Neural Network Deep Learning

Assignment 9

Name: Pranavya Chanamolu

Student ID: 700739974

GitHub Link: https://github.com/pxc99740/Neural-Network-assignment-9.git

Video Link:
https://drive.google.com/drive/folders/1GVWc0l2bt51cOZCNBAkeUn7No9TStjgt

Lesson Overview:

In this lesson, we are going to discuss types of ANNs and Recurrent Neural Network.

Use Case Description:

1. Sentiment Analysis on the Twitter dataset

Programming elements:

1. Basics of LSTM

2. Types of RNN

3. Use case: Sentiment Analysis on the Twitter data set

In class programming:

1. Save the model and use the saved model to predict on new text data (ex, "A lot of good things are

happening. We are respected again throughout the world, and that's a great thing.@realDonaldTrump")

2. Apply GridSearchCV on the source code provided in the class

Files

```
[7] !pip install keras==2.12.0
    !pip install keras.utils
```

```
Requirement already satisfied: keras==2.12.0 in /usr/local/lib/python3.10/dist-packages (2.12.0)
Collecting keras.utils
  Downloading keras-utils-1.0.13.tar.gz (2.4 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Keras>=2.1.5 in /usr/local/lib/python3.10/dist-packages (from keras.utils) (2.12.0)
Building wheels for collected packages: keras.utils
  Building wheel for keras.utils (setup.py) ... done
  Created wheel for keras.utils: filename=keras_utils-1.0.13-py3-none-any.whl size=2631 sha256=405754aae144670d730febe9d1d913567286dd0795669d4c8d40af4e216aa
  Stored in directory: /root/.cache/pip/wheels/5c/c0/b3/0c332de4fd71f3733ea6d61697464b7ae4b2b5ff0300e6ca7a
Successfully built keras.utils
Installing collected packages: keras.utils
Successfully installed keras.utils-1.0.13
```

Files

```python
[10] import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
     from keras.preprocessing.text import Tokenizer
     from keras.utils import pad_sequences
     from keras.models import Sequential
     from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
     from matplotlib import pyplot
     from sklearn.model_selection import train_test_split
     from keras.utils import to_categorical

     import re

     from sklearn.preprocessing import LabelEncoder

     data = pd.read_csv('Sentiment.csv')
     # Keeping only the neccessary columns
     data = data[['text','sentiment']]
     # Filtering the tweets, using Tokenizer to vectorize, convert text into Sequences
     data['text'] = data['text'].apply(lambda x: x.lower())
     data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x)))

     for idx, row in data.iterrows():
         row[0] = row[0].replace('rt', ' ')

     max_fatures = 2000
     tokenizer = Tokenizer(num_words=max_fatures, split=' ')
     tokenizer.fit_on_texts(data['text'].values)
     X = tokenizer.texts_to_sequences(data['text'].values)

     X = pad_sequences(X)
```

Files

```python
[10] embed_dim = 128
     lstm_out = 196
     def createmodel():
         model = Sequential()
         model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))
         model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
         model.add(Dense(3,activation='softmax'))
         model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
         return model
     #print(model.summary())

     labelencoder = LabelEncoder()
     integer_encoded = labelencoder.fit_transform(data['sentiment'])
     y = to_categorical(integer_encoded)
     X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42)

     batch_size = 32
     model = createmodel()
     model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2)
     score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size)
     print(score)
     print(acc)
     print(model.metrics_names)
```

```
<ipython-input-10-b9a66b36efbb>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['text'] = data['text'].apply(lambda x: x.lower())
<ipython-input-10-b9a66b36efbb>:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x)))
291/291 - 50s - loss: 0.8220 - accuracy: 0.6456 - 50s/epoch - 173ms/step
144/144 - 3s - loss: 0.7666 - accuracy: 0.6536 - 3s/epoch - 24ms/step
0.7666357755661011
0.6535605192184448
['loss', 'accuracy']
```

```
[11] model.save('model.h5')
```

```
[12] from keras.models import load_model

    # Loading the saved model
    model = load_model('model.h5')

    # Text data for prediction
    new_text = ["A lot of good things are happening. We are respected again throughout the world, and that's a great thing. @realDonaldTrump"]

    # Tokenize and preprocess the new text data
    new_text = [re.sub('[^a-zA-z0-9\s]', '', text.lower()) for text in new_text]
    new_sequences = tokenizer.texts_to_sequences(new_text)
    new_sequences = pad_sequences(new_sequences, maxlen=X.shape[1])
    # Predict using the loaded model
    predictions = model.predict(new_sequences)
    print(predictions)
```

```
1/1 [==============================] - 0s 497ms/step
[[0.65722406 0.12540568 0.21737024]]
```

```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from keras.layers import LSTM

# Function to create the model, as it's required by KerasClassifier
def create_model(lstm_out=196, dropout=0.2):
    model = Sequential()
    model.add(Embedding(max_fatures, embed_dim, input_length=X.shape[1]))
    model.add(LSTM(lstm_out, dropout=dropout, recurrent_dropout=dropout))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

# Create the KerasClassifier
model = KerasClassifier(build_fn=create_model, epochs=1, batch_size=batch_size, verbose=2)

# Define the grid of parameters to search
param_grid = {
    'lstm_out': [196, 256],
    'dropout': [0.2, 0.3]
}

# Create GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(X_train, Y_train)

# Summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
<ipython-input-13-658eda5ed78a>:15: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See htt
  model = KerasClassifier(build_fn=create_model, epochs=1, batch_size=batch_size, verbose=2)
291/291 - 70s - loss: 0.8228 - accuracy: 0.6458 - 70s/epoch - 242ms/step
Best: 0.663618 using {'dropout': 0.2, 'lstm_out': 256}
```