lstm_loss_function

September 19, 2023

[1]:
```
from lstm_functions import *
from lost_functions import *
import numpy as np
import pandas as pd
from sklearn.compose import ColumnTransformer
import matplotlib.pyplot as plt
```

1 Use Apple Stock for testing purposes only

This is so that LSTM do not take 2 hour to run

[2]:
```
xls = pd.ExcelFile('data/data_for_testing.xlsx')
apple = pd.read_excel(xls, sheet_name="AAPL").set_index("Date").resample("M").
 ↪last().reset_index()
all_data = {"AAPL": apple[["Open", "High", "Low", "Close", "Adj Close",␣
 ↪"Volume"]]}
```

[3]:
```
final_importance_values = {}
final_predictions = {}
# 30 is not a good number of batches, but it's a start for testing
# 60 is a good number of batches, but it takes a long time to train
time_steps = 30
features = 6
```

[4]:
```
best_loss_value = float('inf')  # Initialize with a high value
best_loss_function = None

evaluation_results = {}  # Store evaluation results for each loss function
```

[5]:
```
for loss_func, loss_name in zip(loss_functions, loss_names):
    print("loss_names", loss_name)
    # Define the model
    lstm_model = LstmBuilder(time_step=time_steps, loss=loss_func)
    model = lstm_model.create_model(features=features)
    scaler = MinMaxScaler()
    normalized_data = scaler.fit_transform(all_data["AAPL"])
    X, y = lstm_model.create_sequences(normalized_data)
```

```
    X_train, X_test, y_train, y_test = lstm_model.split_data(X,y)

    # Train the model
    model.fit(X_train, y_train, epochs=3, batch_size=4, verbose=0)

    # Evaluate the model on the validation set
    # You might want to split your data into a validation set beforehand.
    val_loss = model.evaluate(X_train, y_train, verbose=0)

    # Store the evaluation result
    evaluation_results[loss_name] = val_loss

    # Check if this loss function is the best so far
    if val_loss < best_loss_value:
        best_loss_value = val_loss
        best_loss_function = loss_name

print("Evaluation Results:")
for loss_name, val_loss in evaluation_results.items():
    print(f"{loss_name}: {val_loss}")

print(f"The best loss function is: {best_loss_function} with value:␣
 ↪{best_loss_value}")
```

```
loss_names RMSE
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.

2023-09-18 21:48:18.863845: I metal_plugin/src/device/metal_device.cc:1154]
Metal device set to: Apple M1
2023-09-18 21:48:18.863874: I metal_plugin/src/device/metal_device.cc:296]
systemMemory: 16.00 GB
2023-09-18 21:48:18.863880: I metal_plugin/src/device/metal_device.cc:313]
maxCacheSize: 5.33 GB
2023-09-18 21:48:18.863919: I
tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:303]
Could not identify NUMA node of platform GPU ID 0, defaulting to 0. Your kernel
may not have been built with NUMA support.
2023-09-18 21:48:18.863943: I
tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:269]
Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 0
MB memory) -> physical PluggableDevice (device: 0, name: METAL, pci bus id:
<undefined>)
2023-09-18 21:48:19.689621: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114]
Plugin optimizer for device_type GPU is enabled.
2023-09-18 21:54:07.353245: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114]
```

```
Plugin optimizer for device_type GPU is enabled.

loss_names MAE
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.

2023-09-18 21:54:09.225410: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114]
Plugin optimizer for device_type GPU is enabled.
2023-09-18 21:59:55.298010: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114]
Plugin optimizer for device_type GPU is enabled.

loss_names MAPE
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.

2023-09-18 21:59:56.948180: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114]
Plugin optimizer for device_type GPU is enabled.
2023-09-18 22:54:21.932183: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114]
Plugin optimizer for device_type GPU is enabled.

loss_names Huber Loss
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.

2023-09-18 22:54:23.927739: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114]
Plugin optimizer for device_type GPU is enabled.
2023-09-18 23:02:32.229851: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114]
Plugin optimizer for device_type GPU is enabled.

Evaluation Results:
RMSE: 0.014287491329014301
MAE: 0.005887804087251425
MAPE: 117.6821060180664
Huber Loss: 0.00032324157655239105
The best loss function is: Huber Loss with value: 0.00032324157655239105
```