15.095. Homework 4

Kim-Anh-Nhi Nguyen

November 28, 2018

1 Optimization problem formulation

Let us have the following assumptions:

- To determine the optimal levels of stocks for August 15th 2017, we rely only on the data from that specific day
- We know which items are going to be displayed for the targeted day (`ondisplay` column in the sales table)
- Moreover, only items that are displayed can be sold (this has been checked on the data)
- We know expected demand for the day, being the volume of sales for each item that we expected, based on some predictions we have (`unit_sales` column in the sales table)
- We know which items are going to be on promotion for the targeted day (`onpromotion` column in the sales table)
- We know which items are perishable (`perishable` column in the items table)
- We know the cost and the selling price of each item (`cost` and `price` columns in the items table)

So we have the following variables (again, for August 15, 2017):

For fixed variables:

- $display_i = 1$ if item i is on display, 0 otherwise
- $promo_i = 1$ if item i is on promotion, 0 otherwise
- $d_i =$ demand (in units) for item i
- $per_i = 1$ if item i is perishable, 0 otherwise
- $c_i =$ unit cost for item i
- $p_i =$ unit price for item i

For decision variables:

- $z_i =$ number of units of item i that we decide to stock

Finally, the optimization problem is:

$$\max \; revenues = r(d,z) = \sum_{i=1}^{100} [(p_i(1-0.2promo_i) - c_i)\min(z_i, d_i) - (z_i - min(z_i, d_i))c_i per_i]$$

$$s.t.$$

$$\sum_{i=1}^{100} z_i \leq Q$$

$$for\ all\ i, z_i \leq \frac{1}{10} Q\ display_i$$

$$for\ all\ i, z_i \geq 0$$

i.e.

$$\max\ r(d,z) = \sum_{i=1}^{100} [(p_i(1 - 0.2promo_i) - c_i)s_i - (z_i - s_i)c_i per_i]$$

$$s.t.$$

$$\sum_{i=1}^{100} z_i \leq Q$$

$$for\ all\ i, z_i \leq \frac{1}{10} Q\ display_i$$

$$for\ all\ i, z_i \geq s_i$$

$$for\ all\ i, d_i \geq s_i$$

$$for\ all\ i, z_i \geq 0$$

$$for\ all\ i, s_i \geq 0$$

2 Three approaches to solve the prescriptive optimization problem

(a) Make pure predictions: standard supervised learning and point-prediction-driven decision

Given a machine learning model that was trained on past data, and some new data x_{n+1} (i.e. the observation from August 15th 2017), we can predict for each item i, the expected sales volume for item i: $\hat{y}_{i,n+1}$. And we could simply use this value as the expected amount of sales for each item i.

Then, we can plug the values $\hat{y}_{i,n+1}$ into an optimization solver to solve the problem formulated in question 1., with $d_i = \hat{y}_{i,n+1}$

The pro is that this method is straightforward and accounts for auxiliary data. But the con is that it doesn't account for uncertainty.

(b) SAA: Sample Average Approximation

We use historical values y_i and average the profit over them. Then choose the decision z which maximizes this average.

The pro is that it doesn't use any predictions and is simple to calculate. But the con is that z would be the same for each product and doesn't account for uncertainty.

(c) Prescriptive Method

Given a machine learning model, e.g. a **decision tree** that was trained on past data, and some new data x_{n+1} (i.e. the observation from August 15th 2017), we can predict for each item i, the expected sales volume for item i: $\hat{y}_{i,n+1}$.

Then, a second step towards the decisions would be to use the predictions to maximize the following weighted sum taking the the cost function from question 1. into account:

$$\max_{z \in Z} \sum_{j=1}^{N} w_{N,j}(x) revenue(y_j, z)$$

i.e.

$$\max_{z \in Z} \sum_{j=1}^{N} w_{N,j}(x) r(y_j, z)$$

Where, for a decision tree: $w_{N,j}(x) = \frac{1}{|R(x)|}$ = number of data points belonging to the leaf, if $x^j \in R(X)$, 0 otherwise

The pro is that this method is fast. The con is that we split the prescription into two different parts.

3 Machine Learning Model: Decision Tree

Please, see the attached notebook `hw4-ML-Nhi-Q3.ipynb` for the implementation.

(a) Data Preprocessing and feature engineering

Even though we want to predict the variable `sales` which is in the items.csv file, we want to use the columns from items.csv and information.csv as well, so we know everything about every item and information about each day (like oilPrice or isHoliday).

Therefore, a first step is to left merge the 3 tables, sales.csv being the one on the far left.

Secondly, the Decision Tree model can only take numbers (integers and floats) as columns types, so we need to transform some columns as follows:

- Label encoding: for the columns `item_nbr`, `family`, and `class`, we encode the categories into integer values, each value mapping to a unique category for the column.
- Date: create 3 additional columns `Year`, `Month` and `Day` as **integers**, corresponding to the year, month and day respectively, of the observation's date, then, delete the column `date` because the model

Finally, we split the data into a training and a testing set. The training set has all the observations without the observation from August 15th 2017, and the testing set has the 100 observations from August 15th 2017. We also split each set into X and y, y being the target variable `unit_sales`.

(b) Train the Decision Tree

We trained a Decision Tree model, using `DecisionTreeRegressor` from Python `sklearn` machine learning package.

We use the function `GridSearchCV` to do parameter tuning on the parameters `max_depth`, `min_samples_split` and `min_samples_leaf`.

The model with the best parameters finally has the following parameters:

- `max_depth=20`
- `min_samples_leaf=10`
- `min_samples_split=100`
- `max_depth=20`

(c) Model's performance

To predict the expected demand for each item for August 15th 2017, we use the model that was just trained on our testing set `X_test` to make the predictions. As the predictions need to be integers (number of units stocked per item), we round the predictions to the closest integer.

To measure the performance of the model, we use a **baseline model** which predicts the sales from the previous day, i.e., it predicts that the sales for August 15th 2017 are exactly the same as for August 14th 2017.

Then, we perform the Mean Absolute Error (MAE) on both models (the decision tree and the baseline model) to compare the performances.

Here are the results:

- MAE of the baseline model: 1.94
- MAE of the decision tree: 1.49

Therefore, the decision tree makes good predictions as it performs better than the baseline model: it accounts for variability in days.

4 Prescriptions

The best prescription method from the 3 listed is the 3rd (c): we maximize the weighted sum of profits.

Please see the code from the notebook `hw4-ML-Nhi-Q3.ipynb` and the notebook `hw4-ML-Nhi-Q4.ipynb` to see how the observations are extracted from the leaves and how the optimization model is implemented.

Our oracle profit is: 615.998 dollars.
Here is a table summarizing the results for different values of the capacity Q:

Q	1	10	50	100	500	1000	10000
Baseline profit	3.44	34.35	129.84	226.56	412.10	412.10	412.10
Baseline optimality gap	612.56	581.65	486.15	389.44	203.90	203.90	203.90
Model's profit	3.01	30.11	163.68	280.68	539.77	575.03	575.03
Model's optimality gap	612.99	585.89	452.32	335.32	76.23	40.97	40.97
Gain in profit (%)	-12.3	-12.3	26.1	23.9	31.0	39.5	39.5

Our comments and conclusions are:

- For small values of Q (1 and 10), the baseline model performs better than our model. But quickly, as Q increases, our model outperforms the baseline model, reaching a maximum percentage gain in profit of almost 40%, which is quite good.
- As for the optimality gap, it obviously decreases as Q increases, because we have more capacity to stock items.
- But, the minumum optimility gap is around 204 dollars for the baseline model whereas it is around 41 dollars for our prescriptive model (we are less than 7% below the oracle profit). Therefore, we are really close from the oracle profit
- So we can conclude that implementing this prescriptive method to predict inventory levels can improve profits by almost 40% and get as close as possible to perfect levels.