On combining forecasting methods using time series features

Abstract

It is well known that ensemble approaches produce improvements over single methods in statistical learning. Nevertheless, when calculating predictions over a large dataset, computation time for the whole ensemble can be prohibitive, so individual model selection becomes the preferred approach. We present a method for combining forecasting models by posing it as a classification problem using features extracted from the time series. Unlike regular classification problems, we minimize the average forecast error of the selected method rather than the classification error. Not only does this address the aim of accurate forecasting, it also provides measures of relative method accuracy across the time series, and relative difficulty across time series. In contrast, a classic classification approach would give the same importance to all series and methods. The presented classifier is compared with state-of-the-art approaches to forecasting and time series classification. The results show an improvement of error over alternative approaches. These experiments allow us to show the relevance of both the feature set and the proposed optimization approach to several collections of time series. The scalability of the approach allows it to be applied to forecasting a large collection of time series. It can also be efficiently trained to tailor specific domains or datasets.

Keywords: FFORMA (Feature-based FORecast-model Averaging), FFORMS (Feature-based FORecast-model Selection), Time series features, Forecast combination, XGBoost, M4 Competition

1 Introduction

There are essentially two general approaches to forecast a time series: i) use of single model and ii) combination forecast or forecast model averaging. There is a growing consensus that the combination forecast is a way of improving forecast accuracy. Empirical work often provides evidence that the combination of forecasts resulting from several candidate forecasting models are often superior to their individual forecasts. Combining forecasts across different models is considered as a way of reducing the risk of selecting an inappropriate model. However, the

main challenge in forecast combination is the selection of appropriate set of weights.

Granger (1969), was the first to forward the idea of "the combination of forecasts". Since then many approaches have been proposed to derive weights for forecast combination (Timmerman, 2006). Simple averages(ref_clemen1989), regression-based approaches, ... are name to few.

Recently, a few researchers have explored the use of time series features in selecting the most appropriate forecasting method. However, use of time series features to derive weights for forecast combination has been rarely addressed in the literature. In this paper we propose a general framework to obtain weights for forecast combination based on features computed from the time series. We call this framework FFORMA(Feature-based FORecast Model Averaging). The proposed FFORMA framework has been used over the course of M4 competition and placed in second in forecast accuracy in both point forecasts and prediction intervals.

Highligh simplicity of the forecsint methods and no fine tuning (long series only last part, models per type weekly, daily,etc..) No exogenous information, such as macroeconimic, finance, tourism

** EXAMPLE **

- **Contribution LIST / Refutable** *

The rest of the paper is organized as follows. Section 3 presents the methodology underlying the FFORMA framework followed by Section 4 that describes the application of the framework to M4 competition, with specific decisions as the set of features, individual forecasting methods, and learning algortithms. *The method to produce the prediction intervals for the M4 competitionis also explained in this section.*

Finally Section 5 presents the conclusions and future work.

2 Methodology

This section describes the rational behind the proposed framework. In the proposed framework we use meta-learning approach to derive weights for forecast combination. Figure 1 presents the Pseudo code of the proposed framework. This approach has been influenced by the work of Talagala, Hyndman & Athanasopoulos (2018).

2.1 Overview / Intuition

The general objective of meta-learning is to learn to "combine" individidual forecasting methods to minimize forecasting error. We start from a dataset of time series, which we call training set, which may include the time series we want to forecast, and a set of forecasting methods. The forecasting errors of each forecasting method on each time series are calculated, either using the true future values of the series if available or a temporal holdout version of the series.

The meta-learning model is then fitted on the training set, essentially trying to find how to combine the methods in our pool to minimize the forecasting error on each series. The input for training the meta-learning model are features extracted from the series (e.g. length, autocorrelation coefficients,...) and the error that is produced by each of the forecasting methods. Once the meta-learning model is trained, the inductive step consist of assuming the series we want to forecast comes from a "generating process" similar to the one that generated the training set.

One common meta-learning approach is to learn to select the best method for each series, the one that produces the least forecasting error. This transforms the problem into a traditional classification problem and allows the use of existing classifcation algorithms. However, this simplification removes relevant information which may be useful in the metalearning process, such as which methods producing similar error to the best on for a series, or series that produce larger errors than others. We do not apply this simplification and take into account the exact error that each method produces in each series, instead of just considering which one produces the minimum error. We then pose the problem as finding a function that assigns probabilities to each forecasting method for each series, with the goal of minimizing the expected error that is produced if the methods were picked at random following these probabilties. It can be seen that in the extreme, our approach can be easily transformed to a classification exercise if we pick the method with the largest probability, and that if we allow a sufficiently rich hyphotesis space to fit freely, we would end up assigning probability one to the methods with the least amount of error.

The meta-learning process can therefore be summarized as:

- X_i the individual series on the training set.
- e_{ij} the error the individual forecasting method j makes on series i.
- x_i the feature vector extracted from series X_i.
- f is a function belonging to the hypothesis space H that maps features to probabilities of

the j forecasting methods.

$$argmin_{f \in H} \sum_i \sum_j f(x_i)_j e_{ij}$$

Given a new time series, a trained meta-learning algorithm produces probabilities for each of the forecasting methods considered in the pool. The final forecast can be computed from these probabilities in two-ways:

1. A weighted average of the individual forecast methods. This tends to produce the best forecast on average, but requires computing the forecasts of all methods in out pool, which may be time consuming.
2. Selecting the method with the largest probability. This prevents the need to calculate all the forecast of the individual methods in our pool, but produces less acurate forecasts on average.

2.2 Algorithmic description

The FFORMA framework has four main components:

1. The set of forecasting methods, pool of methods.
2. The set of features to be extracted from the time series
3. A training set of time series, the reference set.
4. The forecasting error measure, such as mean squared error, .

The FFORMA framework works in two phases: an **offline** phase and an **online** phase.

In the offline phase a meta-learner is trained using a diverse collection of time series, called the reference set. Each time series in the reference set is divided in to training period and test period. Features are extracted based on the training period of each time series. These features are one of the **two kinds** of inputs to the meta-learner. The second input is the forecasting error produced by each of the methods in the poo for each series. This error is calculated comparing the forecast of the methods fitted in the training period to the test period, using a error measure.

The algorithm produces weights for each forecasting method. These weights can also be interpreted as the probability of each method being best. These weights can be used either to

Algorithm 1 The FFORMA framework - Forecast combination based on meta-learning.

Offline phase - train the learning model

Given:

$O = \{X_1, X_2, \ldots, X_n\}$: the collection of N observed time series.

P : Set of K forecasting algorithms such as ARIMA, ETS, SNAIVE, etc.

F : the set of functions to calculate time series features.

E : A forecasting error measure such as Mean Squared Error.

Output:

FFORMA meta-learner:

A function from the extracted features to a set of K probabilities, one for each method in P.

Prepare the meta-data

For $j = 1$ to N:

1: Split X_j into a training period and test period.
2: Calculate the set of features for the training period by applying F.
3: Fit P models to the training period.
4: Calculate forecasts for the test period from each model.
5: Calculate forecast error measure E over the test period for all models in P.
6: Meta-data: input features x_i (step 4), output errors: e_i (step 5).

Train the meta-learner

7: Train a learning model based on the meta-data and errors, by minimizing:

$$argmin_f \sum_{i=1}^{n} \sum_{k=1}^{K} f(x_i)_k e_k$$

8: meta-learner.

Online phase - forecast a new time series

Given:

FFORMA classifier from step 8 .

Output:

Forecast for the new time series x_{new}.

9: For x_{new} calculate features F.
10: From the features, use the meta-learner to produce w the vector of probabilities.
11: Compute the individual forecasts of the methods in P
12: Compute the final forecast by weighted average using w and the individual forecasts.

select the "best" forecasting method for each series or to combine the forecasts using weighted linear combination. Note that the accuracy of the FFORMA meta-learner depends on three main factors:

1. Forecasting methods used in the pool
2. The set of time series features we considered
3. The collection of time series used to train the classifier.

Section 3 provides a more detailed description of application of the FFORMA framework over

the course of M4 competition. The proposed framework is closely related to the previous work by ref_prudencio which they use machine learning techniques to define weights for the linear combination of forecasts.

3 FFORMA framework: Application to M4 Competition

3.1 Data preprocessing

The M4 competition database consists of 100,000 real-world time series of yearly, quarterly, monthly, weekly, daily and hourly data. The frequency for yearly, quarterly, monthly and weekly data are considered as 1, 4, 12 and 52 respectively. [Q: Daily and Hourly series, what are the frequencies considered?]. We used xxxx time series out of 100000 to train a meta-learner and rest is used to evaluate the proposed framework. In addition to the time series provided in the M4 competition database we used the time series of M1 and M3 competitions (Makridakis & Hibon 2000) to the reference set. Each time series in the reference set is split into training period and test period. The length of test period of each time series was set as according to the rules of M4 competition, 6 for yearly data, 8 for quarterly, 18 for monthly, 13 for weekly, 14 for daily and 48 for hourly.

3.2 Time series features

Table xxx provides a detailed description of features used in this experiment.

[include table]

These features have been previously used by Talagala, Hyndman & Athanasopoulos (2018) and Hyndman, Wang & Laptev (2015). A detailed description of these features are provided in Talagala, Hyndman & Athanasopoulos (2018). All the features are implemented in `tsfeatures` R package by xxx.

Question: calculation of features for time series with multiple seasonality, and short time series?

3.3 Forecasting methods

We considered nine forecasting algorithms implemented in the forecast package in R. They are(the specific R calls for fitting the models are given): i) automated ARIMA algorithm (`auto.arima`), ii) automated ETS algorithm (`ets`), iii) feed-forward neural network with a single hidden layer is fitted to the lags. The number of lags is automatically selected (`nnetar`)

iv) random walk with drift (`rwf` with drift=TRUE), v) TBATS model (`tbats`), vi) Theta method forecast (`thetaf`), vii) naive forecasts (`naive`) viii) STLM-AR Seasonal and Trend decomposition using Loess with AR modeling of the seasonally adjusted series(`stlm` with modelfunction `ar`), ix) seasonal naive forecasts (`snaive`). In the case of an error when fitting the series (e.g. a series is constant), the SNAIVE forecast method is used instead.

Question: i) Calculation of ets models to daily, hourly and weekly series

3.4 Metal-learning model

We choose the gradient tree boosting of **xgboost** as the underlying implementation of the learning model for the following reasons:

1. Ability to customize the model to fit our objective function
2. Computational efficiency: The size of the M4 dataset requires a efficient model
3. Good performance in structure based problems

3.4.1 The xgboost objetive function

We can consider two kinds of inputs to our learning model, the features and the errors of the individual methods in the pool. The xgboost algorithm produces numeric values from the features, one for each forecasting method in our pool. We apply the softmax transform to these values and the compute our objective function (the loss function) using these probabilities and the errors produced by the methods in our pool.

xgboost fits the model by gradient boosting, for which requires the gradient and hessian of the objective function. In our case, the gradient is the direct gradient of the objective, but the *correct* hessian is prone to numerical problems that need to be fixed for xgboost to converge. This is a relative common problem and one simple fix is to use an upper bound of the hessian by clampling small values to a larger one. In our case, we compute an alternate upper bound of the hessian by removing some terms from the *correct* hessian . Although both alternatives converge, our approach works much fastes, requiring less boosting steps to converge. This not only increases the computational efficiency, in our scenario it also generalizes better due to a less complex set of trees produced in the final solution.

The functions involved in computing the objective are the following:

- $y(x)$ is the output of the xgboost algorithm

- $p_j = \frac{e^{y(x)_j}}{\sum_k e^{y(x)_k}}$ is the transformation to probabilities by applying the softmax transform.
- $L = \sum p_j e_j$ is the loss of the function p.
- $G_j = \frac{\partial L}{\partial p_j} = p_j(e_j - \sum_k e_k p_k)$

$$H_j = \frac{\partial G_j}{\partial p_j} \approx \hat{H}_j = p_j(e_j(1 - p_j) - G_j)$$

3.4.2 Hyperparameters

The results of xgboost are particularly dependent on its hyperparameters such as learning rate, number of boosting steps, maximum complexity allowed for the trees or subsampling sizes. In our case we limit the hyperparameter search space based on some initial results and rules of thumb and explore it using bayesian optimization, measuring performance on a 10% holdout version of the training set. We pick the most simple hyperparameter set from the top solutions of the exploration.

3.5 Generate point forecasts

3.6 Prediction Intervals

The M4 Competition also featured a subcompetition over the accuracy of prediction intervals. For this part of the competition, we used a different approach that for the point forecasts. In order to compute the intervals we used the point forecast produced by our meta-learner as the centre of the interval and computed the 95% bounds of the interval by a linear combination of the bounds of three forecasting methods: Thetaf, SNAIVE and NAIVE methods. The coefficients for the linear combination were calculated in a data driven way. The procedure is as follows:

1. For the training period each series in the dataset:
    1. Compute the point forecast of the meta-learning.
    2. Compute the 95% *predicion radius* for the thetaf, snaive and naive, this is the difference between the 95% upper bound and the point forecast for each horizon.
2. For each horizon in the dataset:
    1. Find the coefficients that minimize the MSIS error of the interval with the meta-learning point forecast as center and a linear combination of the radiuses of thetaf, snaive and naive as radius. The coefficients are calculated by gradient descent.

This procedure produces at set of three coefficients for each of the prediction horizons in the M4 dataset and these coefficients will be the same independently of the series we want to forecast. These coefficients are also not restricted to be probabilities, the optimization is unrestricted.

4 Results (Do we need this?)

Fotios email: We would like to ask that this paper focuses on the clear description of the method (and possibly include flowcharts, pseudocode, etc) without any empirical evidence (that would be part of the main M4-competition paper).

Maybe we can show here results about the metalearning, how weights evolve with series length. What is the difference between weightedaveraging and selection in our model? What happens if we remove individual forecasting methods? How robust are we against this? wHich features are more important?

5 Discussion and Conclusions

Fotios email: We would like, though, to see a short section that discusses the reasons behind the good performance of your method.

5.1 Reasons behind the performance

1. As opposed to individual methods in our pool, it is being trained to the specific error in the M4 Competition. The methods in our pool do not minimize the OWA error, but the squared loss, while we generated probabilities based on the OWA error.
2. As opposed to individual selection methods, it exploits domain bias. One method could be discarded in a series when the error is low, because on average it performs bad in the rest of the dataset, while individual methods would pick it up. "
3. Exploiting dependencies between time series. In the M4 competition, some time series are similar, which can be hypothesized as macroeconomic indicators of neighbouring countries, etc. This effect is explited in the metalearning model and the series would produce similar features, and the errors that the model minimizes are more accurate.
4. AS opossed to traditional classification meta-learning approaches: Take the individual errors into account. This works two ways, capturing similarity the results in methods to keep the model simple (. Capturing differences in time series to keep the model simple (difficult do not end up affecting the model much).

5. Compared to a naive averaging of individual methods, it exploits easily discardable methods. Many ensemble approaches work by averaging the output of individual methods. This is especially when the number of individual forecasting methods grows and allows us to include methods with radically different assumptions that would not be appropiate to average. e.g."for really short time series you really cant do better than naive, others just overfit"
6. As opposed to just holdout crossvalidation, our method performs better. If we allow the method to overfit, i will reproduce the results of the holddout crossvaliation error, but in our case it improves it.

References

Hyndman, RJ, E Wang & N Laptev (2015). Large-scale unusual time series detection. In: *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*. IEEE, pp.1616–1619.

Makridakis, S & M Hibon (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting* **16**(4), 451–476.

Talagala, TS, RJ Hyndman & G Athanasopoulos (2018). Meta-learning how to forecast time series. *Technical Report 6/18, Monash University.*