**교육일지**

**교육 제목 : 딥러닝**

**교육 장소 : YGL C6 강의실**

**교육 일시 : 2021/11/04**

**주가 예측하기 (RNN, GRU, LSTM)**

**seq\_length= 2**

**x\_data\_dim=4**

**batch\_size= 100**

**min\_max\_normalization\_flag=True**

**data\_dir = '../dataset'**

**fname = '/content/drive/MyDrive/Colab Notebooks/영우\_4기\_딥러닝/answer/dataset/a\_company\_stock.csv'**

**df = pd.read\_csv(fname)**

**dataset=df.copy()**

**ori\_Y= dataset.pop("Close")**

**ori\_X=dataset.copy()**

**X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(ori\_X, ori\_Y, test\_size=0.2, shuffle=False)**

**X\_train, X\_val, Y\_train, Y\_val= train\_test\_split(X\_train, Y\_train, test\_size=0.2, shuffle=False)**

**## 데이터의 min , max, mean, std 값 구하기.**

**dataset\_stats = X\_train.describe()**

**dataset\_stats = dataset\_stats.transpose()**

**## data normalization**

**## data normalization**

**def min\_max\_norm(x):**

**return (x - dataset\_stats['min']) / (dataset\_stats['max'] - dataset\_stats['min'])**

**def standard\_norm(x):**

**return (x - dataset\_stats['mean']) / dataset\_stats['std']**

**if min\_max\_normalization\_flag==True:**

**min\_max\_norm\_train\_data = min\_max\_norm(X\_train)**

**min\_max\_norm\_val\_data = min\_max\_norm(X\_val)**

**min\_max\_norm\_test\_data = min\_max\_norm(X\_test)**

**data\_gen\_train=tf.keras.preprocessing.sequence.TimeseriesGenerator(min\_max\_norm\_train\_data.values.tolist(), Y\_train.values.tolist(), length=seq\_length, sampling\_rate=1,**

**batch\_size=batch\_size)**

**data\_gen\_val=tf.keras.preprocessing.sequence.TimeseriesGenerator(min\_max\_norm\_val\_data.values.tolist(), Y\_val.values.tolist(),length=seq\_length, sampling\_rate=1,**

**batch\_size=batch\_size) data\_gen\_test=tf.keras.preprocessing.sequence.TimeseriesGenerator(min\_max\_norm\_test\_data.values.tolist(), Y\_test.values.tolist(), length=seq\_length, sampling\_rate=1,**

**batch\_size=batch\_size)**

**else:**

**data\_gen\_train=tf.keras.preprocessing.sequence.TimeseriesGenerator(X\_train.values.tolist(),Y\_train.values.tolist(),length=seq\_length, sampling\_rate=1, batch\_size=batch\_size)**

**data\_gen\_val=tf.keras.preprocessing.sequence.TimeseriesGenerator(X\_val.values.tolist(),Y\_val.values.tolist(), length=seq\_length, sampling\_rate=1,**

**batch\_size=batch\_size)**

**data\_gen\_test = tf.keras.preprocessing.sequence.TimeseriesGenerator(X\_test.values.tolist(),Y\_test.values.tolist(), length=seq\_length, sampling\_rate=1, batch\_size=batch\_size)**

**input\_Layer = tf.keras.layers.Input(shape=(seq\_length, x\_data\_dim))**

**x = tf.keras.layers.SimpleRNN(20, activation='tanh')(input\_Layer)**

**x = tf.keras.layers.Dense(20, activation='relu')(x)**

**x = tf.keras.layers.Dense(20, activation='relu')(x)**

**Out\_Layer= tf.keras.layers.Dense(1, activation=None)(x)**

**model = tf.keras.Model(inputs=[input\_Layer], outputs=[Out\_Layer])**

**model.summary()**

**loss\_function= tf.keras.losses.mean\_squared\_error**

**optimize= tf.keras.optimizers.Adam(learning\_rate=0.001)**

**metric= tf.keras.metrics.mean\_absolute\_error**

**model.compile(loss = loss\_function,**

**optimizer = optimize,**

**metrics = [metric])**

**history = model.fit(data\_gen\_train,**

**validation\_data = data\_gen\_val,**

**steps\_per\_epoch = len(X\_train)/batch\_size,**

**epochs = 50,**

**validation\_freq = 1**

**)**

**print(model.evaluate(data\_gen\_test))**

**# tf.keras.layers.GRU()**

**input\_Layer = tf.keras.layers.Input(shape=(seq\_length, x\_data\_dim))**

**x = tf.keras.layers.SimpleRNN(20, activation='tanh')(input\_Layer)**

**x = tf.keras.layers.Dense(20, activation='relu')(x)**

**x = tf.keras.layers.Dense(10, activation='relu')(x)**

**Out\_Layer= tf.keras.layers.Dense(1, activation=None)(x)**

**model\_1 = tf.keras.Model(inputs=[input\_Layer], outputs=[Out\_Layer])**

**model\_1.summary()**

**# tf.keras.layers.LSTM()**

**input\_Layer = tf.keras.layers.Input(shape=(seq\_length, x\_data\_dim))**

**x = tf.keras.layers.SimpleRNN(20, activation='tanh')(input\_Layer)**

**x = tf.keras.layers.Dense(20, activation='relu')(x)**

**x = tf.keras.layers.Dense(10, activation='relu')(x)**

**Out\_Layer= tf.keras.layers.Dense(1, activation=None)(x)**

**model\_2 = tf.keras.Model(inputs=[input\_Layer], outputs=[Out\_Layer])**

**model\_2.summary()**

**# CNN + RNN**

**input\_Layer = tf.keras.layers.Input(shape=(seq\_length, x\_data\_dim))**

**x = tf.keras.layers.Conv1D(32, kernel\_size = 1, activation='tanh', return\_sequence= True)(input\_Layer)**

**x = tf.keras.layers.SimpleRNN(16, activation='tanh')(x)**

**x = tf.keras.layers.Dense(20, activation='relu')(x)**

**x = tf.keras.layers.Dense(10, activation='relu')(x)**

**Out\_Layer= tf.keras.layers.Dense(1, activation=None)(x)**

**model\_3 = tf.keras.Model(inputs=[input\_Layer], outputs=[Out\_Layer])**

**model\_3.summary()**

**test\_data\_X, test\_data\_Y=data\_gen\_test[0]**

**prediction\_Y=model.predict(test\_data\_X).flatten()**

**Y\_test=test\_data\_Y.flatten()**

**visual\_y=[]**

**visual\_pre\_y=[]**

**for i in range(len(prediction\_Y)):**

**label = Y\_test[i]**

**prediction = prediction\_Y[i]**

**print("실제가격: {:.3f}, 예상가격: {:.3f}".format(label, prediction))**

**visual\_y.append(label)**

**visual\_pre\_y.append(prediction)**

**time = range(1, len(visual\_y) + 1)**

**plt.plot(time, visual\_y, 'r', label='ture')**

**plt.plot(time, visual\_pre\_y, 'b', label='prediction')**

**plt.title('stock prediction')**

**plt.xlabel('time')**

**plt.ylabel('value')**

**plt.legend()**

**plt.show()**

**loss = history.history['loss']**

**val\_loss = history.history['val\_loss']**

**epochs = range(1, len(loss) + 1)**

**plt.plot(epochs, loss, 'bo', label='Training loss')**

**plt.plot(epochs, val\_loss, 'b', label='Validation loss')**

**plt.title('Training and validation loss')**

**plt.legend()**

**plt.show()**