Project 2: Walmart Store Sales Forecasting

Tyler Zender (tzender2)
Alejandro Pimentel (ap41)
Matthew Lind (lind6)
CS598 Practical Statistical Learning, Fall 2022
November 14, 2022

Introduction:

This project utilizes the dataset from the Walmart Store Sales Forecasting Kaggle challenge. The goal was to predict future weekly sales of each department of every store based on historical sales data.

The data contained weekly sales for 45 Walmart stores spanning 81 departments arranged in 5 columns (Store ID, Department ID, Date, Weekly Sales, IsHoliday) with each row representing one week's sales of one department in one store. This project required our model utilize these features to produce an optimal set of predictions, but prediction was complicated by challenges in the data. For example, weekly sales were not reported for every week of the year for all stores and departments. There were also 5 holiday weeks identified which did not align on the calendar when comparing the corresponding weeks across years.

The initial data (train_ini.csv) spans the dates February 2010 through February 2011, with additional data through October 2012 available in 2-month folds (fold_1.csv, fold_2.csv, …). Predictions are made for the two months immediately following the training data. After each prediction, the date range is incremented by 2 months while the next fold is appended to grow the training data set. For example, after the first set of predictions the training dates were modified to be February 2010 to April 2011 and sales data for March and April 2011 appended to the training data. This process was repeated for 10 folds.

For each fold, the weighted mean absolute error (WMAE) was computed:

$$WMAE = \frac{1}{\Sigma w_i} \sum_{i=1}^{n} w_i \left| y_i - \hat{y}_i \right|$$

where:

- n = number of data points
- $\hat{y}_i$ = predicted sales
- y_i = actual sales
- w_i = weight (5 for holiday dates, 1 for non-holiday dates).

WMAE was the metric employed to determine how well predicted sales match the actual sales. Lower score is better.

Pre-processing:

The testing procedure for this project is centered around our function **`mypredict()`** which is responsible for organizing data and producing predictions. **`mypredict()`** accesses the training data, test data, and fold index via global variables (**`train`**, **`test`**, and **`t`** respectively) supplied by the calling evaluation code. The evaluation code was provided by course instructional staff, and suggestions provided in a series of discussion forum posts entitled "What we have tried".

We filter the Weekly_Sales feature by replacing negative values with zero. One could argue negative values represent refunds or customer returns and not clerical errors, but we wanted to focus on sales.

Inspection revealed sales figures were varied, but the general sales pattern within each department was similar across stores. Therefore we leveraged the truncated Singular Value Decomposition (SVD) to smooth training data using our function **`get_train_svd()`**. The data was organized by department, each with a matrix formed from rows representing different stores and columns containing dates where weekly sales data was provided. In the event weekly sales were not available, the missing values were replaced with zeros. Matrices were centered so the mean weekly sales of each row was equal to zero, the SVD was applied with a specific number of principal components for each fold, then the subtracted row mean was restored. The resulting smoothed matrices are stacked into a larger matrix and returned to `mypredict()`.

The smoothed data is manipulated to be more suitable for a model. Dates in the year 2010 were shifted one week earlier to compensate for days (e.g. holidays) appearing in different weeks across calendar years because sales are recorded on Fridays to mark the end of the week, not Saturdays as is done on the calendar. Dates are then translated to dummy variables of 52 distinct columns representing each week of the year, and a column for the year. Interaction terms for the year are also created with columns for the 2nd and 3rd degree polynomials as it was suspected sales may not follow a strictly linear pattern due to compounding factors like inflation. Thus adding these terms would improve the model. Since the complete data set includes dates spanning years 2010 through 2012, it is possible to train with data from 2010 and 2011 to make predictions for 2012. We artificially restrict prediction year to maximum year found in the training data to ensure predictions are made only from data in the model which has been trained. The feature 'IsHoliday' from the data set is not used in the design matrix.

Model:

We use a linear regression model as defined by the `lm.fit()` function in the R programming language. `lm.fit()` was chosen due to its efficiency compared to the more commonly used `lm()` function when fitting large quantities of small models. Coefficients are obtained from this model and applied manually to test predictors.

The model includes predictors for the 52 distinct weeks of the year expressed as dummy variables, the year, and second and third degree polynomials for the year. The year variables are numeric. The model is applied to each unique store/department combination common to both the training and test data in the prediction interval.

Results:

Computations were performed on a Dell Precision Workstation 7820, 48 GB RAM, Intel Xeon Gold 6242R processor @ 3.10 GHz (20 cores / 40 logical), Nvidia Quadro RTX 4000 GPU w 8 GB VRAM, Microsoft Windows 10 professional operating system and R version 4.2.1.

Ten folds were computed with WMAE and total time recorded for each method as shown in the following table:

	WMAE		
Fold	**Naive Linear**	**SVD fixed (d)**	**SVD variable (d)**
1	2045.243	1941.718 (8)	1870.968 (3)
2	1466.912	1363.279 (8)	1363.279 (8)
3	1449.852	1382.283 (8)	1379.315 (11)
4	1593.998	1526.783 (8)	1526.783 (8)
5	2324.496	2310.182 (8)	2305.561 (11)
6	1669.102	1625.125 (8)	1621.560 (6)
7	1646.078	1613.820 (8)	1611.169 (11)
8	1365.314	1355.041 (8)	1340.341 (14)
9	1358.247	1336.934 (8)	1321.609 (14)
10	1344.849	1333.951 (8)	1319.525 (13)
Mean	**1626.409**	**1578.912**	**1566.011**

Total Time (s)	62.86	148.06	149.08

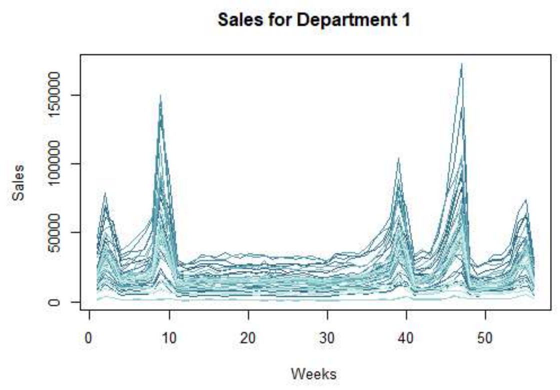
The naive linear method (no SVD pre-processing) produced a WMAE of 1626.409 in just under 63 seconds as a baseline score.

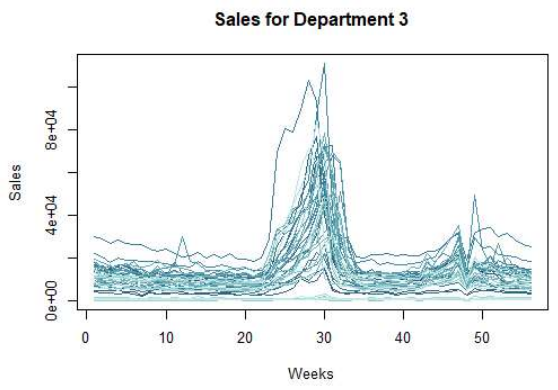
The *SVD fixed* method applied SVD to the data choosing the top *d* number of components. The number was selected by inspecting results of mean WMAE when test runs of SVD size 2 through 14 top components were applied to the data. While SVD introduced significant additional computation, it also improved the result considerably.

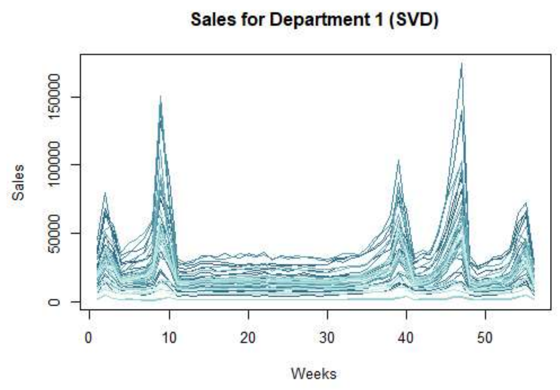
The *SVD Variable* method chooses the SVD's top *d* number of components in each fold according to the value producing the best WMAE per fold from test runs of the *SVD fixed* method. This further improved WMAE with negligible additional computational cost while minimizing effort required to implement as it's derived from work already performed. While not predictive, this method demonstrates enough improvement that a model which varies d is worth consideration.

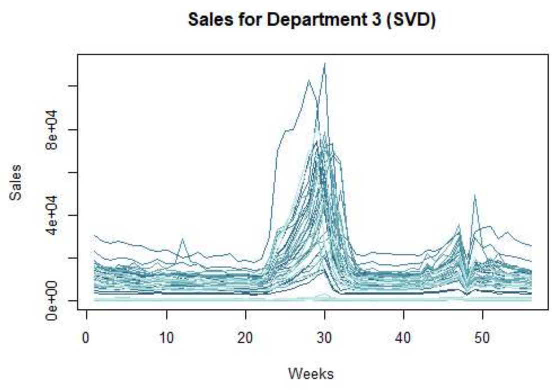
Discussion:

The use of SVD to produce a smoother version of the training data showed improvement in all data folds. In other words, SVD acted as a means to reduce anomalies or noise in the data. Below we show the sales for departments 1 and 3 for the first fold before applying SVD (top) and after SVD was applied (bottom).









In the graph each line represents a store. We can see they follow the same pattern. Although it may be difficult to discern whether the lines are smoother after applying SVD, the linear model performed better with this transformed data.

In terms of execution time it was very important to do a one-shot transformation on stacked data when possible as opposed to transforming smaller chunks of it inside loops.

Employing 2nd and 3rd degree polynomials for the year was questioned during development, but showed improvement of nearly 32 points when implemented and tested with the SVD variable method.

Removing negative values from Weekly_Sales and replacing them with zeros improved WMAE, but only slightly. This is likely due to the fact most of the negative values were already very close to zero.

Contributions:

Alejandro Pimentel: Setup of evaluation code, as well a number of earlier models, implementation of SVD, written report.
Tyler Zender: Assistance with SVD, implementation of interaction terms and 'year limiting', written report.
Matthew Lind: SVD component optimization, code testing and verification, data generation, written report.