Overcoming the computational demands of time series: Scaling R-based demand forecasting with RapidMiner

2/12/2020
Goal
Provide Supply Chain with highly accurate, highly scalable food demand forecasts

Problem
Shared resources limited; ecosystem of projects expanding rapidly

Solution
Make extensible open source time series forecasting tool; think creatively to keep footprint small
Store Inventory Lifecycle

**Hungry Customer**
Orders food, and depletes Store’s inventory

**Supply Chain**
Fulfills Store’s food order, and restocks Store’s inventory

**Store Operator**
Counts inventory, and orders replenishment

This stream of data can be mined for insights
Highly Accurate, Highly Scalable Demand Forecasts

Business Value

Improve Supplier Relations
Enable timely, and accurate purchase plan to suppliers

Reduce Food Waste
Avoid food spoilage

Scale Labor to Demand
Avoid idle labor and overtime
Available Resources

50+ Team Members

Many with advanced degrees:
- PhD Chemistry, PhD Computer Science, PhD Physics, Masters in Applied Statistics,
- Masters in Electrical Engineering, Masters in Epidemiology, Masters in Industrial and Operations Engineering

Comprehensive Tech Stack

**User Desktop:** RapidMiner Studio, R, Python, Jupyter, SSMS

**AI/ML:** RapidMiner, Jupyterhub, R Studio, Nvidia GPU Server, ArcGIS, Hive, Spark

**Data Stores:** Sql Server, Hadoop

**RapidMiner**
Prototype

**RapidMiner**

Read demand history, promotion history, planned future promotions, & important holiday dates from database

Pass as inputs into the R implementation of Facebook’s opensource timeseries forecasting package prophet

Write results into downstream applications
Prototype – (important bits of) the R Script

```r
# function to forecast
runProphet <- function(data, SCC, sku, holidays, future) {
  df <- data %>%
    filter(SCC_NUMBER == SCC & INVENTORY_CODE == sku) %>%
    select(INV_DATE, IDEAL_USAGE, <add any external regressor here>) %>%
    arrange(INV_DATE) %>%
    rename(id = INV_DATE, y = IDEAL_USAGE)

  m <- prophet(holidays = holidays, growth = "linear",
               interval.width = 0.95,
               daily.seasonality = F,
               weekly.seasonality = T,
               yearly.seasonality = T)

  m <- add_regressor(m, <add any external regressor here>)

  m <- fit.prophet(m, df)

  forecast <- predict(m, future)

  output <- data.frame(INV_DATE = forecast$ds,
                        SCC_NUMBER = rep(SCC, nrow(forecast)),
                        INVENTORY_CODE = rep(sku, nrow(forecast)),
                        Forecast = forecast$yhat,
                        Lo_95 = forecast$yhat_lower,
                        Hi_95 = forecast$yhat_upper)

  return(output)
}
```

R-script receives from RM three inputs retrieved from sql:

1. 3-yrs history of demand
2. Forecasting period (i.e., 8-weeks of future)
3. List of important holidays

Forecast function filters to a single scenario
(Supply Chain Center-SKU)

Forecast function defines prophet model, fits it, & forecasts demand, by week, for the next 8-week period

Forecast function is wrapped within a doParallel process to use 16 cores, concurrently

https://facebook.github.io/prophet/
https://github.com/facebook/prophet
**Timeline of Enhancements**

- **Launch Prototype**
  - 1 VM (single queue)
  - 200 forecasts
  - 8-hour run-time

- **Stage Inputs**
  - 7+ hours run-time for training history
  - SELECT
  - Remove cursor-based outlier replacement
  - Remove self-joins, replace with windowing function
  - Perform history aggregation once, save for later use
  - 1 VM (single queue)
  - 200 forecasts
  - 15-minute run-time

- **Hyperparameters**
  - Thesis: single set of hyperparameters exists with performance better than default
  - Use Grid Search + Bayesian Optimization
  - Parallelize Grid Search over 6 VMs
  - 10-hours elapsed vs. 60-hours
  - MAPE improved from 6.5% to 6.23%
  - 1 VM (single queue)
  - 4,000 forecasts
  - 8+ hour run-time
  - Staged Inputs db footprint > 150 GB

- **Expand Workload**
  - 15-minute run-time
  - 4,000 forecasts

- **CCI**
  - Stable results encouraged business
  - Added SKUs & Supply Chain Centers
  - Data Volume & Compute needed expanded 20x
  - Replaced Heap + Nonclustered Index with Clustered Columnstore Index (CCI)
  - Event-based trigger leaves slack in system
  - 6 VM (3 queues)
  - 4,000 forecasts
  - 1.3 hours run-time

- **Event-based Trigger**
  - Time-based launch
  - Use all available VMs to run mutually exclusive segments of workload
  - Event-based trigger runs instantly after all dependents complete
  - 6 VM (3 queues)
  - 4,000 forecasts
  - 27-minute run-time

- **Parallel, Parallel Execution**
  - Use all available VMs to run mutually exclusive segments of workload

- **Uncertainty Off**
  - Disable Prophet’s Uncertainty Intervals
Tune Hyperparameters

Launch Prototype
- 1 VM (single queue)
- 200 forecasts
- 8-hour run-time

Stage Inputs
- 7+ hours run-time for training history
- SELECT
- Remove cursor-based outlier replacement
- Remove self-joins, replace with windowing function
- Perform history aggregation once, save for later use
- 1 VM (single queue)
- 200 forecasts
- 15-minute run-time

Hyperparameters
- Thesis: single set of hyperparameters exists with performance better than default
- Use Grid Search + Bayesian Optimization
- Parallelize Grid Search over 6 VMs
- 10-hours elapsed vs. 60-hours
- MAPE improved from 6.5% to 6.23%

Expand Workload
- 1 VM (single queue)
- 4,000 forecasts
- 8+ hour run-time
- Staged inputs db footprint > 150 GB
- Staged inputs db footprint ~ 5 GB

CCI
- Stable results encouraged business
- Added SKUs & Supply Chain Centers
- Data Volume & Compute needed expanded 20x
- Replaced Heap + Nonclustered Index with Clustered Columnstore Index (CCI)

Event-based
- Time-based launch leaves slack in system
- Event-based trigger runs instantly after all dependents complete

Parallel, Parallel
- Use all available VMs to run mutually exclusive segments of workload
- 6 VM (3 queues)
- 4,000 forecasts
- 1.3 hours run-time

Uncertainty Off
- Disable Prophet’s Uncertainty Intervals
- 6 VM (3 queues)
- 4,000 forecasts
- 27-minute run-time
Tune Hyperparameters: Grid Search

Parameterize the forecast function

```r
m <- prophet(holidays = holidays,
growth = "linear",
interval.width = 0.25,
change-point.prior.scale = parameters$change-point_prior_scale,
n.change-points = parameters$n.change-points,
daily.seasonality = F,
weekly.seasonality = F,
yearly.seasonality = F
}

m <- add_seasonality(m, "yearly", period=365.25,
prior.scale=parameters$yearly.seasonality_prior_scale, fourier.order=parameters$yearly_fourier_order)
```

Pass the new function some random values

```r
rand_search_grid = data.frame(
  change-point.prior.scale = sort(runif(20, 0.01, 0.1)),
n.change-points = sample(8:25, 20, replace = F),
yearly.prior.scale = c(sort(sample(c(runif(5, 0.01, 0.05), runif(5, 1, 10)), 10, replace = F)),
  sort(sample(c(runif(5, 0.01, 0.05), runif(5, 1, 10)), 10, replace = F)),
yearly.fourier.order = sample(6:80, 20, replace = F),
Value = rep(0, 20)
)}
Tune Hyperparameters: Parallelize Grid Search

Run 6x Instances of Grid Search concurrently
Tune Hyperparameters: Bayesian Optimization

Wrap new forecast function with Bayesian Optimization

```r
library(rBayesianOptimization)

# Optimize prophet with Bayesian Optimization
changepoint_bounds = range(rand_search_grid$changepoint_prior_scale)
n_changepoint_bounds = as.integer(range(rand_search_grid$n_changepoints))
year_bounds = range(rand_search_grid$yearly_prior_scale)
year_fourier_bounds = as.integer(range(rand_search_grid$yearly_fourier_order))

bayesian_search_bounds = list(changepoint_prior_scale = changepoint_bounds,
                                n_changepoints = as.integer(n_changepoint_bounds),
                                yearly_prior_scale = year_bounds,
                                yearly_fourier_order = as.integer(year_fourier_bounds))

ba_search = BayesianOptimization(prophet_fit_bayes,
                                 bounds = bayesian_search_bounds,
                                 init_grid_dt = rand_search_grid,
                                 init_points = 5,
                                 n_iter = 12,
                                 acq = 'ucb',
                                 kappa = 1,
                                 eps = 0,
                                 verbose = TRUE)
```

Seed Bayesian Optimization with Grid Search results

Search feature-space for global optimal value of the model evaluation metric (MAPE)

Since rBayesianOptimization seeks to maximize the target (MAPE) pass it MAPE x (-1)
Tune Hyperparameters: Results

Parameter tuning generated MAPE improvement from 6.5% to 6.23% with negligible change in standard deviation of errors.

Grid search with serial execution would have elapsed 60+ hours. Parallel execution, across RapidMiner queues, on all nodes, took little more than 10-hours.
Event-based Trigger

Launch Prototype
- 1 VM (single queue)
- 200 forecasts
- 8-hour run-time

Stage Inputs
- 7+ hours run-time for training history
- SELECT
- Remove cursor-based outlier replacement
- Remove self-joins, replace with windowing function
- Perform history aggregation once, save for later use

Hyperparameters
- Thesis: single set of hyperparameters exists with performance better than default
- Use Grid Search + Bayesian Optimization
- Parallelize Grid Search over 6 VMs
- 10-hours elapsed vs. 60-hours
- MAPE improved from 6.5% to 6.23%

Expand Workload
- 1 VM (single queue)
- 4,000 forecasts
- 8+ hour run-time
- Staged Inputs db footprint > 150 GB

CCI
- Stable results encouraged business
- Added SKUs & Supply Chain Centers
- Data Volume & Compute needed expanded 20x
- Replaced Heap + Nonclustered Index with Clustered Columnstore Index (CCI)

Event-based Trigger
- Time-based launch leaves slack in system
- Event-based trigger runs instantly after all dependents complete

Parallel, Parallel-execution
- Use all available VMs to run mutually exclusive segments of workload
- 6 VM (3 queues)
- 4,000 forecasts
- 1.3 hours run-time

Uncertainty Off
- Disable Prophet's Uncertainty Intervals
- 6 VM (3 queues)
- 4,000 forecasts
- 27-minute run-time
Event-based Trigger

Time-boxed process start can lead to process launch before all dependents are ready, or to **lost opportunity to begin ahead of schedule**

Read-only replicas of EDW databases are “snapped” to the Data Science environment daily, at 4 AM, but exact timing varies

This process checks for “snap” completion, and only then allows the down-stream forecasting process to begin

The Event-based process allows our forecasting model to start “as soon as it can”
What’s Next?

Launch Prototype
- 1 VM (single queue)
- 200 forecasts
- 8-hour run-time
- 7 hours run-time for training history
- CREATE
- Remove cursor-based outlier treatment
- Remove and replace with windowing function
- Perform history aggregation once, save for later use
- 1 VM (single queue)
- 200 forecasts
- 15-minute run-time
- MAPE improved from 6.5% to 6.23%

Tune Hyperparameters
- Thesis: single set of hyperparameters exists with performance better than default
- Use Grid Search + Bayesian Optimization
- 10-hour epoch vs. 60-hours

Expand Pilot 20x
- Stable results encouraged business
- Added SKUs & Supply Chain Centers
- Data Volume & Compute needed expanded 20x

Event-based Trigger
- Replaced Heap + Nonclustered Index with Clustered Columnstore Index (CCI)
- Time-based launch leaves slack in system

Parallel, Parallel
- Use all available VMs to run mutually exclusive segments of workload
- 6 VM (3 queues)
- 4,000 forecasts
- 1.3 hours run-time

Uncertainty Off
- Disable Prophet’s Uncertainty Intervals
- 6 VM (3 queues)
- 4,000 forecasts
- 27-minute run-time

CCI + Partitioning
- Today
- Disable Uncertainty Sampling

Parallel execution
- Today
RapidMiner Enabled Success

- Low-code interface
  - Speedy development
  - Speedy testing
- Integration of scripting languages
- Orchestration across systems
- Server-side hosting
- Parallel execution
- Event-based process

**Goal**
Highly accurate, highly scalable demand forecasts

**Problem**
Shared resources limited; ecosystem of competing projects expanding rapidly

**Solution**
Creating thinking to keep footprint small