Assignment No: 2A

Roll No. : - 19121028

Title of the Assignment: Binary classification using Deep Neural Networks

Example: Classify movie reviews into positive" reviews and "negative" reviews, just based on the text content of the reviews.Use IMDB dataset

```
from keras.datasets import imdb

# Load the dataset
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

```
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
    17464789/17464789 [==============================] - 0s 0us/step
```

```
import numpy as np

# Preprocess the dataset
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)

y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

```
from keras.models import Sequential
from keras.layers import Dense

# Build the DNN model
model = Sequential()
model.add(Dense(units=16, activation='relu', input_dim=10000))
model.add(Dense(units=16, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))
```

```
# Train the model
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10, batch_size=512)
```

```
    Epoch 1/10
    49/49 [==============================] - 3s 47ms/step - loss: 0.4935 - accuracy: 0.8022
    Epoch 2/10
    49/49 [==============================] - 2s 41ms/step - loss: 0.2936 - accuracy: 0.8985
    Epoch 3/10
    49/49 [==============================] - 2s 45ms/step - loss: 0.2254 - accuracy: 0.9196
    Epoch 4/10
    49/49 [==============================] - 3s 52ms/step - loss: 0.1918 - accuracy: 0.9309
    Epoch 5/10
    49/49 [==============================] - 3s 58ms/step - loss: 0.1666 - accuracy: 0.9413
    Epoch 6/10
    49/49 [==============================] - 3s 66ms/step - loss: 0.1501 - accuracy: 0.9472
    Epoch 7/10
    49/49 [==============================] - 2s 49ms/step - loss: 0.1324 - accuracy: 0.9550
    Epoch 8/10
    49/49 [==============================] - 2s 44ms/step - loss: 0.1196 - accuracy: 0.9603
    Epoch 9/10
    49/49 [==============================] - 2s 45ms/step - loss: 0.1096 - accuracy: 0.9635
    Epoch 10/10
    49/49 [==============================] - 2s 35ms/step - loss: 0.0954 - accuracy: 0.9694
    <keras.callbacks.History at 0x7f0bf601e500>
```

```
# Test the model
loss, accuracy = model.evaluate(x_test, y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
    782/782 [==============================] - 2s 2ms/step - loss: 0.3754 - accuracy: 0.8711
    Test Loss: 0.3754294812679291
    Test Accuracy: 0.8711199760437012
```

```
from keras.datasets import imdb
from keras.preprocessing.text import Tokenizer
import numpy as np

# Load the dataset
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)

# Convert the integer sequences to text
word_index = imdb.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
decoded_review = ' '.join([reverse_word_index.get(i - 3, '?') for i in train_data[0]])

# Define the tokenizer
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts([decoded_review])  # Use a list of strings instead of integers

# Define the function to vectorize sequences
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results

# Convert the new review to a sequence of word indices
new_review = "This movie was terrible. I would not recommend it to anyone."
new_review = tokenizer.texts_to_sequences([new_review])[0]

# Vectorize the sequence
new_review = vectorize_sequences([new_review])

# Make the prediction
prediction = model.predict(new_review)
print("Prediction:", prediction[0][0])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_index.json
1641221/1641221 [==============================] - 0s 0us/step
1/1 [==============================] - 0s 178ms/step
Prediction: 0.48108512
```

Colab paid products - Cancel contracts here

✓ 6s completed at 12:33 PM