MXN442- Assessment 2: Final Project

By James Donovan (n9164090)

1.0 Abstract

The following document details an attempt to replicate the results of Florian Ziel's 2022 publication: "*Smoothed Bernstein Online Aggregation for Short-Term Load Forecasting in IEEE DataPort Competition on Day-Ahead Electricity Demand Forecasting: Post-COVID Paradigm*". Data discrepancies between the host website and what was reported in the paper meant exact replication was not possible but the goal to implement a working sBOA (smoothed Bernstein Online Aggregation) remained. The project redefined the training and testing dataset by subsetting the available training data which provided a method for comparing improvements to the Ziel's methodology in a like-for-like environment. Iterative testing of the smoothing parameter revealed predictive performance could be improved but its positive impact had questionable practical value. The project introduced RNNs (Recurrent Neural Networks) in place of STLs (Seasonal Trend Decomposition models) to Ziel's original ensemble but this failed to improve sBOA MAE (mean absolute error). Recommendations were made to extend this work by: adding a step-wise procedure for whittling the model ensemble list, convert this code into a functional library.

2.0 Introduction

The following report is a critical review of Florian Ziel's 2022 publication: "*Smoothed Bernstein Online Aggregation for Short-Term Load Forecasting in IEEE DataPort Competition on Day-Ahead Electricity Demand Forecasting: Post-COVID Paradigm*". Detailed in the following sections are: a summary of the paper and its findings, identified methods for extending the research, then the sequential process of implementing the suggested improvements and the results of the exercise.

In line with the MXN442 task sheet, this report evaluates Ziel's 2022 paper by attempting to replicate most of the code the author created to answer their publication's research question. An indicator for success in this facet is producing similar results to what was stated. Additionally, a requirement of the following evaluation was to broaden Ziel's findings by addressing the limitations they identified in the paper's recommendations.

Prior to continuing, it is important to assert the assumed knowledge and intended audience of this document. Readers are expected to be peers, tutors, and lecturers of MXN442. As such, this paper has chosen to focus on the statistical theory that underpins why certain concepts should work versus an introduction on how it works.

3.0 Background information

Whilst this section is titled background information, its purpose is to detail: how the smoothed Bernstein Online Aggregation (sBOA) algorithm works, the key decisions Ziel made in implementing the model, and a brief explanation on why two of the models were included in the ensemble. The rationale for focusing on the algorithm and Ziel's data transformations are to compliment the code (section 11.0) structure.

Ziel's publication details the creation and application of a smoothed Bernstein Online Aggregation (sBOA) algorithm for a short-term energy load forecasting competition. The competition was hosted by IEEE where contestants were given historical electricity demands and weather forecasts (mid 2017- end 2020) and then asked to produce day-ahead predictions for electricity load (kWh). The complexity in this problem was that the training data extended over the COVID-19 pandemic, which transformed the response as a deterministic trend (stationarity) into a stochastic state. sBOAs are considered an appropriate solution to forecasting non-stationary phenomenon due to their iterative weight updates as described below:

$$\hat{y}_{d,h,k} = L_{d,h}^{\text{comb}}$$

Step 1 $$L_{d,h}^{\text{comb}} = \sum_{k} w_{d-1,h,k}\, \hat{L}_{d,h,k}$$

$$\hat{L}_{d,h,k} = f_k(X_{d,h,k})$$

Step 2 $$r_{d,h,k} = AD^{\nabla}(L_{d,h}^{\text{comb}}, y_{d,h,k}) - AD^{\nabla}(\hat{L}_{d,h,k}, y_{d,h,k})$$

$$AD(y, x) = |y - x|$$

Step 3 $$E_{d,h,k} = \max\left(E_{d-1,h,k}, |r_{d,h,k}|\right)$$

Step 4 $$\eta_{d,h,k} = \min\left(\frac{E_{d,h,k}}{2}, \sqrt{\frac{\log_e(K)}{\sum_t (r_{t,k})^2}}\right)$$

Step 5 $$R_{d,h,k} = R_{t-1,k} + \frac{r_{d,h,k}(\eta_{d,h,k} r_{d,h,k} - 1)}{2} + E_{d,h,k}(-2\eta_{d,h,k} r_{d,h,k} > 1)$$

Step 6 $$w_{d,h,k} = \frac{\eta_{d,h,k} e^{(-\eta_{d,h,k} R_{d,h,k})} w_{d-1,h,k}}{\frac{1}{K}\sum_k \eta_{d,h,k} e^{(-\eta_{d,h,k} R_{d,h,k})}}$$

Step 7 $$w_{d,h,k} = B(B^T B + \lambda \Delta^T \Delta)^{-1} B^T w_{d,h,k}$$

The above functions capture the structure of the sBOA and it is important to identify how each step collectively combines to produce highly accurate predictions on stochastic states. Step 1 computes the combined prediction, $L_{d,h}^{\text{comb}}$, as a weighted sum of individual model predictions, $\hat{L}_{d,h,k} = f_k(X_{d,h,k})$, where each weight, $w_{d-1,h,k}$, reflects a model's performance up to the previous time step. This aggregation allows for a dynamic ensemble that accounts for historical accuracies. The instantaneous regret, $r_{d,h,k}$, is seen to measure the difference in absolute deviation between the combined prediction and each individual model's forecast. Step 2 accounts for each model's contribution to the overall error which influences the range ($E_{d,h,k}$). Updating the range is done by selecting the larger of either it's previous value or the instantaneous regret. This ensures that the learning rate, $\eta_{d,h,k}$, prevents overly aggressive weight updates when encountering large differences. The choice to make the learning rate as the minimum of half the range or the inverse proportion of the cumulative regret (squared) balances the need for responsive adjustments with the risk of instability. Step 5 calculates the regret ($R_{d,h,k}$) which leverages a penalty mechanism that increases more rapidly for models with higher recent regret. It can be seen that the condition$(-2\eta_{d,h,k} r_{d,h,k} > 1)$ serves as a safeguard against excessively large weight updates that could see a see-saw change in a model's contribution. Step 6 uses an exponential decay based on cumulative regret to ensure that consistent poorly performing models have their contribution progressively lowered whilst the normalisation in the denominator maintains the weights as a valid probability distribution.

Of greatest importance to this report is step 7, the smoothing of the weight updates which is dependent on: a cubic B-spline matrix, a difference matrix (Δ), and a smoothing hyperparameter (λ). At the time of writing, there is no statistical software which creates sBOAs, demonstrating the novelty of Ziel's algorithm. B contains basis functions defined over a set of knots (the specific points that determine where and how the pieces of the spline join). By representing the weights in terms of these spline functions, the algorithm imposes a smooth structure on the weight vector, preventing abrupt changes that could result from volatile model performances. The incorporation of Δ is to measure the roughness/ variability in the weights through capturing differences between adjacent spline coefficients. As such, their component in step 7 adds a regularization term that penalizes excessive fluctuations in the weight adjustments. λ controls the strength of the smoothing effect imposed by the computation of the difference matrices. Holistically, all steps work in cohesion to produce a robust ensemble of predictions that can handle non-stationary data and mitigates the impact of transient errors or outliers to the aggregated output.

It is pertinent to clarify that Ziel does not leave the sBOA alone to deal with the stochastic state of the data. A log transformation is one of the adjustments made to the original dataset from IEEE. Whilst not explicitly stated in either Ziel's report or the code in section 11.0, applying `tseries::kpss.test()` and `MASS:: boxcox()` to the response variable shows a transformation is necessary. The KPSS (Kwiatkowski-Phillips-Schmidt-Shin) test returns a statistically significant ($p < 0.01$) result, and the Box-Cox returns a lambda of ≈0.06. It is reasonable to assume Ziel was aware of this result and opted for the easier conversion (and explainability) of `$\log x = e^x$` verses the marginal gain in stationarity from selecting $\lambda = 0.06$.

Ziel also introduces daily rolling means of the weather forecasts in a clear attempt to capture the underlying trends whilst simultaneously smoothing out short-term fluctuations in the data. Including such an adjustment allows modelling of the indirect autoregressive relationship of immediate and lagged weather effects on electricity demand.

The last preprocessing action applied to the data was a holiday adjustment to the log load for December 11th and 18th (all years). It is relevant to point out that Ziel identifies these dates "by eye" through comparing daily and hourly dependant trends- a reasonable inference given the origin of the data was hidden(competition rules). Whilst this change is valid, other holiday adjustments were made to dates using this same intuition however Ziel fails to specify which: here Ziel inadvertently removes the ability to fully replicate their methodology. A LASSO regression model is trained on a matrix of weather forecasts (hourly and daily rolling means) and lagged log load to predict and impute the electricity demand if it was not a holiday. Using a LASSO model for this task is well supported in literature as it combines variable selection and regularization through its L1 penalty parameter (λ). This penalty encourages sparsity by shrinking less significant coefficients to zero, making predictions based on only the most impactful variables. It is worth noting, this choice in model contrasts the recommendations of the MXN442 textbook for imputation of outliers/ anomalies. The textbook suggests using ARIMA models which initially seems a suitable decision considering their ability to handle autocorrelations. However, ARIMAs cannot account for exogenous variables (weather) making their comparative performance to LASSOs in holiday load adjustments less effective.

Ziel is seen to include LASSOs for the aforementioned reasons in the final sBOA ensemble used for the IEEE competition. The following two paragraphs will focus on briefly describing two of the models taught in MXN442 which appear in the sBOA of the publication. Below is the formula for LASSO:

$$\min_{\beta}\left(\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|\right)$$

As previously stated, the L1 (λ) regularisation term performs variable selection and coefficient shrinkage. The above paints a clearer picture of the balancing act between the first component of the formula, the sum of squared errors, and the model complexity observed in the number of predictors (β_j). This structure is suitable for energy prediction as its feature selection mitigates overfitting.

Generalized Additive Models (GAMs) are the second model of interest and are characterised by the inclusion of non-linear functions of each predictor variable. GAMs take on a form similar to regression models but differ in the inclusion of the smoothing function (f_j) for $x_{j,i}$ in place of the traditional $\beta_j x_j$. The smoothing function models any non-linear effects caused by temporal changes in the predictor which intuitively captures timestep patterns or fluctuations. Its formula is below:

$$y_i=\beta_0+\sum_{j=1}^{p}\left(f_j\left(x_{i,j}\right)\right)+\varepsilon_i$$

4.0 Replicating and extending the findings from the paper

Given the novelty of Ziel's research, there is no package/ library which contains an sBOA algorithm. As such, this project asserts the complexity of developing a custom function in R Studio meets the assignment requirements of replicating code. The primary focus of this document is therefore to create a robust function which accepts an ensemble of models and can produce weighted predictions which update based on a parsed horizon hyperparameter. Section 5.0 details how this project attempted to replicate the following:

- The log transformation of the electricity demand (load- kWh) and then applying a holiday adjustment
- Creation of ensemble models (STL, AR(P), GAM, LASSO) trained on different dataset sizes
- Creation of an MVP (minimal viable product) sBOA algorithm defined by:
    - A custom function which accepts arguments: ensemble of models, testing dataset, prediction horizon, name of the response variable, smoothing parameter (λ)
    - A custom function which returns: a data frame of each models' prediction for d,h (day and hour), a record of the updated weights for d,h,k (day, hour, model), a data frame containing the absolute error (loss) of each model at d,h
    - Returns no errors

Outside of the scope of this project was replicating: 1) the forward step-wise selection of K models to develop a final sBOA, and 2) producing forecasts that matched the time and dates of the IEEE competition. The reason for omitting the forward step-wise selection is that developing an sBOA algorithm was a complex enough task. Coding a forward step-wise selection which compares the optimal K for the lowest MAE added an additional time constraint which was not feasible given the project run time. The decision to avoid forecasts for 19/1/2021 to the 15/2/2021 was made because the site hosting the IEEE competitions training and testing data was missing observations < https://ieee-dataport.org/competitions/day-ahead-electricity-demand-forecasting-post-covid-paradigm>. It is important to discern this is not just missing observations in isolation, the data gaps introduced NA/NULL values in the lag load calculations which meant certain models in the ensemble (GAM and LASSO) could not accept the X matrix.

In addition to the above, the MXN442 task sheet had requirements of extending the research. This project suggests Ziel's work could be progressed by: 1) comparing the performance (MAE) of sBOA models built using different smoothing parameters, and 2) trialling substitution of the lowest contributing model type in Ziel's original ensemble for RNNs (Recurrent Neural Networks). Both extensions are supported by the conclusions of the paper: "*Slightly better results may be obtained by a more advanced tuning parameter selection design*" [pg. 211], "*Even though linear models designed with some degree of expert knowledge performed best adding high-depth non-linear models like gradient boosting machines or artificial neural networks may improve the results further*" [pg. 211]. Whilst the decision to trial different ensembles is suggested in the conclusion, this report explicitly suggests RNNs due to the assignment requirements (implement a model taught in during the course) and an understanding of their suitability to forecasting time-series.

In an RNN, each node (neuron) in a hidden layer receives inputs not only from the current input data x_t but also from its own previous hidden state h_{t-1}, updating it to $h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$ where W_{xh} and W_{hh} are weight matrices, b_h is the bias vector, and σ is the activation function (ReLU in this project). The algorithms architecture allows the network's layers to capture temporal dependencies through creating a memory of previous inputs, thus enhancing the accuracy in modelling a time-series with high autocorrelation (non-stationarity).

5.0 Methodology

As previously mentioned, the objective of this project was to produce an sBOA algorithm and test its performance by replicating the IEEE competition conditions detailed in the methodology of Ziel's paper. This is with the exception of a forward step-wise tuning of the model ensemble parsed to the sBOA. Section 11.0 of this report directs readers to a GitHub containing the code and data used to address the project focus: both serve as better descriptors of the methodology than can be expressed within this report's word limit. As such, this section preferences discussion on what is explicitly different between the output of this project verses Ziel's paper.

Ziel approached development of the sBOA using a 4-step structure: 1) data cleaning and preprocessing, 2) applying the holiday adjustment procedure, 3) training the individual forecasting models, 4) combing the trained models as an ensemble and producing forecasts using the sBOA algorithm. As previously mentioned in sections 3.0 and 4.0, Ziel and the IEEE data hosting page omit information on holiday adjustments, step 2, and missing dates, step 3; respectively. Regarding the holiday adjustments, the paper makes mention of amending load values for dates in October and November which exhibited holiday phenomenon. In the interest of producing a test environment that mimics the same training conditions Ziel developed the sBOA under, this project opted against speculating Ziel's

unspecified adjustments days. The same logic was extended to the missing dates (16/1/2021 to 17/1/2021) between the testing and training datasets: no triage/ imputation was completed.

The impact of not imputing the gap between datasets meant lag calculations were not possible for the original testing data. An executive decision was made to measure the total number of days the original testing dataset contained, 27, and then create a new training/testing set by subsetting the original training data via `tail(training_dataframe, 27days*24hours)`. This created a domino effect whereby the following were no longer possible: 1) comparison of MAE performance across the same dates as reported by Ziel and this project, 2) training models for the ensemble on a subset of 1123 days.

6.0 Resources

This project used R studio and a single script to produce the sBOA. All libraries have been detailed in the code which can be accessed using section 11.0. WARNING: the "keras" library utilises the tensorflow package and a python compiler to create RNNs. If your computer does not have this software installed, the code will fail on the first try until you use Git (or manually) to install it. Further instructions have been supplied in the "readme" file

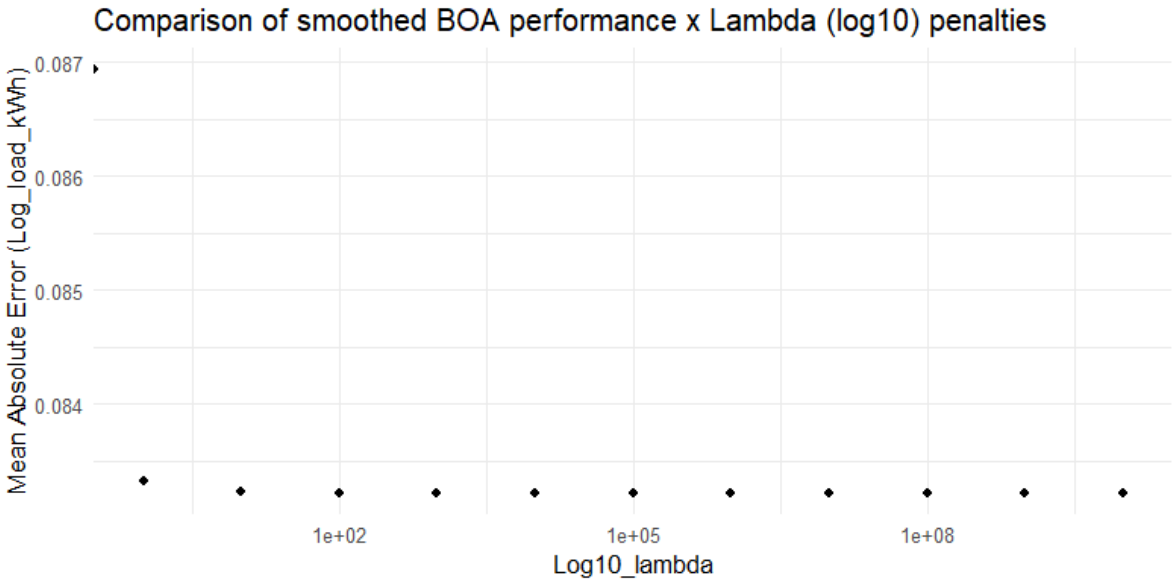
7.0 Results

The below metrics were produced using the custom function and methodology detailed in code for this project. A custom function called `Smoothed_BOA_algorithm()` was created which is noted to function without errors.

METRIC	sBOA MODELS		
	Pseudo Ziel sBOA	**Best Lambda sBOA**	**RNN Altered sBOA**
MAE (kWh)	1.0868	1.0868	1.1417
Computation Time	59.4332 seconds	58.4005 seconds	280.4743 seconds

Best performing model: sBOA(model list = c(STL, AR(P), GAM, LASSO), lambda = 1e+10) [Best Lambda]
Lowest computation model: sBOA(model list = c(RNN, AR(P), GAM, LASSO), lambda = 10) [RNN Altered]

7.1 Ideal smoothing parameters (λ) in the sBOA



Best performing sBOA: $\lambda = 1 \times 10^{10}$

7.2 RNNs vs AR(P)

sBOA MODELS	ENSEMBLE MODEL WEIGHTS				
	STL (w/ ES)	AR(P)	GAM	LASSO	RNN
Pseudo Ziel sBOA	0.0357	0.0357	0.0357	0.0357	n/a
RNN altered sBOA	n/a	0.0359	0.0355	0.0352	0.0363

Lowest average model class weight (Pseudo Ziel sBOA): STL

Lowest average model class weight (RNN altered sBOA): LASSO

8.0 Discussion

Firstly, we can assert that the primary focus of this project, creating a sBOA and testing it on the same data as Ziel's paper, was a success. Despite previously advising we could not compare Ziel's reported sBOA to the models produced in this report, it is still worth a brief mention in the context of verifying the code in section 11.0. Ziel's paper states their final sBOA had a MAE of 11.89 kWh over the course of the competition. The best performing ensemble in this report had a MAE of 1.0868 kWh. It would be erroneous to make a statement on which sBOA structure was the "best" as they were trained and tested on different datasets but the result undoubtedly shows the value of a model which leverages the law of large numbers (increasing the sample size increases the probability of accurately approximating the true value). After Ziel applied a forward step-wise procedure to their model ensemble, the final sBOA was only aggregating the output of 5 models (3 LASSO, 1 GAM, 1 STL). With this in mind, 60% of Ziel's model ensemble made predictions on a sparse list of model coefficients. It is easy to envision how a drastic change to x_i , a variable in common, could skew the sBOAs prediction at t. This creates the conditions for a larger MAE. The results of the differing MAE across the three sBOAs exhibit support for larger model ensembles over smaller K.

Moving on, the two extensions to Ziel's work (the other focus of this project) are seen to have been answered. Section 7.1 shows a clear difference in the performance of the sBOA if $\lambda = 0|1$. Whilst not detailed as part of replicating the publication, the visualisation in this section provides a comparison between the performance of a standard BOA ($\lambda = 0$) to the novel algorithm Ziel created. The standard BOA is seen to have a higher MAE (lower performance) than any sBOA. This result highlights the add value of using spline functions to update the weight vector: section 7.1 shows that preventing abrupt weight changes means sBOAs are more robust to single instances of volatile model performance verses BOAs.

Beyond setting $\lambda = 0|1$, Ziel's assertion of trialling different smoothing parameters arguably produces no practical gain in performance. The true difference between pseudo Ziel and best lambda is <0.0001 kWh MAE- this is despite a huge difference in how the weights are updated. This potentially infers all models in the ensemble describe the temporal electricity demand phenomenon with similar error: having harsher penalties for erroneous predictions produces no marginal difference than applying a lenient penalty.

The marginal difference in performance extends to the results described in section 7.2. The "RNN altered sBOA" sees lower performance than Ziel's original model ensemble. This echoes statements made throughout the publication that more complex non-linear models were not providing discernible improvements to the MAE. The difference in performance alone is negligible until comparing the computation time of the different algorithms: the sBOA containing RNNs took nearly 5x longer to calculate predictions than the other 2. Section 7.2 also shows that the predictive performance of "Pseudo Ziel sBOA" varies little across the four model groups: their average weights are identical when rounded to 4 decimals. However, this statement inverts when describing the average weights in "RNN altered sBOA". The RNN model predictions clearly have the largest contribution to the aggregated forecast which proxy represents their prediction accuracy. This result infers testing of a lesser smoothing penalty ($\lambda < 10$) might allow the RNN model to have a larger contribution to the extent the MAE of the sBOA is reduced.

9.0 Evaluation

This project supports Ziel's intuition of different lambdas producing different sBOA performance. However, the model ensemble Ziel uses suggests the marginal gain of this is limited. When juxtaposing this result with the range of computation times for sBOAs (1 to 5 minutes), the trade-off between improved accuracy and expended resources (time and energy) suggests only trialling two algorithms: $\lambda = 0 | \lambda \neq 0$.

In this same vein, building a step-wise selection for retaining/ omitting irrelevant models in the sBOA ensemble would drastically improve the repetitive testing of different model groups. Adapting the code in section 11.0 to be more equipped at trialling different parameters could extend Ziel's work by identifying the "best" ad-hoc hyperparameters. As Ziel mentions in the conclusion, they elected to use ad-hoc values for almost all models and this returned a high performing algorithm. Identifying ad-hoc values would allow for easier creation of a package in R studio or Python. The results of this report show the clear merits of applying an sBOA to a stochastic modelling task, it is therefore fair to state making such an algorithm open source would benefit any future data-driven decision making.

10.0 Conclusion

Ziel's article extended the traditional BOA algorithm to include a smoothing component. This report showed this adjustment to be practical at improving electricity forecasts and performance compared to the original algorithm. Whilst this report was unable to exactly copy the methodology of Ziel's paper (due to data discrepancies), a functional sBOA function was produced in R studio The results and discussion showed there were marginal differences in MAE when adjusting the smoothing parameter between sBOAs. However, a more meaningful difference was derived from trialling different model ensembles which identified appropriate selection of models had the greatest influence on predictive accuracy. The evaluation suggested this project could have benefited from including the forward step-wise selection for model ensemble and recommended developing the algorithm into a statistical package.

11.0 GitHub

https://github.com/TheRugbyDr/MXN442_Assessment2