413 Final Project Report

R Sangole

2018-06-12

Introduction

In this paper, I selected to approach the problem from a different angle that I usually do. In place of my usual approach of attempting to solve the problem using the training from this course and previous ones, I decided to investigate what published journal or white papers have done previously, especially trying to pick those papers which have reported higher scores in the competition. This approach allowed me to learn new techniques used by more experienced practitioners as well as extend my deep learning material I learnt in predict 453 few quarters ago.

This paper is organized as follows.

Secion I deliniates the overview of the methodologies used, the papers referenced and the challenges faced at a high level. It also explains some technical challenges faced.

Section II explains some of the exploratory work done.

Section III explains the data preparation activities.

Section IV outlines details of the Method Of Analogues model.

Section V outlines details of the deep learning models.

Section VI talks about the Bayesian Regression model.

Section VII wraps up the paper.

Section I - Overview of Methodologies Used

I quickly read a plethora of published papers, white papers and class notes on this problem set. The difficulty of the problem revealed itself since almost everyone had used a different approach to solving the problem. Folks have attempted to solve this using everything from ensembled linear models to non-linear deep learning approaches to heuristic computational methods. I chose two papers to try and replicate. Both papers used methods not taught in the northwestern courses, but built upon techniques already taught in the courses so far. While I realized that trying to replicate a paper an entire paper created by a professor with his 3 PhD students within a span of a few weeks is not easily possible, I was determined to try. If nothing, I would learn new methods which I can apply at work.

The first paper [1] is an ensemble model of three sub-models: 10s of Method Of Analogue (MOA) models, 1000s of Additive Holt Winters models and naive models, with a novel median-voting based weight scheme. The MOA [3] is a method invented in 1969 for prediction of weather. It is widely used in meterological model building, and has been used for influenza prediction as well.

Since there is no pre-written package in R for this method, it required me to chase down the mathematically nitty gritties [4] in a few papers and implement my own version of the model. There are many versions of MOA depending on the search algorithm, or the analogue selection algorithm. I studied a few of them, and decided to implement the simplest version. I could not implement the paper as is, with the main constraints being computational time required to solve these search based models iterative models on such a large forecast horizon.

The second paper I read relied on an ensemble of linear regression, weighted linear regression, and Bayesian regression models Out of these, I decided to learn a bit about the Bayesian model.

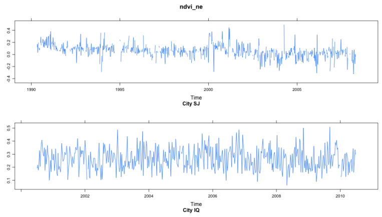

Figure 1: Lack of seasonality in NDVI NE

The third model I decided to investigate is a Recurrent Neural Network (RNN) model, specifically the Gated Recurrent Unit (GRU) and the Long Short Term Memory (LSTM). These models were ones I was looking into at the end of the Predict 490 (Deep Learning) course. This was my first foray into these recurrent models.

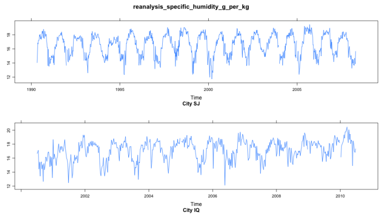

Figure 2: Seasonality in Spec Humidity

Section II - EDA

UNIVARIATE STUDIES Time series plots were run for all the variables to get an idea of the underlying structure. While some signals don't show strong seasonal patterns like in figure 1. Others show very strong seasonality, like in figure 2. Depending on the chosen solution, this is useful information. The response variable `total_cases` shows the peaks and available information for the two cities. Note teh different time scales on the x-axis.

MULTIVARIATE STUDIES (LINEAR CORRELATIONS) Linear correlation study between the Xs and Y for the two cities show remarkable difference between the cities, along with some key insights into the underlying structure of the data. Some key highlights:

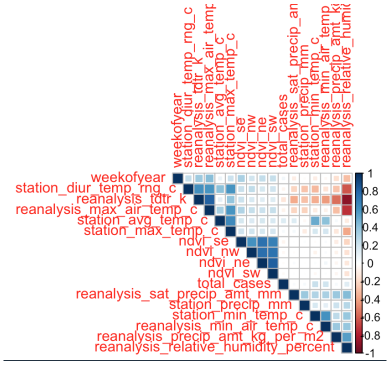

Figure 3: Cor Plot IQ

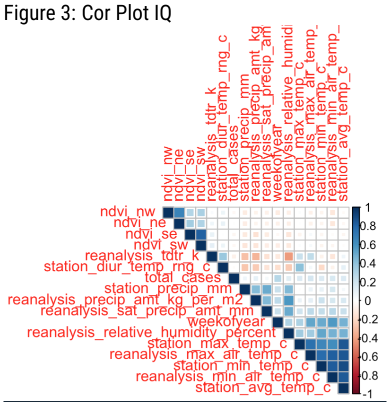

Figure 4: Cor Plot SJ

- `total_cases` is very weakly correlated (if at all) with any of the Xs. Doesn't let itself to a simple way of predicting the values. It's weakly correlated with the `weekofyear` variable, which makes sense. When it's hotter, and wetter, there is a higher chance of dengue.

- SJ's corrplot shows us that almost all the correlations are positive, it at all. As expected, all the vegetation indices are correlated positively. As are all the temperature related variables. Further investigation using PCA showed me that for these variable groups, at max 2 PCs were needed to achieve ~97%+ of explanatory power for the variation in each group.

- IQ's corrplot has a few strong negative correlations, especially with the `tdtr` variable, which explains the daily temperature fluctuation. When it's very hot, or very humid, there is less temperature variable over the day. Given IQ's geographic location, perhaps this makes sense from a weather dynamics standpoint.

Section III - Data Preparation

The summary of the data preparation activities is:

Imputation

The few variables which had missing values were imputed using linear approximation using the `zoo:approx` function. There are two spots in which 14+ rows were missing in a row. To preserve seasonality, seasonal (weekly) averages were used from neighbouring (+/-3 weeks) were used.

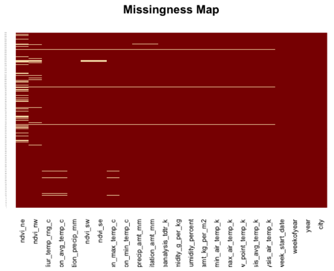

Figure 5: Missingness Map

UNIT CONVERSIONS All Kelvin temperatures were converted to Celcius to be on the same scale.

DATA STANDARDIZATION For models sensitive to scale of Xs (like RNNs), the data was scaled using the MinMax scaling.

FEATURE ENGINEERING No feature engineering was performed.

Section IV - Method of Analogues

The MOA [3] is a method invented for prediction of weather. There are many versions of MOA depending on the search algorithm, or the analogue selection algorithm. I attempted at the simplest version. Conceptually, this is how the model works:

1. If we want to predict the forecast for (t0+h) at time (t0), we first look back a certain length (L). So we select observations (t0-L), (t0-L+1),... (t0-1). [Green bubble]
2. Then, we search for all possible combinations of length L in the history of observations which are analogous to the recent history. [Blue bubbles]
3. The 'level of closeness' can be measured by eucledian distance (among other measures)
4. For the top few (V) closes such analogues, we find the (t+h) forecasts. Thus we will have (V) such numbers.
5. The desired forecast (t0+h) can be a mean/median of these (V) numbers.

Graphically, it can be shown as:

METHOD DETAILS

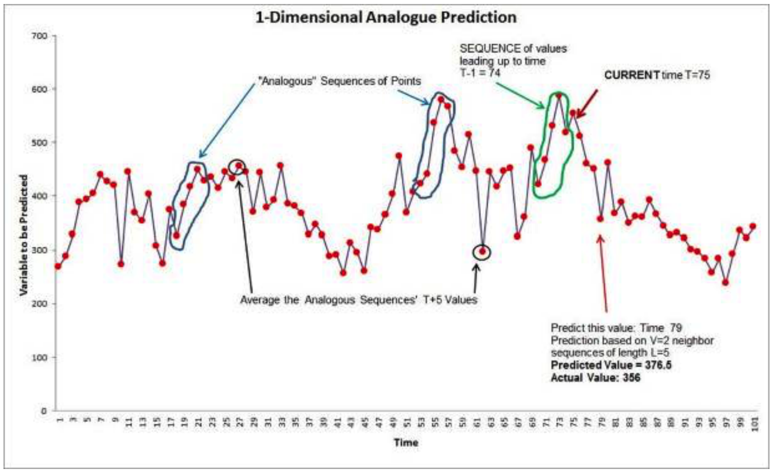

Figure 6: MOA

The paper [3] uses a two dimensional eucledian distance computed using the response `total_cases` and one other variable (X_unknown). This other variable is computed using transfer entropy.

Transfer Entropy Transfer entropy is a non-parametric statistic measuring the amount of directed (time-asymmetric) transfer of information between two random processes. Transfer entropy from a process X to another process Y is the amount of uncertainty reduced in future values of Y by knowing the past values of X given past values of Y. I used the package `TransferEntropy` to calculate the TE values. There are two tuning parameters to search for: Which X_unknown to pick, and what the correct value of the lag of X_unknown is, such that it transfers the most amount of information to Y. I performed a grid search across all X values, and lags of {1, 2, . . . 53} weeks to find the appropriate X_unknown & lag. For San Juan `station_precip_mm` at a lag of 8-10 weeks shows the most transfer of information to the total cases of dengue. This could be linked to the life cycle of mosquito.

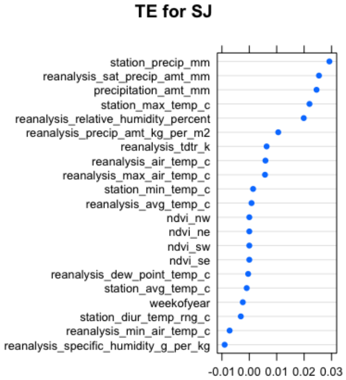

Figure 7: TE Variable Search

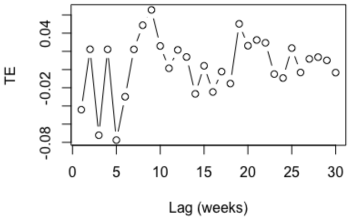

Figure 8: TE Lag Search

Computational Details

This is where things got challenging. The original paper implements 100s of such MOA models, for each city, for each forecast horizon {h=1,. . . h=52}, and then performs a median-vote based ensemble calculation on these forecasts. The challenge is that a single model took anywhere from 10-20 minutes to run on my single core laptop because there are two nested for loops. In each loop, there is a exhaustive search across all historic data points + a distance calculation. Unfortuntaely, since this is a dynamic model, these loops cannot run in parallel using techniques like `foreach...%dopar%`. As a result, even tuning one model's parameters proved computationally very expensive.

Given enough time, one could port the code over to a faster search and compute in a lower level code like C to allow building large number of models.

Internal Test Set

Running this MOA model on a 70-30 train test split, we can see the weakness of running a single model. The red line shows the original data with the spiky behaviour, while the black line shows a single model's results on a dynamic forecast. [That is, t_hat(h+i) uses the forecasted value of t_hat(h+i-1), . . .].

You can see that where the model uses the total cases from the training data, i.e. known good values of total cases, the method works remarkably well, since essentially it's doing only a 1-step-ahead forecast. But, as soon as we hit the test dataset, error starts compounding and the forecasts are off.

RMSE = 28.62 MAE = 3.73

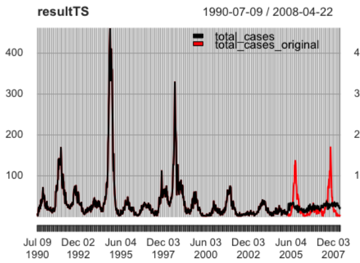

Figure 9: TE Lag Search

Drivendata.org Test Set

On the website, this model scored *29.94*. Not that great, but expected to be a poor score since it doesn't leverage the strengths of being an ensemble model.

Section V - Deep Learning Models

In my 490 Deep Learning class, the focus was on image classification using CNNs. I wanted to take my knowledge further into RNNs and LSTMs since they have direct application in my day to day job.

So, for this project, I learnt how to install and get Keras + Tensorboard running on my laptop, as well as get an AWS cluster with a GPU setup. Ultimately, I built many models on my laptop, given the expenses of the AWS GPU compute power.

RNNs are neural networks which have some `memory` since they pass on the earlier test state to the next calculation in the time series. GRUs and LSTMs are special instances of RNNs which get progressively more sophisticated in their ability to learn the past. LSTMs are able to learn long and short term memories of the time series, and also learn when to forget something from the past.

LSTM graph tree

I attempted various versions of the GRU/LSTM models, including such configurations as:

- Single vs stacked layers
- Number of units in each layer
- Drop outs
- Recurrent drop outs

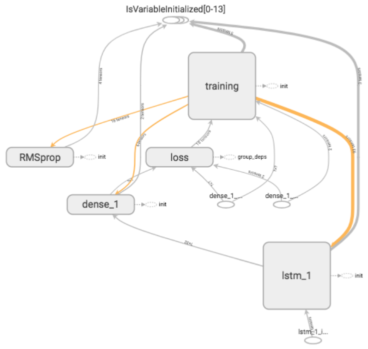

Figure 10: Single layer LSTM architecture

Internal Test Set

A MAE on the validation set was around 13.89. The figure shows an example of the training and validation results. While training error keeps decreasing, the validation error stabilizes at around 0.225.

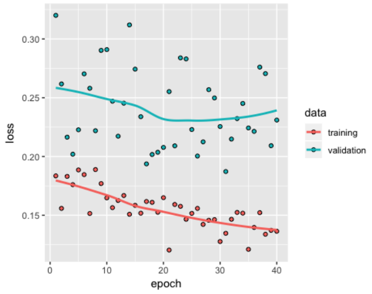

Figure 11: Single layer LSTM architecture

Drivendata.org Test Set

Depending on the configuration, the results score around *29.07*, a slight improvement in the Section V model.

Section VI - Bayesian Regression Model

This was an attempt at replicating a bayesian ridge regression model, which I absolutely did not understand, so not reporting anything here!

Section VII - Wrap Up

1. The MOA model shows promise for one-step-ahead (or few-steps-ahead) forecasting, as long as there is enough history of the multivariate signals. Further investigation into methods of identification of analogues (PCA, Mahanobalis distance etc) might prove useful. Also, the search algorithm is expensive as implemented using for loops in R. Moving a lower level language like C++ might show execution speed improvements.

2. Similarly, for the GRU/LSTM models, the lack of compute power was severly limiting. A dedicated GPU machine would help converge results without paying AWS a lot of money.

3. Ensemble models seem to be the way to go based on a few papers. This would be the logical next step towards building a workable model.

References

1. Buczak, A. L., Baugher, B., Moniz, L. J., Bagley, T., Babin, S. M., & Guven, E. (2018). Ensemble method for dengue prediction. PLoS ONE, 13(1), e0189988. http://doi.org/10.1371/journal.pone.0189988
2. Vicente, R., Wibral, M., Lindner, M., & Pipa, G. (2011). Transfer entropy—a model-free measure of effective connectivity for the neurosciences. Journal of Computational Neuroscience, 30(1), 45–67. http://doi.org/10.1007/s10827-010-0262-3
3. Viboud C, Boelle P-Y, Carrant F, Valleron A-J, Flahault A. Prediction of the spread of influenza epidemics by the Method of Analogues. American Journal of Epidemiology 2003; http://10.1093/aje/kwg239
4. Lorenz E. Atmospheric predictability as revealed by naturally occurring analogies. Journal of Atmospheric Science 1969; 26:636–646 1.https://github.com/amschwinn/dengue_prediction

5. https://cran.r-project.org/web/packages/TransferEntropy/TransferEntropy.pdf
6. https://ral.ucar.edu/sites/default/files/public/images/events/WISE_documentation_20170725_Final.pdf
7. https://tensorflow.rstudio.com/keras/
8. https://tensorflow.rstudio.com/blog/time-series-forecasting-with-recurrent-neural-networks.html
9. https://en.wikipedia.org/wiki/Transfer_entropy