Time Series Ensembling with Optimized Learning Schedule (最適化された学習スケジュールを伴う時系列アンサンブル)

Dept. of Mathematical Informatics 48176232 Wang Zeyu
Supervisor Sei Tomonari

1 Introduction

Model ensembling is a powerful technique to increase accuracy on a variety of machine learning tasks by combining several base prediction models. The current time series prediction model ensembling approaches are mostly blending or simple out-of-sample stacking, i.e. an initial part of the base models' output is used for fitting meta-model and left part is held out for testing.

However for time series forecasting, the performance of base models could be inconsistent via time which make the standard ensemble approach ineffective.

There are two main contributions in this study: (1) for inconsistent base models, we discuss the importance of selecting a proper "look back"–how long we look back to fit the meta-model for stacking, (2) we propose a novel stacking framework that can optimize the learning schedule of the stacking generalization.

The proposed framework is validated in two open-source data sets and the result showed the proposed stacking framework is capable to detect the potential seasonality and possible to improve the performance even with a smaller training set compared to simple out-of-sample stacking. This study shows the potential of optimizing the learn schedule for time series prediction model ensmebling and gives the comparison with standard method.

2 Time series stacking

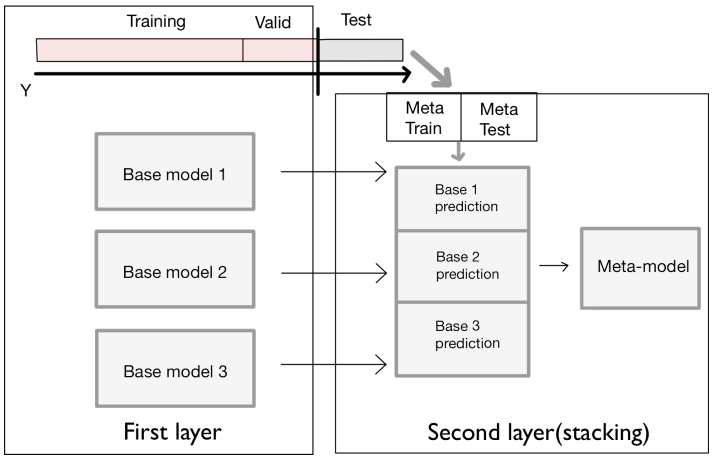

Fig. 1. Out-of-sample stacking in type of sequential data (Y-axis is order or time)

As an important type of ensemble learning, stacking was first introduced by [2] in 1992. The paper introduced the concept of a meta model being trained on the outputs of various base models with the scope of minimizing the generalization error of a target variable.

The stacking generalization can be applied to the sequential data, the standard out-of-sample stacking in type of sequential data is illustrated in Figure 1.

On the other hand, time series stacking looks similar to time series prediction: both take sequential data as training set for predicting the future value. However, the original purpose of the stacking is "to combine the $\{G_j\}$ (the base models) rather than choose one amongst them" by Wolpert in [2]. The input and the structure of the meta-model is relatively simple. For example, in order to predict y_t, AR(K) model gives the estimation

$$\hat{y}_t = AR([y_{(t-1)}, ..., y_{(t-K)}])$$

with time delay embedding dimension K, and the meta-model gives

$$\hat{y}_t = S((y_t^1, y_t^2, y_t^3))$$

where (y_t^1, y_t^2, y_t^3) is the outputs from three base models at time t only. See table 1 for detailed comparison.

	Time series prediction	Time series stacking
embedding dimension	select by information criteria or algorithms	usually the same time stamp (= 1)
optional models	RNN, CNN, AR model, VAR model…	logistic regression, KNN, NN, GBM…
input	original time series	output from other models, extra features(option)
train cost	depend on models, data size and embedding dim	relatively low, the meta-model is less complex

Table 1. Difference between time series prediction and time series stacking

3 Motivation and problem setting

As we mentioned, for time series forecasting, the performance of base models could be inconsistent via time which makes the standard ensemble approach ineffective. In our paper we designed an experiment to discuss the importance of tuning "look back" – how long we look back to fit the meta-model of stacking.

We first generate the sequence y_t which represents the target sequence to predict. We set $t = 1, ..., 1000$ and y_t could be 1 or 0 with equal possibility

$$P(y_t = 1) = P(y_t = 0) = 0.5, \quad t = 1, ..., 1000. \quad (1)$$

And we also simulated three "classifiers" $x^{(1)}, x^{(2)}$ and $x^{(3)}$ that attempt to predict y_t with inconsistent performance.

The classifier $x^{(1)}$ has constant accuracy that equal to 0.7 which means it has 70% probability to give correct prediction in each time t. The classifier $x^{(2)}$ has 50% at the beginning where $t = 0$ and linearly increases to 1 at $t = 1000$.And the classifier $x^{(3)}$ is the opposite, the accuracy decreases linearly from 1 to 0.5.

In this experiment we applied stacking by both Logistic Regression and LightGBM. The variable in this experiment is the length of the train set (lookback) and we held out last 30% data as test set.

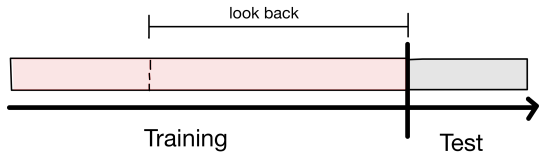

Fig. 2. Lookback represent the length of the data used for training

In this experiment we found with different "lookback", the accuracy can be varied from 60% to 90%.

4 Proposed method

From last experiment we notice the importance of tuning the size of training data. As a generalization, we propose a stacking framework that optimizes the learning schedule of the meta-model.

As shown in fig 3, in our proposed stacking framework, we not train the meta-model once as the standard out-of-sample stacking, the meta-model is keep to be trained until cover the whole test set.

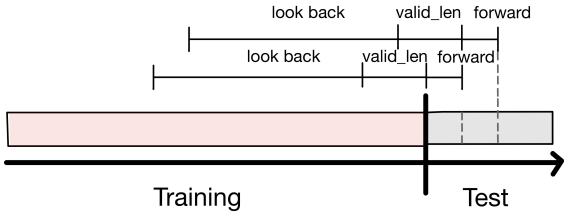

Fig. 3. In proposed stacking framework, we train meta-model multiple times to cover the test set.

Three parameters "lookback", "valid_len" and "forward" respectively represent "how long we look back to fit the meta-model", "how long should the valid set be" and "how long the model will be used until next retaining". All three parameters are tuned by non-gradient searching algorithms according to the validation score.

We also compared standard out-of-sample stacking with the proposed framework in our paper. The dataset is historical stock prices (2006-01-01 to 2018-01-01) of IBM accessible at Kaggle [1]. For test two stacking approaches, we trained four RNN base models with different structures and different embedding dimensions.

The meta-model for both proposed approach and standard stacking are trained by LightGBM. For the proposed approach we apply Bayesian Optimization for 300 round hyper-parameter tuning experiments. Fig 4 shows the final result. In the experiment the performance of our proposed approach has smaller average MSE and smaller variation (after selection) compare to the standard out-of-sample stacking.

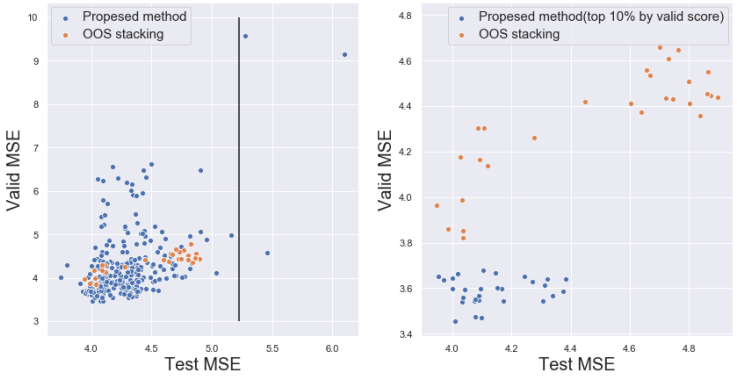

Fig. 4. Comparison of two models. Each point is an experiment with different parameters.

5 Conclusions

In this thesis we want to answer two main questions: for time series emsbeling, is the larger training data always better, and is it possible to optimize the learning schedule.

To answer the first question, we design the numerical experiment with synthesized series. It is nature to find that stacking is hard to "learn more" since the inconsistent performance of the first level models.

And for the second question, we proposed a stacking framework that optimizes the learning schedule of training meta-model and design the experiment to testify it. The practical result shows the performance of our proposed approach has better average score and smaller variation compare to the standard out-of-sample stacking.

Bibliography

[1] DJIA 30 Stock Time Series. https://www.kaggle.com/szrlee/stock-time-series-20050101-to-20171231

[2] Wolpert, D. H. (1992). Stacked generalization. Neural Networks, 5(2), 241-259.