MONASH
BUSINESS
SCHOOL

ISSN 1440-771X

Department of Econometrics and Business Statistics

http://business.monash.edu/econometrics-and-business-statistics/research/publications

FFORMA: Feature-based Forecast Model Averaging

Pablo Montero-Manso, George Athanasopoulos,
Rob J Hyndman, Thiyanga S Talagala

October 2018

Working Paper 19/18

FFORMA: Feature-based Forecast Model Averaging

Pablo Montero-Manso
Department of Mathematics, University of A Coruña, Spain
Email: p.montero.manso@udc.es
Corresponding author

George Athanasopoulos
Department of Econometrics and Business Statistics, Monash University, Australia
Email: george.athanasopoulos@monash.edu

Rob J Hyndman
Department of Econometrics and Business Statistics, Monash University, Australia
Email: rob.hyndman@monash.edu

Thiyanga S Talagala
Department of Econometrics and Business Statistics, Monash University, Australia
Email: thiyanga.talagala@monash.edu

12 October 2018

JEL classification: C10,C14,C22

FFORMA: Feature-based Forecast Model Averaging

Abstract

We propose an automated method for obtaining weighted forecast combinations using time series features. The proposed approach involves two phases. First, we use a collection of time series to train a meta-model to assign weights to various possible forecasting methods with the goal of minimizing the average forecasting loss obtained from a weighted forecast combination. The inputs to the meta-model are features extracted from each series. In the second phase, we forecast new series using a weighted forecast combination where the weights are obtained from our previously trained meta-model. Our method outperforms a simple forecast combination, and outperforms all of the most popular individual methods in the time series forecasting literature. The approach achieved second position in the M4 competition.

Keywords: Time series features, Forecast combination, XGBoost, M4 competition, Meta-learning

1 Introduction

There are essentially two general approaches for forecasting a time series: (i) generating forecasts from a single model; and (ii) combining forecasts from many models (forecast model averaging). There has been a vast literature on the latter motivated by the seminal work of Bates & Granger (1969) and followed by a plethora of empirical applications showing that combination forecasts are often superior to their individual counterparts (see, Clemen 1989; Timmermann 2006, for example). Combining forecasts using a weighted average is considered a successful way of hedging against the risk of selecting a misspecified model. A major challenge is in selecting an appropriate set of weights, and many attempts to do this have been worse than simply using equal weights — something that has become known as the "forecast combination puzzle" (see for example, Smith & Wallis 2009). We address the problem of selecting the weights by using a meta-learning algorithm based on time series features.

There have been several previous attempts to use time series features combined with meta-learning for forecasting (see for example Prudêncio & Ludermir 2004; Lemke & Gabrys 2010; Kück, Crone & Freitag 2016; and Kang, Hyndman & Smith-Miles 2017). Recently, Talagala, Hyndman & Athanasopoulos (2018) proposed the FFORMS (Feature-based FORecast Model Selection) framework that uses time series features combined with meta-learning for forecast-model selection. That is, features are used to select a single forecasting model. In this paper, we build on this framework by using meta-learning to select the weights for a weighted forecast combination. All candidate forecasting methods are applied, and the weights to be used in combining them are chosen based on the features of each time series. We call this framework FFORMA (Feature-based FORecast Model Averaging). FFORMA resulted in the second most accurate point forecasts and prediction intervals amongst all competitors in the M4 competition.

The rest of the paper is organized as follows. In Section 2 we describe the FFORMA framework in a general sense. Section 3 gives the details of our implementation of FFORMA in the M4 competition for generating both point and interval forecasts. This includes the required preprocessing steps, the set of features and forecast methods, as well as the specific implementation of the meta-learning model. We show empirical evidence on the performance of the approach in Section 4 by quantifying the difference between our proposed learning model and a traditional classifier approach. Section 4 also provides some final remarks and conclusions.

2 Methodology

2.1 Intuition and overview of FFORMA

The objective of our meta-learning approach is to derive a set of weights to combine forecasts generated from a *pool of methods* (e.g., naïve, exponential smoothing, ARIMA, etc.). The FFORMA framework requires a set of time series we refer to as the *reference set*. Each time series in the reference set is divided into a training period and a test period. From the training period a set of *time series features* are calculated (e.g., length of time series, strength of trend, autocorrelations, etc.). These form the inputs to the meta-learning model. Each method in the pool is fitted to the training period, forecasts are generated over the test period, and *forecast errors* (the difference between actual and forecast values) are computed. From these, a summary forecast loss measure from a weighted combination forecast can be computed for any given set of weights.

The meta-learning model learns to produce weights for all methods in the pool, as a function of the features of the series to be forecasted, by minimizing this summary forecast loss measure. Once the model is trained, weights can be produced for a new series for which forecasts are required. It is assumed that the new series comes from a *generating process* that is similar to some of those that form the reference set.

A common meta-learning approach is to select the best method in the pool of methods for each series; i.e., the method that produces the smallest forecast loss. This approach transforms the problem into a traditional classification problem by setting the individual forecasting methods as the classes and the best method as the target class for each time series. However, there may be other methods that produce similar forecast errors to the best method, so the specific class chosen is less important than the forecast error resulting from each method. Further, some time series are more difficult to forecast than others, and hence have more impact on the total forecast error. This information is lost if the problem is treated as classification.

Consequently, we do not train our meta-learning algorithm using a classification approach. Instead, we pose the problem as finding a function that assigns *weights* to each forecasting method, with the objective of minimizing the expected loss that would have been produced if the methods were picked at random using these weights as probabilities. These are the weights in our weighted forecast combination. This approach is more general than classification, and can be thought of as classification with *per class weights* (the forecast errors) that vary per instance, combined with *per instance weights* that assign more importance to some series.

2.2 Algorithmic description

The operation of the FFORMA framework comprises two phases: (1) the offline phase, in which we train a meta-learner; and (2) the online phase, in which we use the pre-trained meta-learner

to identify forecast combination weights for a new series. Algorithm 1 presents the pseudo-code of the proposed framework.

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

OFFLINE PHASE: TRAIN THE LEARNING MODEL

Inputs:

$\{x_1, x_2, \ldots, x_N\}$: N observed time series forming the reference set.
F: a set of functions for calculating time series features.
M: a set of forecasting methods in the pool, e.g., naïve, ETS, ARIMA, etc.

Output:

FFORMA meta-learner: A function from the extracted features to a set of M weights, one for each forecasting method.

Prepare the meta-data

1: **for** $n = 1$ to N: **do**
2: Split x_n into a training period and test period.
3: Calculate the set of features $\boldsymbol{f}_n \in F$ over the training period.
4: Fit each forecasting method $m \in M$ over the training period and generate forecasts over the test period.
5: Calculate forecast losses L_{nm} over the test period.
6: **end for**

Train the meta-learner, w

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

$$\underset{w}{\operatorname{argmin}} \sum_{n=1}^{N} \sum_{m=1}^{M} w(\boldsymbol{f}_n)_m L_{nm}.$$

ONLINE PHASE: FORECAST A NEW TIME SERIES

Input:

FFORMA meta-learner from offline phase.

Output:

Forecast the new time series x_{new}.

8: **for** each x_{new}: **do**
9: Calculate features $\boldsymbol{f}_{new}$ by applying F.
10: Use the meta-learner to produce $\boldsymbol{w}(\boldsymbol{f}_{new})$ an M-vector of weights.
11: Compute the individual forecasts of the M forecasting methods in the pool.
12: Combine individual forecasts using $\boldsymbol{w}$ to generate final forecasts.
13: **end for**

3 Implementation and application to the M4 competition

3.1 Reference set

The M4 dataset includes 100,000 time series of yearly, quarterly, monthly, weekly, daily and hourly data. All 100,000 series form the reference set. Each series is split into a training period and a test period. The length of the test period for each time series was set to be equal to the forecast horizon set by the competition. Series with training periods comprising fewer than two observations, or series that were constant over the training period, were eliminated from the reference set.

Table 1: *Features used in FFORMA framework.*

Feature		Description	Non-seasonal	Seasonal
1	T	length of time series	✓	✓
2	trend	strength of trend	✓	✓
3	seasonality	strength of seasonality	-	✓
4	linearity	linearity	✓	✓
5	curvature	curvature	✓	✓
6	spikiness	spikiness	✓	✓
7	e_acf1	first ACF value of remainder series	✓	✓
8	e_acf10	sum of squares of first 10 ACF values of remainder series	✓	✓
9	stability	stability	✓	✓
10	lumpiness	lumpiness	✓	✓
11	entropy	spectral entropy	✓	✓
12	hurst	Hurst exponent	✓	✓
13	nonlinearity	nonlinearity	✓	✓
13	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓
14	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓
15	hwalpha	ETS(A,A,A) $\hat{\alpha}$	-	✓
16	hwbeta	ETS(A,A,A) $\hat{\beta}$	-	✓
17	hwgamma	ETS(A,A,A) $\hat{\gamma}$	-	✓
18	ur_pp	test statistic based on Phillips-Perron test	✓	✓
19	ur_kpss	test statistic based on KPSS test	✓	✓
20	y_acf1	first ACF value of the original series	✓	✓
21	diff1y_acf1	first ACF value of the differenced series	✓	✓
22	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓
23	y_acf10	sum of squares of first 10 ACF values of original series	✓	✓
24	diff1y_acf10	sum of squares of first 10 ACF values of differenced series	✓	✓
25	diff2y_acf10	sum of squares of first 10 ACF values of twice-differenced series	✓	✓
26	seas_acf1	autocorrelation coefficient at first seasonal lag	-	✓
27	sediff_acf1	first ACF value of seasonally differenced series	-	✓
28	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓
29	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓
30	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓
31	seas_pacf	partial autocorrelation coefficient at first seasonal lag	✓	✓
32	crossing_point	number of times the time series crosses the median	✓	✓
33	flat_spots	number of flat spots, calculated by discretizing the series into 10 equal sized intervals and counting the maximum run length within any single interval	✓	✓
34	nperiods	number of seasonal periods in the series	-	✓
35	seasonal_period	length of seasonal period	-	✓
36	peak	strength of peak	✓	✓
37	trough	strength of trough	✓	✓
38	ARCH.LM	ARCH LM statistic	✓	✓
39	arch_acf	sum of squares of the first 12 autocorrelations of z^2	✓	✓
40	garch_acf	sum of squares of the first 12 autocorrelations of r^2	✓	✓
41	arch_r2	R^2 value of an AR model applied to z^2	✓	✓
42	garch_r2	R^2 value of an AR model applied to r^2	✓	✓

3.2 Time series features

Table 1 provides a brief description of the features used in this experiment, F in Algorithm 1. The functions to calculate these are implemented in the `tsfeatures` R package by Hyndman et al. (2018b). Most of the features (or variations of these) have been previously used in a forecasting context by Hyndman, Wang & Laptev (2015) and Talagala, Hyndman & Athanasopoulos (2018), and are described in more detail there. The ARCH.LM statistic was calculated based on the Lagrange Multiplier test of Engle (1982) for autoregressive conditional heteroscedasticity (ARCH). The heterogeneity features 39–42 are based on two computed time series: the original time series is pre-whitened using an AR model resulting in z; a GARCH(1,1) model is then fitted to z to obtain the residual series, r.

Features corresponding only to seasonal time series are set to zero for non-seasonal time series. For the sake of generality, we have not used any of the domain-specific features such as macro, micro, finance, etc., even though this information was available in the M4 data set.

3.3 Pool of forecasting methods

We considered eight methods implemented in the `forecast` package in R (Hyndman et al. 2018a) for the pool of methods, P in Algorithm 1:

1. naïve (`naive`);
2. random walk with drift (`rwf` with drift=TRUE);
3. seasonal naïve (`snaive`).
4. theta method (`thetaf`);
5. automated ARIMA algorithm (`auto.arima`);
6. automated exponential smoothing algorithm (`ets`);
7. TBATS model (`tbats`);
8. STLM-AR Seasonal and Trend decomposition using Loess with AR modeling of the seasonally adjusted series (`stlm` with model function `ar`).

The R functions are given in parentheses. In all cases, the default settings are used. If any function returned an error when fitting the series (e.g. a series is constant), the `snaive` forecast method was used instead.

3.4 Forecast loss measure

The forecasting loss, L in Algorithm 1, was adapted from the Overall Weighted Average (OWA) error described in the M4 competitor's guide *M4 Competitor's Guide* (2018), which adds together the Mean Absolute Scaled Error and the symmetric Mean Absolute Percentage Error. For each series and method, the Mean Absolute Scaled Error and the symmetric Mean Absolute Percentage Error were divided by the respective error of the Naive 2 method *over all series in the dataset* (i.e., MASE by the average MASE of Naive 2), and then added.

3.5 Meta-learning model implementation

We used the gradient tree boosting model of `xgboost` as the underlying implementation of the learning model (Chen & Guestrin 2016). This is a state-of-the-art model that is computationally efficient and has shown good performance in structure based problems. The great advantage

of its application here is that we are able to customise the model with our specific objective function.

The basic `xgboost` algorithm produces numeric values from the features, one for each forecasting method in our pool. We applied the softmax transform to these values prior to computing the objective function. This was implemented as a *custom objective function* in the `xgboost` framework.

`xgboost` requires a gradient and hessian of the objective function to fit the model. The *correct* hessian is prone to numerical problems that need to be addressed for the boosting to converge. This is a relatively common problem and one simple fix is to use an upper bound of the hessian by clamping its small values to a larger one. We computed a different upper bound of the hessian by removing some terms from the correct hessian. Although both alternatives converged, the latter worked faster, requiring less boosting steps to converge. This not only increased the computational efficiency, it also generalized better due to a less complex set of trees produced in the final solution.

The general parameters of the meta-learning in Algorithm 1 were set as follows.

- $p(\boldsymbol{f}_n)_m$ is the output of the `xgboost` algorithm corresponding to forecasting method m, based on the features extracted from series x_n.
- $w(\boldsymbol{f}_n)_m = \frac{\exp(p(\boldsymbol{f}_n)_m)}{\sum_m \exp(p(\boldsymbol{f}_n)_m)}$ is the transformation to weights of the `xgboost` output by applying the softmax transform.
- L_{nm} is the contribution to the OWA error measure of method m for the series n.
- $\bar{L}_n = \sum_{m=1}^{M} w(\boldsymbol{f}_n)_m L_{nm}$ is the weighted average loss function.
- $G_{nm} = \frac{\partial \bar{L}_n}{\partial p(\boldsymbol{f}_n)_m} = w_{nm}(L_{nm} - \bar{L}_n)$ is the gradient of the loss function.
- The hessian H_{nm} was approximated by our upper bound $\hat{H}_{nm}$:

$$H_{nm} = \frac{\partial G_n}{\partial p(\boldsymbol{f}_n)_m} \approx \hat{H}_n = w_n(L_n(1 - w_n) - G_n)$$

The functions G and $\hat{H}$ were passed to `xgboost` to minimize the objective function $\bar{L}$.

The results of `xgboost` are particularly dependent on its hyper-parameters such as learning rate, number of boosting steps, maximum complexity allowed for the trees or subsampling sizes. We limited the hyper-parameter search space based on some initial results and rules-of-thumb and explored it using Bayesian optimization (implemented in the R package `rBayesianOptimization`, Yan 2016) measuring performance on a 10% holdout version of the reference set. We picked the simplest hyper-parameter set from the top solutions of the exploration.

3.6 Prediction intervals

For each series x_{new}, we used as the centre of the interval the point forecast produced by our meta-learner. Then the 95% bounds of the interval were generated by a linear combination of the bounds of three forecasting methods: naïve, theta and seasonal naïve. The coefficients for the linear combination were calculated in a data-driven way over the M4 database. The complete procedure was as follows:

1. We divided the M4 dataset into two parts: A and B. We trained the FFORMA learner using the training periods of the series in part A and produced point forecasts over the test periods of the series of part B, and vice versa.
2. We computed the 95% *prediction radius* for the naïve, theta, and seasonal naïve methods. This is the difference between the 95% upper bound and the point forecast for each forecast horizon.
3. For each forecast horizon we found the coefficients that minimized the MSIS of the interval, as defined in the M4 Competitor's guide (*M4 Competitor's Guide* 2018), with the FFORMA point forecast as the centre and a linear combination of the radii of naïve, theta, seasonal naïve forecasts as the interval. The minimization was done by gradient descent over the test period of the series.

This method produced a set of three coefficients for each prediction horizon in the M4 dataset and these coefficients were the same independently of the series we want to forecast. Unlike the point forecasts, these coefficients were not restricted to be probabilities.

4 Discussion and conclusions

We have presented an algorithm for forecasting using weighted averaging of a set of models. The objective function of the learning model assigns weights to forecasting methods in order to minimize the forecasting error that would be produced if we picked the methods at random using these weights as probabilities. This contrasts with how the final forecasts are produced, which is a weighted average, not a selection.

These weights can however be used as part of a model selection algorithm, if one picks the method receiving the largest weight. This can be useful for interpretability or computational reasons, at the cost of forecasting performance.

In order to evaluate the impact of our contribution, we compared the average forecast error produced by FFORMA with a model selection approach. All implementation details were kept the same as in FFORMA; specifically we used the same set of features, the same pool of forecasting methods, the same underlying implementation (`xgboost`) but with a standard cross-entropy loss, and the same hyper-parameter search. This enabled us to measure the impact of the FFORMA loss function against a model selection approach, all other things being equal. We applied both FFORMA and the model selection approach to the M4 dataset and compared their overall point forecast errors. The average OWA error of the FFORMA approach was 10% smaller than that for model selection. This improvement is entirely due to the proposed model averaging rather than model selection.

One advantage of our approach is that its form is independent of the forecasting loss measure. Forecast errors enter the model as additional pre-calculated values. This allows FFORMA to adapt to arbitrary loss functions when models that directly minimize them would be restricted. For example, our approach can be applied to non-differentiable errors.

The source code for FFORMA is available at github.com/robjhyndman/M4metalearning.

References

Bates, JM & CWJ Granger (1969). The Combination of Forecasts. *Journal of the Operational Research Society* **20**(4), 451–468.

Chen, T & C Guestrin (2016). Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, pp.785–794.

Clemen, R (1989). Combining forecasts: a review and annotated bibliography with discussion. *International Journal of Forecasting* **5**, 559–608.

Engle, RF (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 987–1007.

Hyndman, RJ, G Athanasopoulos, C Bergmeir, G Caceres, L Chhay, M O'Hara-Wild, F Petropoulos, S Razbash, E Wang & F Yasmeen (2018a). *forecast: Forecasting functions for time series and linear models*. R package version 8.3. http://pkg.robjhyndman.com/forecast.

Hyndman, RJ, E Wang, Y Kang & T Talagala (2018b). *tsfeatures: Time Series Feature Extraction*. R package version 0.1. https://github.com/robjhyndman/tsfeatures.

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.

Kang, Y, RJ Hyndman & K Smith-Miles (2017). Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting* **33**(2), 345–358.

Kück, M, SF Crone & M Freitag (2016). Meta-Learning with Neural Networks and Landmarking for Forecasting Model Selection - An Empirical Evaluation of Different Feature Sets Applied to Industry Data Meta-Learning with Neural Networks and Landmarking for Forecasting Model Selection. *International Joint Conference on Neural Networks*, 1499–1506.

Lemke, C & B Gabrys (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing* **73**(10). Subspace Learning / Selected papers from the European Symposium on Time Series Prediction, 2006–2016.

M4 Competitor's Guide (2018). https://www.m4.unic.ac.cy/wp-content/uploads/2018/03/M4-Competitors-Guide.pdf. Accessed: 2018-09-26.

Prudêncio, R & T Ludermir (2004). Using machine learning techniques to combine forecasting methods. In: *Australasian Joint Conference on Artificial Intelligence*. Springer, pp.1122–1127.

Smith, J & KF Wallis (2009). A simple explanation of the forecast combination puzzle. *Oxford Bulletin of Economics and Statistics* **71**(3), 331–355.

Talagala, TS, RJ Hyndman & G Athanasopoulos (2018). *Meta-learning how to forecast time series*. Working Paper 6/18. Department of Econometrics & Business Statistics, Monash University.

Timmermann, A (2006). "Forecast Combinations". In: *Handbook of economic forecasting*. Ed. by Graham Elliot and Clive W. J. Granger and Allan Timmermann. Amsterdam, North-Holland. Chap. 4, pp.135–196.

Yan, Y (2016). *rBayesianOptimization: Bayesian Optimization of Hyperparameters*. R package version 1.1.0. https://cran.r-project.org/web/packages/rBayesianOptimization/index.html.