the history object in the source code.

2. Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.

3. We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.

4. Run the same code without scaling the images and check the performance?

```python
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical

(train_images,train_labels),(test_images, test_labels) = mnist.load_data()

print(train_images.shape[1:])
#process the data
#1. convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
print(dimData)
train_data = train_images.reshape(train_images.shape[0],dimData)
test_data = test_images.reshape(test_images.shape[0],dimData)

#convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')
#scale data
train_data /=255.0
test_data /=255.0
#change the labels frominteger to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)
```



```python
train_data /=255.0
test_data /=255.0
#change the labels frominteger to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)
```



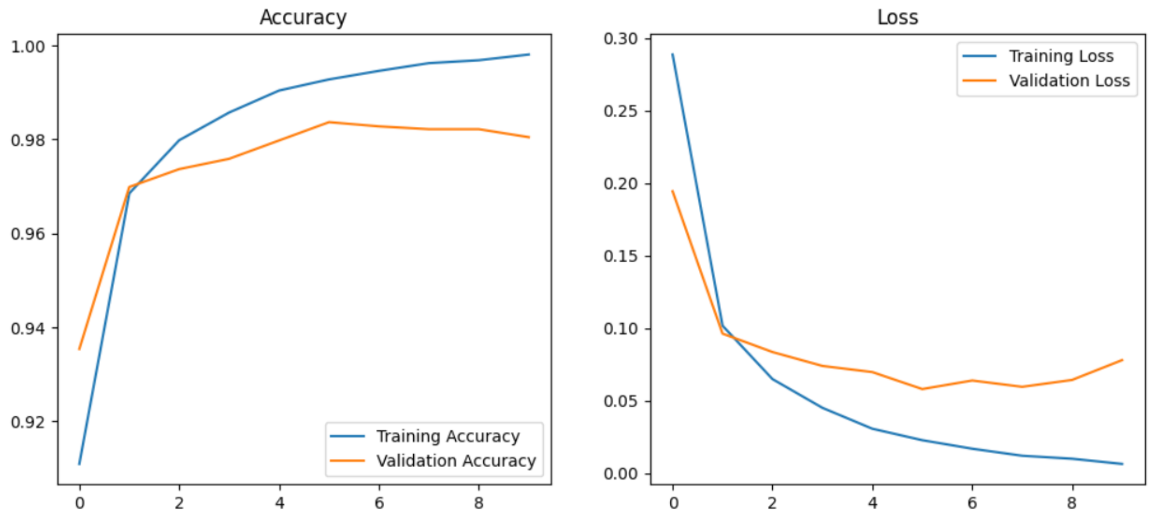
```python
#creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))
```

2.1 Plot the loss and accuracy for both training data and validation data

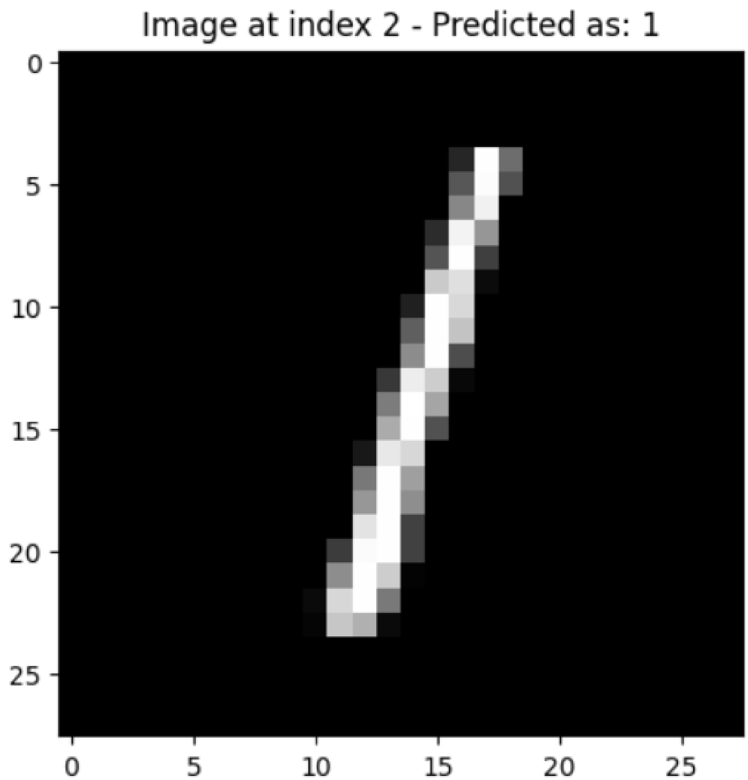
```python
#plotting the graph
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.legend()
plt.show()
```




```python
def predict_single_image(image_data):
    image_data = image_data.reshape(1, dimData).astype('float32') / 255
    prediction = model.predict(image_data)
    predicted_class = np.argmax(prediction)
    return predicted_class
    # Choose an image from the test set
image_index = 2  # Change this to see different predictions
predicted_class = predict_single_image(test_images[image_index])
print(f'Predicted class for image at index {image_index}: {predicted_class}')
plt.imshow(test_images[image_index], cmap='gray')
plt.title(f'Image at index {image_index} - Predicted as: {predicted_class}')
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step
Predicted class for image at index 2: 1
```



2.3 We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.

```
[22] model_tanh = Sequential([
         Dense(512, activation='tanh', input_shape=(dimData,)),
         Dense(10, activation='softmax')
     ])

     model_tanh.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
     model_tanh.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))
```

2.4 Run the same code without scaling the images and check the performance?

```
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical

# Load dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

print(train_images.shape[1:])

# Process the data
# 1. Convert each image of shape 28x28 to 784-dimensional vector
dimData = np.prod(train_images.shape[1:])
print(dimData)
train_data = train_images.reshape(train_images.shape[0], dimData)
test_data = test_images.reshape(test_images.shape[0], dimData)

# Convert data to float (without scaling)
train_data = train_data.astype('float')
test_data = test_data.astype('float')
```

```
# Convert labels to one-hot encoding
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

# Creating neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))
```