Deep Learning inlämningsuppgift #2

Publishing: 3:e Maj 2023

Deadline 18:e Maj 2023

G : 70+ , VG : 90+

About project:

Trying to get a good model in RNN that fit multi-features datasets is very complex RNN setup that need very high-performance computing resources that cost a lot of money to rent or to own.

So, to save cost of training our RNN models for multiple feature (columns) , we are going to create our own MultiRNN that help AI engineers as AI developers to build best models faster using a normal computers regardless how many data rows we have into our data frames.

To achieve that we need to think about new algorithm that based on:

1. Developers should prepare clean data ready for processing.
2. Developers decide the train and test chunks of their data based on timestamp of their dataset as we do into our process.
3. Then our new class, **multiRNN**, that has special hyperparameters as explained into tables below will:
    a. **Validate** data frame in training and test if they are ready to process them.
    b. will generate new single column datasets, where each new dataset will have the original dataset timestamp as index, as well as one of each column into original train and test dataset.
    as next **example:**

Original train and/or test dataset:

*X are features or columns, 100 is not max of feature counts more it should be from 1 to **any** positive integer values.*

Index	X1	X2	X3	….	….	X100
1999-05-01						
1999-05-02						
……						
2023-05-01						

New Generated train and/or test datasets:

Index	X1
1999-05-01	
1999-05-02	
……	
2023-05-01	

Index	X2
1999-05-01	
1999-05-02	
……	
2023-05-01	

……

Index	X100
1999-05-01	
1999-05-02	
……	
2023-05-01	

c. Build RNN LSTM model for each generated dataset, so if we have till example 100 features, we shall train 100 models over generated train dataset and validate using generated train dataset.
d. Plot loss against val_loss per each model/feature.
e. Plot evaluation of the predictions of each feature model in the same plot of related generated test dataset.
f. Can predict all features with **single predict function** that gets input a row of all features that we have.

Here come how and what should you do to build this class:

1- **Main class MultiRNN**:

- MultiRNN is a class that helps developers to build Keras RNN model for multifeatured datasets.

2- **MultiRNN Class Parameters descriptions**: **(20 Points)**

parameters_name	Descriptions	Points
train **Type = Pandas Data frame**	Is a **valid** training pandas data frame of original dataset having index as original data index.	**3**
Test **Type = string**	Is a **valid** testing part of original dataset having index as original data index.	**3**
length **type = positive int**	Length of the output sequences.	**3**
LSTM_units **type = positive int or list of positive integers**	Positive integer, for cells in LSTM of keras or Pytorch **Or list** of positive integers for each feature LSTM cells needed for this feature that matches the index.	**3**
activation **Type = string or list of strings**	Valid Activation function for RNN LSTM layer default is 'tanh', other options: 'sigmoid', 'softmax', 'relu' **Or list** of activation function per each feature	**3**
optimizer **Type = string or list of strings**	Valid Optimizer for the output layer, the default is 'adam' other options for this class is 'rmsprop' and 'sgd' **Or list** of optimizers each per feature	**3**
batch_size. **type = int**	Integer Number of time series samples in each batch (except maybe the last one). batch_size will default to 1	**1**

epochs **Type = int or list of integers**	The number of epochs to train the model, default is 25 epochs per feature. Or list of epochs per each feature	**1**

3- Supporting Methods: **(80 Points)**

Methods name	Descriptions	Points
__init__	Constructor that takes above hyperparameters in and validate them.	5
ready_for_processing(self,data_set):	Method to check if the dataset is ready to process so it checks *Nan* values if any and has no object panda's data type etc…	5
generate_data_set_per_column_with_original_index (**self, data_set: pd.DataFrame):**	A Method that will generate datasets as described in **c above** *Optional: can save them to local folder.*	5
build_model_per_column(self, **train,** **test,** **length,** **LSTM_units,** **activation,** **batch_size,** **epochs):**	A method that takes inputs as you see in left column and creates RNN LSTM for the generated single column dataset, based on other parameter. This method will do needed scaling , generating time series and build the RNN LSTM Keras/pyTourch model based on hyper parameter *with early stopping* based on val_loss and patience of max 2 epochs.	25

	Optional: save losses data frame as csv to our working folder *Optional: save model as keras model/pyTorch model with .h5* This method also will evaluate the model,it creates the model predictions of the part of train data like och shorter than test dataset length and save it as DataFrame with original index in test values in order to plot both of them then. It returns: Model created. Losses, And related model name.	
generate_model_list_per_column_in_data_set(self, **train,** **test,** **length,** **LSTM_units,** **activation,** **batch_size,**	A method that creates list of models per each column. It start by calling *generate_data_set_per_column_with_original_index()* on both train and test Pandas DataFrame. Then send them into build_model_per_column() method in order to get output of keras/PyTorch model , losses DataFrame and model name. These will be run on each column into train dataset	10

epochs):	and collected into dictionary that has key as column name, and values is a dictionary that contains model, losses, and model name. (it is i dictionary of dictionaries) Return: The dictionary contains all columns. Tips: constructor will call this method!	
predict(self,data_row:pd.DataFrame)	A method to generate prediction for full columns (all features) and input data frame that has columns match the original dataset. Return: Prediction in form of Pandas Data frame	10
plot_predict_against_test_dataset_per_column(self ,column:str, figure_width:int , figure_hight:int , save_plot_name:str ='plot_test_vs_predict_'):	A method to plot the real test values and predicted values on the same plot using plot dimensions mentioned and column name. And save the plots.	10
plot_loss_val_loss_per_column(self,column:str, figure_width:int, figure_hight:int, save_plot_name:str ='Plot_Loss_val_loss_'):	A method to plot Loss against val_loss of input column and save it.	10

Good Luck!