User Guide

August 31, 2022

All code, pre-trained models, and primary experimental data can be found on our github: `https://github.com/359444284/Cryoto_predictor.git`.

1 Hardware Requirement

Because this experiment uses a very large dataset, we used an RTX 3090 GPU and an i7-11700K CPU for training. We will list the minimum requirements for the experiment below:

- Space to be used for different data sets:
  - BTC-50: 92 GB
  - BTC-10: 17 GB
  - ETH-14: 23 GB
  - BTc-14: 25 GB
  - FI-2010: 694MB
- Time consumption in our setting: 2 min per epoch for cryptocurrency data and 20s for FI-2010

2 Data Preparation

Because the datasets are so large, we don't have a common cloud space to store them all. BTC_50 and FI-2010 can be download form our google drive directly: `https://drive.google.com/drive/folders/1GnSQhcWIKwVtmvpsDAPR82f5tUBfFIvn?usp=sharing` Otherwise, please give me an email : 359444284a@gmail.com or find me in the github issue section:

For the data setting, please following the guideline in our github: Crypto_predictor/code/data/guideline.

3 Config

For the pip requirement, we give We have given a requirements.txt in our repository.

3.1 Experiment Setting

All model structure can be find in Crypto_predictor/codes/model_card, while the structure of LSTM and DeepLOB while the structure of LSTM and DeepLOB can only be changed in the model file. On the other hand, all other hyper-parameter can be set up in Crypto_predictor/codes/config.py. The following list shows the parameters that need to be changed when using different configurations. In addition, all options are written in the common of the code.

- Common Parameter:
  - name_dataset: select the name of the dataset to be used.

- backbone: Select the model to be used.
- select_fun: Whether to use deep learning feature selection methods.
- batch_size.
- lr: learning rate.
- Normalize: data normalization strategy.
- LC_window: window size for LC-Norm.
- loss_fun: Choose between Cross Entropy(CE) and Dice Loss(DSC)
- DSC_alpha: Alpha for DSC Loss
- lockback_window: input size.

- Specially for FI-2010 dataset.
  - k: horizon for prediction.
- Specially for our dataset.
  - split_data: splitting the data set by days into a train-val set and a test set.
  - train_ratio :How much percentage of the train-val set data is used as the train set.
  - feature_type: Select the feature selection mode to be used: 'all' – use all features, 'list' – use a list of feature subset, 'selected' – use selected features.
  - feature_list: if feature_type = 'list', select the feature subset to be used.
  - subset_name: if feature_type = 'selected', select the feature subset selected by different method.
  - trade_fee: trading fee for backtesting.
  - trade_delay: trading delay for backtesting.
  - signal_threshold: how many signals using for backtesting.
  - plot_forecast: whether to draw the results of the model prediction.
  - backtesting: whether to perform backtesting for the test set
- Model setting. (the parameter of LSTM and DeepLOB should modify in model_card directly)
  - tran_emb_dim: hidden dimension for transformer
  - tran_layer: number of encoder layer for transformer
  - tran_fc_dim: number of hidden size for FFNN in transformer
  - tran_num_head: number of head using in transformer attention layer
  - tran_drop: dropout rate for transformer
  - use_channel_att: use vanilla transformer or FanLOB

Furthermore, We can find the location in the figure (1) at config.py to modify the label and horizon to be used.

```
# choose label and horizontal
# Label Equation 1: 160: 20 161: 50 162: 70 163: 100
# Label Equation 2: 164: 20 165: 50 166: 70 167: 100
feature_index.extend([161])
```

Figure 1: Selection the Label To Be Used

4 Running Experiments

You can run python main.py directly to train and validate the model after all configurations are done.

Alternatively, you can run python predict.py to validate the data using the pre-trained model.All pre-trained models can be found in the /codes/checkpoints folder. In predict.py, you can find model.load() which is used to read the pre-trained model, and it loads the freshly trained model by default. If you need to load a custom pre-trained model, please use model.load(name=model_path), where model_path is 'checkpoints/file_name'.

Moreover, we can use python cross_validdation.py to cross-validate the dataset. You can set the folder where you want to save and load the model by modifying the 'save_root'. You can also decide whether or not to only perform evaluations by removing the following line of code.

```
trainer.train_epoch(config, model, optimizer, device, loss_fn, train_loader,
                    valid_loader, epochs=epoch, scheduler=scheduler, save_path=s
```

Figure 2: Turn to evaluation model for cross validation process