Recurrent neural network implementations on the M4 dataset

Thomas J. Delaney

Contents

1 Introduction

The purpose of this document is to describe the recurrent neural networks (RNNs) applied to the M4 dataset as an attempt to forecast the financial series in the M4 dataset.

2 Objectives

Our main objective in this experiment was to create a forecasting method similar to that of Smyl et al (2018), which won the M4 forecasting competition [3]. More specifically, we wanted to create forecasting method that blended statistical forecasting with machine learning methods using recurrent neural networks. So, our aims were:

- To train non-seasonal and seasonal forecasting methods using statistical techniques and RNNs together.
- To use these methods to forecast the financial series in the M4 dataset.
- To compare the performance of these methods to simpler statistical methods.
- To compare the performance of these methods to the M4 winning statistical-RNN hybrid model.

Our hope (rather than expectation) was that a model simpler than that in [3] could be used to forecast the financial time series.

3 Background

For background on the M4 competition, see [1] or [2].

The winner of the M4 competition was a hybrid forecasting model that combined simple exponential smoothing methods with recurrent neural networks. It was developed by Slawek Smyl while working at Uber. In this model, simple exponential smoothing is used to estimate a multiplicative level and seasonality of each time series. These level and seasonality components are used to scale the time series before the time series is passed to the RNN. An RNN with a different architecture is used for each time series frequency. Each RNN uses LSTM cells and dilation, and the attention mechanism is used for some frequencies.

The learning mechanism and model is hierarchical in nature. The exponential smoothing parameters are learned for each individual time series, whereas the weights in the RNN are learned across all the scaled time series. For more information on this model see [3] or [4]. For more information on recurrent neural networks, see [4].

The statistical modelling in this experiment was performed in `R` using the `forecasting` package. We specifically used an *ets model* for modelling both seasonal and non-seasonal data. For more information about ets models, other statistical models, and the `forecasting` package see [5].

The RNNs in this experiment we created and trained in `R` using *RStudio* and *Keras*. Keras is a high-level neural networks API enabling quick and easy building of and experimentation with neural networks. We used *tensorflow* as the back-end for Keras. For for information on Keras, see [6]. For more information on Tensorflow, see[7].

4 Methods

4.1 Statistical Modelling

In this section, we describe the statistical models applied to the time series. Bare in mind that these models were not intended to accurately model or forecast the time series. The intention was to use the level and seasonal components extracted to scale the time series before passing the scaled series into the RNN.

4.1.1 Non-seasonal

To statistically model the yearly time series, we used a non-seasonal ets model with a multiplicative error and no trend component. This models the time series as a level component with multiplicative white noise error. The model is defined by

$$y_t = \ell_{t-1}(1 + \epsilon_t) \tag{1}$$
$$\ell_t = \ell_{t-1}(1 + \alpha\epsilon_t) \tag{2}$$

where y_t is the time series, ℓ_t is the level component, and α is a smoothing parameter.

4.1.2 Seasonal

To statistically model quarterly time series, we used an ets model with a multiplicative seasonality, multiplicative error, and no trend component. This models the time series as a level component with a multiplicative seasonal component, and a multiplicative white noise error. The model is defined by

$$y_t = \ell_{t-1}s_{t-m}(1 + \epsilon_t) \tag{3}$$
$$\ell_t = \ell_{t-1}(1 + \alpha\epsilon_t) \tag{4}$$
$$s_t = s_{t-m}(1 + \gamma\epsilon_t) \tag{5}$$

where y_t is the time series, ℓ_t is the level component, α is a level smoothing parameter, s_t is the seasonal component, γ is the seasonal smoothing parameter, and m is the number of time points (or seasons) in a full seasonal cycle (e.g. 4 for quarterly data).

4.2 Recurrent Neural Network

The RNN that we used in this experiment is fairly simple compared to those used in [3]. Time contraints were the main reason for this. The network consisted of an input layer, two hidden layers, and an output layer.

The first hidden layer consisted of 32 gated recurrent units (GRUs), with $\tanh$ output activation and *hard sigmoid* recurrent activation (see 4.2.1). A standard dropout rate of 10% and recurrent dropout rate of 50% were implemented.

The second hidden layer consisted of 64 GRUs. This time a recitfied linear unit output activation was used.The same dropout rates were used.

The output layer consisted of a number of densley connected units equal to the number of points to be forecast (6 for yearly, 8 for quarterly).

4.2.1 Hard sigmoid activation function

The hard sigmoid function is a piece-wise linear approximation of the sigmoid function. When plotted in two dimensions, the function consists of three linear segments, one along the x-axis, one with a positive slope, and one parallel to the x-axis with $f(x) = 1$. The fuction can be defined by

$$f(x) = \begin{cases} 0, & \text{if } x < -\frac{1}{2m} \\ mx + \frac{1}{2}, & \text{if } \frac{1}{2m} \leq x \leq \frac{1}{2m} \\ 1, & \text{if } x > \frac{1}{2m} \end{cases} \tag{6}$$

where $m > 0$ is the slope of the middle section of the function.

This function is commonly used in place of the sigmoid function because it's much quicker to calculate the hard sigmoid. Instead of performing the calculations necessary for $\frac{1}{1+e^{-x}}$, all that is required is to calibrate m.

4.3 Combining the statistical model and RNN

4.3.1 Non-seasonal

In order to combine the statistical model with the recurrent neural network, and attempt to use the power of both we did the following:

1. We split the dataset of non-seasonal M4 series into training, validation, and test sets.
2. We fit the non-seasonal ets model described in section 4.1.1 to each of the training series. This returned a level series, and a residual series for each time step in each series.
3. We trained the RNN to forecast the residual series, using the mean absolute error as the loss function. If we examine equation 1, we can see that forecasting the residuals is equivalent to forecasting a scaled and translated version of the time series, namely $\frac{y_t}{\ell_{t-1}} - 1$.
4. In order to obtain a descaled and detranslated forecast, we used the values for ϵ_t given to us by the RNN output along with equations 1 and 2 to calculate y_t and ℓ_t for each timestep in the forecast horizon.

We used the validation set during the training of the RNN, and to quantify the performance of our hybrid method compared to the baseline method.

4.3.2 Seasonal

The process for the seasonal time series was similar to that for the non-seasonal, except that the fitted ets model (as described in section 4.1.2) returned a seasonal series as well as a level series and a residual series. The residual series was equivalent to $\frac{y_t}{\ell_{t-1}s_{t-1}} - 1$ In order to obtain the final forecast, we had to use the level and seasonal series with equations 3, 4, and 5 to descale and deseasonalise the output of the RNN.

4.4 Baseline Methods

In order to assess the performance of our hybrid model, we compared its forecasting performance to that of a simpler statistical model. We chose different simple models for non-seasonal and seasonal data.

4.4.1 Non-seasonal

For non-seasonal forecasting we used a random walk model with a drift component as the baseline model.

4.4.2 Seasonal

For seasonal forecasting, we used a seasonal naïve method as the baseline model.

5 Results

6 Conclusions

References

[1] Spyros Makridakis, *M4 Competition*. https://www.m4.unic.ac.cy/about/, University of Nicosia, (2018)

[2] Thomas Delaney, *Meta-learning for financial forecasting*. https://github.com/thomasjdelaney/Financial-Forecasting/blob/master/latex/meta-learning method review/meta-learning_method_review.pdf, (2018)

[3] Slawek Smyl, Jai Ranganathan, Andrea Pasqua, *M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model*. https://eng.uber.com/m4-forecasting-competition/, (2018)

[4] Thomas Delaney, *Forecasting Methods: An Overview*. https://github.com/thomasjdelaney/Financial-Forecasting/blob/master/latex/forecasting lit review/forecasting_lit_review.pdf, (2018)

[5] Rob J Hyndman, George Athanasopoulos, *Forecasting: Principles and Practice*. Monash University, Australia, (2018)

[6] Keras, *R interface to Keras*. https://keras.rstudio.com/, (2018)

[7] Tensorflow, *TensorFlow*. https://www.tensorflow.org/, (2018)

[8] Evangelos Spiliotis, Spyros Makridakis, Vassilios Assimakopoulos, *The M4 Competition in Progress*. https://www.m4.unic.ac.cy/wp-content/uploads/2018/06/Evangelos_Spiliotis_ISF2018.pdf, (2018)