Kaggle M5 Forecasting - Uncertainty 4th Place Solution

TOC

1. Contents
2. System Requirements
3. Instructions
4. Create Your Own Model

1. Contents

directory or filename	contents
models/	binary files of models and states of random number generator in order to reproduce the solution
src/	source files and hyper parameter config files
LICENSE	license of all contents in this repository (MIT license)
ModelSummary.md	document about model
README.md	this file
directory_structure.txt	directory structure of this repository
requirements.txt	python package list
*.sh	shell scripts to download, preprocess, reproduce and train

2. System Requirements

The following specifications were used to create the original solution.

Hardware:

- Microsoft Azure "NC12_Promo" Virtual Machine
    - 12 core CPU
    - 110 GB RAM
    - 2 x NVIDIA Tesla K80 GPU

This machine has 2 GPUs, but only 1 GPU was used.

Software:

- Microsoft Azure "Data Science Virtual Machine - Ubuntu 18.04"
    - Ubuntu 18.04 LTS
    - NVIDIA CUDA 10.2
    - Anaconda

3. Instructions

Here are instructions to reproduce the original solution by the trained models.

3-1. Setup the Kaggle API

Download your Kaggle API Key, and save it to `~/.kaggle/kaggle.json` .

You need to join not only "Kaggle M5 Forecasting - Uncertainty", but also "Kaggle M5 Forecasting - **Accuracy**" to download both competitions datasets.

3-2. Create Python Virtual Environment

Execute following commands to create python virtual environment by anaconda.

```
conda create --name m5 python=3.8
conda activate m5
pip install -r requirements.txt
```

3-3. Download Data

Execute following command to download both Accuracy and Uncertainty datasets. You can also download Federal Holidays USA 1966-2020 dataset, which I used as additional datasets by executing this program.

```
./run1_download.sh
```

3-4. Preprocess

Execute following command to preprocess. It could take several hours.

```
./run2_preprocess.sh
```

3-5. Reproduce

Execute following command to reproduce the solution by the trained models. It could take about 15 minutes.

```
./run3_reproduce.sh
```

Then, submission files are created at `./submissions` directory. A file whose name begins with `acc` is an Accuracy submission file, and a file whose name begins with `unc` is an Uncertainty submission file.

4. Create Your Own Model

You can create your own model by executing following command,

```
./train.sh
```

or executing following command.

```
cd src
python train.py trainer=[agg_cv/agg_submit/each_cv/each_submit]
```

These are hyper parameter config files.

- ./src/config.yaml
- ./src/trainer/agg_cv.yaml
- ./src/trainer/agg_submit.yaml
- ./src/trainer/each_cv.yaml
- ./src/trainer/each_submit.yaml

`agg_submit.yaml` and `each_submit.yaml` are hyper parameters of the original solution. You can change these settings and try to train models with changed settings.

`agg_cv.yaml` and `each_cv.yaml` are example settings to execute local CV.

When you want to create submission files with your own models, please read comments in `train.sh`.

4-1. Hyper Parameters

parameter	description
experiment	MLflow experiment name.
seed	random seed.
submit	if True, train models and create temporal submission DataFrames. if False, run CV.
each	if True, create "Each Model". if False, create "Agg Model". see `ModelSummary.md` about these models.
useval	if True, use Validation Phase Data (2016/04/25 - 2016/05/22) when training models.
num_workers	`num_workers` parameter of `torch.utils.data.DataLoader` .
batch_size	mini batch size.
max_epochs	maximum epochs to train.
min_epochs	minimum epochs to train.
patience	patience for early stopping. If 0, early stopping is disabled.
val_check	`val_check_interval` parameter of `pytorch_lightning.Trainer`
over_sample	if True, data in 2015 is sampled twice and data in 2016 is sampled 4 times in 1 epoch to train.
objective	loss function. [train_loss / train_wloss / train_rmse / train_wrmse]
n_hidden	hidden unit size of LSTM layer.
use_prices	if True, use price features.
dist	probability distribution. [Normal / StudentT / NegativeBinomial]
df	degree of freedom (only StudentT).
dropout	dropout probability.
optimizer	optimizer. [Adam / SGD]
lr	learning rate.
weight_decay	weight_decay parameter of optimizer.

scheduler	scheduler. [None / CosineAnnealingLR]
T_max	T_max parameter of CosineAnnealingLR scheduler.
use_te	if True, use target encoding feature.

End of document