WALMART STORE FORECASTING

Atom Group: Jifu Zhao (jzhao59), Jinsheng Wang (jwang278)

Nuclear, Plasma, and Radiological Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801, USA

1. INTRODUCTION

In this project, our goal is to predict the weekly sales for each department in each store for Walmart, initiated by Kaggle three years ago. The original training data include the historical data from Feb 2010 to Feb 2011 while the test data start from Mar 2011 to Oct 2012, with totally 20 months. In each prediction iteration for each month in test data, we will be given all the previous historical sales as training data, then we are asked to predict the sales for this month. After one prediction, the true sales value will be added to training data and then again we will predict for the next month. This process will repeat untill all the sales in test have been predicted.

The data set contains multiple features, including Store, Dept, Date, Weekly_Sales, IsHolday. After some initial analysis, we find that there are 81 unique departments and 45 unique stores. In this project, we first explore the given training data set. After some pre-processing methods, we applied three stepwise improved models to predict the next month's sales. More details will be described in the following sections.

2. PRE-PROCESSING

After exploring the training dataset, along with other's experience, we converted the time to week number in a year, which if well built, can greatly simplify the data features for algorithms establishment. There is only 52 weeks in a year, when we set 2010-02-05 as week 5 then each year will have the same week number. Besides, year and month information can be extracted from original time information, which can help us to locate training data information we need to use in the first model. Also, month and year will be used as loop index in algorithm to cycle through all the Weekly_Sales that have to be filled.

In our R code, we also loaded three libraries to simplify our task, including lubridate, forecast, plyr. The first one is for time conversion and the last two have functions to perform more complicated model prediction.

3. METHODS

3.1. Average from Nearby Weeks

In our first model, Model 1, we basically choose the nearby three weeks of previous year to make prediction. For example, for a given store s and department d in week w, we locate the corresponding data of the same department d and store s from the previous year, then choose the median of the sales from week w - 1, w, w + 1 as the prediction. If the data are missing for the previous year, which could be because the department has not opened yet one year before, then we assign this prediciton as 0. The same principle applies to NA values or missing data scenarios.

3.2. Time Series Forecasting

In Method 2, we choose ts() to create a time series whose frequency is 52 and auto.arima() to forecast for a give step. More specially, for a given store and department, we first extract the data from previous years that is from the same department and store. Then we use the ts() function to create a time series whose frequency is 52. Finally, through feeding the time series into auto.arima(), we make the prediction that corresponds to next month's sales. One trick here is that ts() seasonality model requires the data should show at least two complete period cycles, which means we could only use seasonality after Feb 2012. This proves not good as it can not make the best of all the training data. Our strategy is that if the total available previous weeks for certain store and department is less than two years, we will assign prediction with values from Model 1. This combination method turned out to be better than typical method using seasonality only after 2 years, which performed badly from Mar 2011 to Feb 2012 due to lack of two complete cycles.

3.3. Ensemble of Weighted Model 1 & 2

Considering the long running time of Model 2, in this model, we choose the weighted average of Model 1 and Model 2 as our third model, Model 3. After some careful study, we found the the best weight should be 0.7 for Model 1 and 0.3 for Model 2. The final expression for Model 3 is shown below:

$$Model\ 3 = 0.7 \times Model\ 1 + 0.3 \times Model\ 2 \quad (1)$$

4. CODE DESCRIPTION

All of our code is contained in the file named mymain.R. There are basically three parts in the R file. At the very beginning, the code will automatically check whether or not the required packages/libraries are already installed. In the second part of the code, we do data preprocessing: transform the data information. For the last part, we mainly build a function with three models: Model 1, simple average; Model 2, time series forecast and Model 3, ensemble of weighted Model 1 & 2. These built models will make predictions and the results are saved as required by the project description.

Due to the amount of data to be looped, the code need a lot of time to run. As tested, the total running time is around 2 hours.

5. RESULTS

To evaluate our model, we choose the metric described on Kaggle:

$$WMAE = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i |y_i - \hat{y}_i| \quad (2)$$

where n is the number of test cases, $\hat{y}_i$ is the predicted sales, y_i is the actual sales and w_i is the weights. For this project, we set $w = 5$ if the week is a holiday and 1 otherwise. The final WMAE for three models are shown in Table 1.

Table 1. Summary of Models

Model	WMAE
Model 1	2093.603
Model 2	2395.633
Model 3	1956.629

From Table 1, one can find that, Model 1 and Model 2 performs similar to each other. Through weighted averaging, the final model, Model 3, performs better than Model 1 and Model 2.

Acknowledgement

The authors would like to thank Xichen Huang for his tutorial notebook on Piazza and David Thaler for his online code.