MineThatData Forecasting Challenge: proposed solution with Bayesian Structural Time Series models

0. Index

1. Introduction & purpose

The main purpose of this document is to provide a solution for the MineThatData Forecasting Challenge. Equally important, it provides an argument for the modelling choices of the authors; the aim is to give an explanation which is readable for somebody who is not so familiarized with predictive analytics.

Thus, the objective is not to give a full introduction either into time series analysis, or structural models, or Bayesian statistics. This document is intentionally brief, but there is plenty of literature around for the interested reader (some pointers in References section). In the Methodology section, the description of the modelling techniques try to be simple and practical; whenever it is possible, translations into a business context are suggested.

Additionally, there is a very complete open source package in R, called “bsts”, developed by Steve Scott (see Acknowledgements). The solution proposed relies on this packet. All the code used for fitting the model is available in a HTML file annex to this document, and it is also posted on the web as a blog post. In consequence, along the document, no complex equations or mathematical derivations are used either.

The MineThatData Forecasting Challenge provides a target series of total sales data for 80 months, and requests, as the main objective, a forecast of the series for the 3 following months. There is no other data available (thus no explanatory variables to the prediction).

Final note: in the paragraphs that follow, the explanations refer to generic time series and prediction problems; however, whenever it makes the explanation more clear, the features of the target series of the Forecasting Challenge are used: the fact that it is a sales series, the data is provided monthly, etc.

2. Summary of results

Find below the proposed forecast of the Total Annual Sales series for the 3 following months.

month	80	81	82
sales	81,532,403	81,236,545	81,002,410

Note that, due to the lack of explanatory variables, this prediction should be considered by the business as a baseline, since it takes the strong assumption that the past behaviour of sales is going to determine their future values.

The prediction uses a trend component only. In the exploratory analysis of the series, a weak seasonal effect can be found, with period equal to 1 year; however, when trying to fit an additional component to capture that periodicity, the improvement of the results is negligible and does not justify the additional complexity.

3. Methodology: Bayesian Structural Time Series models (BSTS)

3.1. What is a BSTS model

BSTS models are reasonably well explained from its initials:

- *Time Series*: i.e. they only have sense in the context of time series. They are mostly used for forecasting, although the *state-space* models, for which BSTS is just an extension (see Nomenclature), have traditionally been used in engineering active filters, and are currently used in a variety of applications
- *Structural*: meaning that BSTS provide a structural approach to the modelling, in which a kit of components are available for capturing different aspects of the series; the architecture of the model can include or exclude any of those components, as a kind of building game
- *Bayesian*: meaning that the implementation used for this proposal has a Bayesian approach. This does not mean that one has to "turn" into Bayesian Statistics. In a pragmatic way, it has 2 main consequences, [1] all the outputs of the model will come in a distribution with a certainty interval (and actually all parameters inside the model will have a distribution), and [2] it is possible to express prior knowledge about the target series through *Bayesian priors* (which can be regarded as hyperparameters of the model)

In its different forms (state-space models, Kalman filters, see again Nomenclature), the BSTS have been used "traditionally" (i.e. since the 60s), are still in use, since they are a very good fit for some scenarios. They are not so commonly covered in online courses and resources, tutorials, etc, and are a bit more difficult to use compared to other machine learning models; however there is a very high quality open source library available in R, reasonably straight-forward in its use, and well documented.

3.2. Comparison with other approaches

The target series to predict, there are two main alternatives that would come to mind:

- ARIMA-like models: yet another "traditional" approach for forecasting of time series
- Neural networks/deep learning: maybe the most trendy topic nowadays, could be certainly used to provide a prediction

Other approaches, like regression and ensemble models (random forest, gradient boosting, etc), cannot be used since there is just the time series to predict, with no explanatory variables.

ARIMA/Box-Jenkins models could certainly be a robust approach for the problem in hand. One of their main limitations is that they do not allow for explanatory variables, however in this case it is not a problem. The following are other disadvantages of this approach:

- They are not very intuitive and require quite some knowledge on Statistics, mostly since the series to predict must be converted into stationary as a preparation step (this transformation is not at all easy, requires knowledge and experience, and anyhow the criteria to remove trend and seasonality of a series is not easy to establish)
- In contrast, BSTS allows non-stationary series and supports a sensible and more natural treatment of the trend and seasonality of the series
- Besides, most ARIMA models can be expressed as a *state-space* equation, which in essence means they can be modelled as BSTS as well

See Harvey (1989) chapter 2 for a full discussion on the comparison between Structural models and ARIMA; also Brockwell (2016) for a light introduction to ARIMA/Box-Jenkins; and for a brief summary through presentation slides see Scott (2015).

On the other hand, the problem could be certainly tackled by a recurrent neural network. Even if the length of the series is pretty short, a LSTM layer with a few cells may be able to provide a good prediction. It is not easy to discard upfront the feasibility of this kind of approach. Among the disadvantages:

- Needs more more technical knowledge in order to set-up the LSTM layer; there is no off-the-shelf library based in RNN for prediction of time series. On the other hand, using BSTS also requires some technical knowledge, even if the R bsts library if fairly easy to use
- The problem will tend to be over-specified very quickly; neural networks have many parameters to train. The architecture should be kept small for an example like this one
- The results of the neural network will be hard to interpret from a business point of view. Pretty much the only thing you will be able to say is whether it predicts well or not

3.3. Structural components

BSTS provides a kit of components which can be used for modelling the series to predict. These components will capture different aspects of the series, and can be added or removed according to needs, like in a building game. In the *classical decomposition*, they are classified roughly as:

- Trend components, which capture the slowly changing aspects of the series
- Seasonal components, which capture the periodic aspects of the series
- Regression components, which capture the influence of the explanatory variables (i.e. variables external to the series to predict that provide some information to the prediction)

For the proposed solution, only the trend components are really relevant:

- The target series has only a mild seasonal aspect (with period equal to 1 year). If a seasonal component is added, it captures a small amount of the signal, but the effect is so small that it does not justify the additional complexity
- There are no explanatory variables available for the target series. There is only the values of the sales, and there is not even a reference of which month is which (if this was available, some synthetic variables could be added, e.g. marking holiday periods, hoping that they would have some influence on the sales)

The full kit of components is described in the vignette of the BSTS library. For the scope of this document, let's just describe the simplest and the most complex of the trend components:

- The simplest trend component is called *local level*, and, as the name suggest, it basically captures the *level* of the series. Which begs the question: how much of the information of the series should be considered as *level*? The component can be parameterized to indicate how much energy of the series actually belongs to the trend. See section on Bayesian approach for details
- The most complex component is called *semi-local linear trend*, and it describes the trend in a much more complex way, including the *level* (as in the component above), but also a *slope* (meaning the how fast or slow is the level changing), which is modelled as a autoregressive AR(1) process. Regardless of the details, this component will capture as trend some more intricated aspects of the series, and will usually perform better in a long-term forecast

3.4. What are BSTS for and how they work internally (roughly)

BSTS models two main work around two main operations: *filtering* and *smoothing*.

- *Filtering* provides a one step prediction of the series, given all the data available up to the moment; i.e. in a monthly series like the target, it provides the prediction for next month, given all the data available up to the current month
- *Smoothing* corrects the status of the model when a new observation of the series is available; i.e. if the model has provided a prediction for this month, the month comes to an end and the observation of the real sales (i.e. of the month just finished) is available, the model compares the prediction with the observation, and uses the error to correct its own status

When training the model, the library internally goes through all the sales data available, month by month, running successive filtering and smoothing operations, up to the last month of sales data available.

After that, the model may be used for forecasting future values of the series by applying consecutive filtering steps, as many of the number of months to predict (usually called the *horizon* of the prediction).

It is important to note that these consecutive filtering operations become less and less accurate for an increasing *horizon* of the prediction; the predictions will not be very useful for the last months of the prediction horizon, if it is very wide. It is difficult to define an optimal width of the horizon, depending on the characteristics of the series and the complexity of the model. However, it is definitively true that for the requirements of the challenge in hand, which is 3 months, BSTS should be fine.

3.5. Bayesian approach: setting up priors

In each the components of a BSTS model, it is possible to setup *Bayesian priors* that capture prior information about the target series to predict. First glance, it does not seem easy for the analyst to translate the business knowledge about the series and the domain into a *sigma prior,* for example. However, it should be possible for a business analyst (who maybe knows what is *variance* but anyhow has no idea about what is a *sigma prior*), to figure out the necessary information.

Let's stick for this explanation to the simplest trend component available, the *local level* component, which is a trend component. Looking at the documentation in the R package, it has 2 priors:

- *initial.state.prior*: "describes the prior distribution of the the initial state vector (at time 1)"
- *sigma.prior*: "describes the prior distribution for the standard deviation of the random walk increments"

The *initial.state.prior* may be skipped, in particular if the analyst is unsure on what is the *state vector*. In most practical situations, the model figures out correctly the initial value of the state vector with no need of adding prior info.

What about the *sigma.prior*? The *local level* component describes the trend in a fairly simple way; it can be seen as an *exponential smoothing* of the previous values of the series (see a short paper in which this is explained quite intuitively -you may skip the mathematical derivations). Thus, the model predicts the value of the next step (i.e. for the target series, the sales value for next month) as a *smoothing* of the past values. The *prior* can be then seen as a parameter for this *smoothing*: it determines how tight is the trend component going to follow the latest values of the series. A high value of *sigma* indicates that the component will follow the series tightly, and a low value indicates the opposite.

Maybe more clearly, it will behave as follows:

- If the difference between the last available real value of the series and the second last is big, and the *sigma* is big, the prediction for the next month will have again a big difference with the last value (it will follow the trend tightly).
- In the same situation, if the *sigma* is low, the prediction for the next month will not have such a big difference (it will not follow the trend so tightly).

The question is of course how to decide what value is best for *sigma*; this is the point in which a business translation is handy, so that the consequences of the possible values of the *prior* are understood. Better done with two examples of the two extremes:

- In a business context in which there is a high seasonality, i.e. all the sales are concentrated in one period.
  If there is an increase of the sales outside of the high season period, probably it should not be understood as a hint of even increased sales in the high season period. In this case, the trend component should follow the series loosely (low *sigma*) and leave the seasonal component to capture the behaviour in the high season period

- In a business context in which there is an adoption rate, e.g. there's contagion effects in the sales (mouth to ear, a critical mass is needed, etc).
  In this case, probably there's no seasonality at all and the trend component should follow the changes as tight as possible (high *sigma*). If there is a high increase in sales in the last month, it will probably mean an even higher increase next month

3.6. What to do when you do not have clear priors

Last section, attempting to provide an intuitive idea on how to setup *Bayesian priors*, may not be convincing enough. It could be that certain business contexts are not so closed to one of the extremes; maybe in some time periods or situations it would be better that the trend follows the series tightly, but a few months later it is the opposite.

It could happen as well, as in the example for this Forecasting Challenge, that there is no info available for the business context.

In these situations, the values of the priors can be set using *brute force*. Disadvantages:

- It might take a longer time
- It might provide a solution that it a *local minimum*; i.e. is better than other solutions, but not the best of all
- Might make you feel *brute*

The advantage is that you get a solution, probably even a good one. The procedure is just equivalent to treating the *Bayesian priors* as hyperparameters for the machine learning problem. Which basically means:

- Setting a grid of possible values of the hyperparameters that you want to test (i.e. the *sigma*'s of the priors).
- Testing the precision of the model for each combination of values of the hyperparameters (for how to check the precision, see next section on cross-validation).
- Selecting the hyperparameters that give the best precision overall.

3.7. How to validate the BSTS model

For validating the model, one of the possible approaches is called *cross-validation*; amongst its advantages, it is quite intuitive.

In a time series, *cross-validation* divides the training data into different *folds*, and in each iteration a part of the data is reserved for validating the results. Concretely, for the target series (having 80 months of training data), the schema can be as follows for 10 folds of horizon 3:

- For *fold* 1, 77 months of data are used for training the model, while the 3 last months are reserved for checking the results
- For *fold* 2, 74 months are used for training the model, while the last 3 are for validation
- … successively until…
- For *fold* 10, 50 months are used for training the model, while the last 3 are for validation

This cross-validation scheme is useful for:

- Training the model while at the same time avoiding *overfitting*, i.e. the precision obtained by the model will likely generalize for unseen data, so that the 3 predictions proposed for the challenge will be good
- Running the cross-validation loop using each of the possible trend component will allow choosing the best one: the one giving best predictions for all folds
- An analogous procedure will be used for setting the hyperparameters: running the cross-validation loop for each of its possible values, and selecting the best one

Note that, in the literature, this flavour of cross-validation for time series is usually called *forward chaining*.

4. Results of BSTS applied to the Forecasting Challenge

The full implementation in R is in the HTML annex to this document, and also as a blog post.

As a summary of the steps followed for providing a forecast:

- Loading and cleaning the data is pretty straight-forward
- Exploratory analysis of the data gives only a week yearly periodicity
- The next logical step is to fit some initial models and explore the residuals
- Eventually, brute force is applied, with two objectives
    - Deciding among the 3 trend components available in the library, in increasing order of complexity: *local level*, *linear local*, and *semi-local linear*. More complex

components should perform better in a prediction with long horizon, but the cross-validation setup finds the most suitable
    - For each component, the possible values of its parameters are set in a grid, and the cross-validation setup selects the best of the values
- Finally, the chosen model is fit, and the prediction for the Forecasting Challenge is produced

5. Annexes

5.1. Some messy nomenclature

For anybody interested on reading more about BSTS, everything can become a bit confusing at the beginning, due to the long history of this kind of models, and the messy nomenclature. A couple of basic concepts below.

State-space models: modelling methodology in which the system is described as composed of a state vector and an observation vector, both time series. The relation between state and observation is described by the state-space model; the objective is to infer the properties of the state, which is hidden, from the observations available in the past. Forecasts are then produced from the estimated future states.

Kalman filter: designed in the 60's, and famously used in the Apollo 11 guidance system, is actually the first state-space model ever. That's why some concepts are still called like Kalman did coin them: Kalman filter, Kalman gain.

5.2. References and acknowledgements

Acknowledgements and praise go to Steve Scott, who is the creator and maintainer of the bsts library. Full credits in the vignette. The library is straight-forward in its use, and well documented. Version 0.7.1 has been used.

The following references have been used along the document:

- More materials from Steve Scott

Scott S.L. (2015) Bayesian Structural Time Series Models. Google. Presentation slides: https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxzdGV2ZXRoZWJheWVzaWFufG%20d4OjI2ZGEwMTk4M2VmOWRkOTE

- Classical books on BSTS (with the all the messy nomenclature in them)

Durbin J., Koopman S.J. (2012) Time Series Analysis by State Space Methods (Second Edition). Oxford University Press.

Harvey A.C. (1989) Forecasting, structural time series models and the Kalman filter. Cambridge University Press.

- A short and interesting paper on the Kalman filter

Meinhold R.J., Singpurwalla N.D. (1983) Understanding the Kalman filter. The American Statistician, vol. 37.

- A classic reference for Time Series

Brockwell P.J., Davis R.A. (2016) Introduction to Time Series and Forecasting. Springer Texts in Statistics.

5.3. Sisifo's Page team

Karolina Serna: sernamayolo.karolina@gmail.com
Agustín López: agustinldg@gmail.com
Lorenzo Rubio: lrnzcig@gmail.com

http://sisifospage.tech/