High Performance Computing in Business: A Case Study Collaboration Among A National Retailer and A University

Hemanya Tyagi, Geoffrey R Lentner, Matthew A. Lanham
Purdue University Krannert School of Management
tyagih@purdue.edu; glentner@purdue.edu; lanhamm@purdue.edu

Abstract

We developed a machine learning workflow using a high-performance computing (HPC) architecture available at the Rosen Center for Advanced Computing (RCAC) at Purdue University with a nation-wide grocery retailer to demonstrate how the retailer could perform data science experiments at much greater scale, efficiency, and cost compared to using their current design and tool.

The motivation of our study is that many retailers continue to interface relational database management systems with analytics tools (R, Python, SAS) to train and evaluate models. The tools might be local, server-side, or cloud-based, but using our modeling design workflow with RCAC's GPU architecture could run models many times faster which would allow the grocer to identify the best forecast for their SKUs more accurately leading to improved planning and replenishment decision-support.

Introduction

Most data science problems follow a workflow which is described in Figure 1. This primarily includes querying data, cleaning, modeling, deployment. and monitoring. However, there are times when parts of this process or the whole process need to rerun to incorporate changes in the data or modeling. For small datasets and simple problems, this is fine, however, for large datasets and complex problems, this poses challenges.

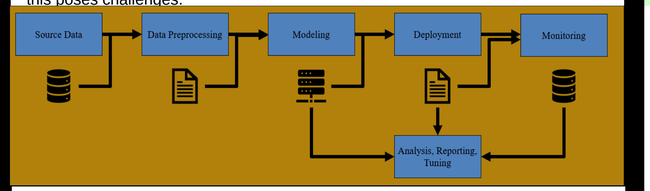

Figure 1. Generic Workflow in Data Science

To solve this problem, we develop a workflow that is automated and intelligently executes the modeling steps. Also, we aim to utilize the parallel processing capabilities of the Brown and Gilbreth clusters to provide efficient solutions. We show how to set up massive data sets, partition those datasets efficiently to train models using RCAC's Gilbreth/Brown cluster, save the models, and evaluate the model's performance for hundreds of SKU groups and predictive model combinations. We provide runtime using Gilbreth versus estimated runtime provided by the retailer's system, as well as demonstrate how having the ability to run hundreds of models led to more questions on where to design and run future experiments to support their business needs. The primary research questions we address are:

How to design efficient, automated and scalable pipelines for data science problems?

What is the impact of parallel processing and GPUs in a data science workflow, specifically measuring the runtime improvement?

Literature Review

The business problem translates into a data problem in which a time-series model needs to be built to predict demand. Prior studies have been conducted in this field with good results. However, there is little research on the use of high-performance computing in such problems.

Author	Study	Algorithms	Comments
G. Peter Zhang	Time series forecasting using a hybrid ARIMA and neural network model	ARIMA, ANN, Hybrid	This is one of the baseline studies in the field of time-series analysis for forecasting. It runs ARIMA and ANN and finally a hybrid model to incorporate the positives of both the model.
Nikolaos Kourentzes	Intermittent Demand Forecasts with Neural Networks	Bi-variate Neural Networks	This study shows how Neural Networks overcome the limitations of other models such as Croston's methods.
Ajla Elmasdotter, Carl Nyströmer	A comparative study between LSTM and ARIMA for sales forecasting in retail	LSTM, ARIMA	This study concludes that the LSTM model is promising in the field of sales forecasting in retail and able to compete against the ARIMA model.

Table 1. Literature Review

Methodology

Figure 2 outlines our study design. Each step is distributed amongst n processes. For our problem, we used n = 24.

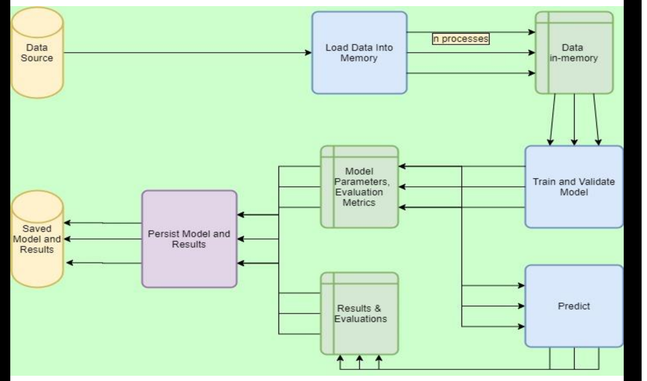

Figure 2. Study Design

Data, Feature Engineering, and Pre-Processing
Product demand data was provided by a major U.S. grocery chain. Models were run for products for which data was available for at least 100 days. Many of the products were discontinued during the length of duration of the study. Such missing data were removed to improve model learning and forecasting. Train-test split was taken to be 80-20.

Model Choice
As it is easy to process batches and scale using Keras, LSTM was the model chosen.

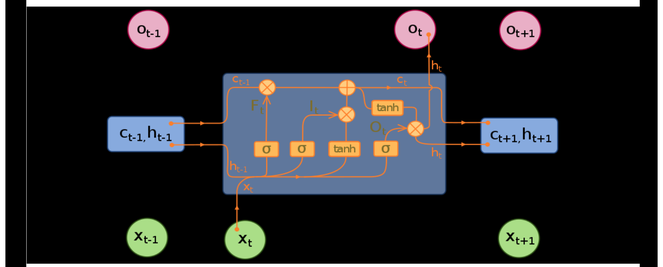

Figure 3. LSTM Architecture

Methodology
To automate the workflow for the time-series problem, we used a Makefile-based approach in which every process would be dependent upon the previous process. Thus, if a change occurs at any point inside the workflow, it will trigger the workflow to rerun at that point and all subsequent processes will sequentially execute.
Furthermore, using Makefiles provides an additional benefit in utilizing the processes of the node on which we are computing. Hence, this approach parallelizes the tasks as well.
The first step in the workflow is to fetch the data. Due to the large data size, data is fetched and stored in memory by using multiple processes. Then, the model is trained and evaluated, parallelized for each product ID. Finally, the model parameters and metrics are stored in the database for further analysis and use.

Model Evaluation
The evaluation of our model is not only based on accuracy parameters but also on the runtime.
To measure the accuracy of the model, we used the Systematic Mean Absolute Percentage Error (SMAPE). We also visualized the fit of our model on the validation data and finally analyzed the runtime per model and the runtime of the whole process

Results

1907 products were analyzed and for the ones which had more than 100 days of data, LSTM was run to provide results showed in Figure 4. For most of the model instances, the SMAPE was well under 50%, which shows that our model fits well for most of the SKUS.

1907
Total SKUS

1702
Models Run

35.89%
Median SMAPE

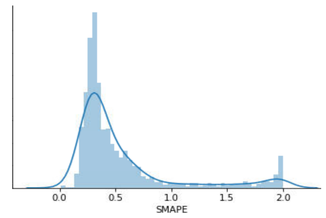

Figure 4. Model Evaluation

Figure 5 exemplifies a model outcome on the validation data of a SKU. The prediction of this model results in a SMAPE of 31%.

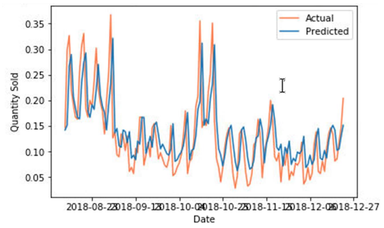

Figure 5. Model Fit Example

Figure 6 shows the distribution of the time taken to run each model. Considering the average model run time, it would to take more than 20 hours without parallel processing to run models for all the SKUs

57 Minutes
Total Run Time

42 Seconds
Mean Model Run Time

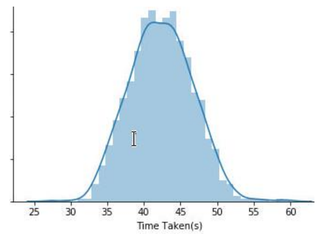

Figure 6. Runtime Performance

Conclusions

With the amount of data increasing exponentially across various industries such as retail and e-commerce, it becomes crucial to apply scalable and robust machine learning and deep learning pipelines. Leveraging the parallel processing capabilities of platforms such as Brown and Gilbreth at Purdue by using Makefile-based approaches can help improve the runtime significantly. Automating model building workflows provides reproducible and robust results. Furthermore, pinpointing any errors during the modeling process becomes easier with this approach.

Acknowledgements

We thank Professor Matthew Lanham for constant guidance on this project.