Contents

Problem statement of facility location problem

Consider a firm that would like to place a single facility per time period over a planning horizon. The firm has available a set of candidate locations, and each location has a revenue per unit of captured demand. The firm moves the facility over time to capture the demand of a set of customers. Note that customers are willing to patronize a subset of candidate locations based on personal preferences. Each customer has an initial demand value as well as lower and upper bounds for cumulative demand over time. The cumulative demand evolves at each time period demand through replenishment and absorption. More specifically, at the beginning of each time period, the cumulative demand of each customer is replenished by an absolute rate (i.e., a fixed value) and by a relative rate (i.e., a percentage). Throughout each time period, the cumulative demand of customers captured by the installed facility is absorbed by an absolute rate and by a relative rate. The demand absorbed by the installed facility is referred to as capturable demand (i.e., the demand that the firm manages to convert to turn into revenue). At the end of each time period, the cumulative demand is consolidated according to the lower bound for each customer. The firm would like to choose where to place the single facility over time to capture customers accordingly while maximizing the total revenue.

Solving this problem using Machine Learning

Method 1

The Assumption here is we have past data where the firm visited (locations) in different sequences and have the respective demand captured from each customer in a location in different time periods. Each location captures demand from several customers.

Training of ML model

We have the input X for ML as the following:

- The previous demand captured from every customer in past time periods
- The sequence of locations we have visited till the current period
- The location we are visiting in the current period

The prediction label Y of ML:

- The current period demand captured from a particular customer at the current location

Data Availability

The ML model was trained on different Data availability conditions.
Training data availability:

- 150 episodes (large, but practically difficult)
- 25 episodes (reasonable size)
- 12 episodes (scarce)

Data Split for training and validating:

- 150 episodes of data is split into 135 episodes for training and 15 episodes for validating the model
- for 25 episodes (22 training, 3 validating)
- for 12 episodes (10 training, 2 validating)

For testing of ML model, we have 50 episodes of untouched data.

Machine Learning models considered for the above training

- Random Forest, Logistic Regression, Support Vector Regressor
- LSTM
- GRU
- bi-LSTM & bi-GRU

We initially tried basic ML models like Random Forest, Logistic regression, and SVM but the predictions from these models were not that good. We know the data is sequential and past information affects the future so the next idea is to move to sequential models that can retain past information, So we tried LSTM (Long short-term Memory) and GRU (Gated Recurrent Unit).
Both these models performed well compared to basic Machine Learning Models and prediction errors were similar.

ML model	Training episodes	Test episodes	MSE	Expected range(MSE)
RF	150	50	0.25	<0.3
RF	25	50	0.71	<0.8
LSTM	150	50	0.0.036	<0.05
LSTM	25	50	0.042	<0.05
LSTM	12	50	0.018	<0.25
GRU	25	50	0.0328	<0.05
GRU	12	50	0.16	<0.25
biLSTM	12	50	0.1034	<0.15
biGRU	12	50	0.102	<0.15

The LSTM and GRU models performed well (MSE < 0.05) with data availability of 150 and 25 episodes but couldn't perform similarly with 12 episodes of data.
So tried training the model using bi-LSTM and bi-GRU (bi - BiDirectional) and it did better to give an MSE loss of around 0.10 from around a loss of 0.20 with normal LSTM and GRU.
We prefer bi-GRU for all our upcoming models because both LSTM and GRU gave similar prediction errors and GRU architecture is computationally fast compared to LSTM .

Difference Between GRU and bi-GRU

- Bi-GRU extends the GRU architecture by processing input sequences in both forward and backward directions simultaneously
- It consists of two separate GRU layers, one processing the input sequence in the forward direction and the other in the backward direction.
- By doing so, it captures information from past and future contexts, providing a richer representation of the input sequence

Now, With the past information available, we can predict the customer's demand at a particular location in a particular period with great accuracy using the ML model. But we aim to devise a sequence of locations for the firm to visit to maximize the total capturable demand.
One Solution would be Greedy, wherein each period you predict the demand of customers in each location and visit the location whose sum of demand is maximum.

Dynamic Programming / Pseudo Greedy solution

T = total time period in an episode
l(t) = location to visit at time period t
C(L) = Customers visited at a particular location (fixed)
L is set of all locations

l*(t) is the optimal location to visit at time t

Pseudo Code:

captured −− > [[ ]] (empty 2d array) # (will be updating it with locations visited and demand captured from customers)
solution −− > [ ] (empty list) #w (ill update the sequence of locations visited)
For t in T:
. l*(t) < −− arg max (l belongs to L){ P(captured, l, t) +simulate(t =t+1,l(t)=l, captured)}*
. *solution < −− solution.append(I*(t))*
. *update captured based on I**
Return solution

Here,
P(captured, l, t) is the prediction of total demand captured from the location l at time t.
simulate(t=t+1, l(t)=l, captured) is a function that runs a greedy algorithm from time period t= t+1 to T with the assumption that we visited location l at time t. This function returns the total sum of demand captured from time t+1 to T using the greedy method.

We have taken a data with total of 20 periods in an episode with 10 customers and 10 locations. Using the DP-Greedy algorithm, we generated the sequence of locations for the firm to visit.
The DP-Greedy model's solution sequence gave a total captured demand in all time periods of around 670 - 720. This range is due to using different ML models (12 episodes, 25 episodes, 150 episodes) for the prediction.
There wasn't any significant performance difference between models with higher episodes compared to models with lower episodes while using different ML models.
The optimal set of location sequence had a total demand captured value of 777. While our solution lies in the range of 670-720.
Our Soultion is less than optimal solution by 7%-13%
The solution with the mathematical optimization method captured a total demand of 582, which is 25% lower than the optimal solution.

Method 2

We wanted to use complete Dynamic Programming so we made an assumption that we have only the information whether the customer was visited in previous time period or not.

v(t-1)=0 if the customer was not visited in t-1 time instance
v(t-1) =1 if custom was visited in t-1 time instance

ML Training

The input X would have:

- v(t-1) of customer
- time period t
- l(t) location visited at time period t (if customer is visited or not)

The Output Y is:

- Demand captured from a customer at time t

The bi-GRU model of 12 episodes data with only v(t-1) information available gave us MSE of 0.68 which is pretty high compared to our previous model which has MSE less than 0.10

Now to find the sequence of locations we deploy Dynamic Programming

Pseudo Code

solution = [ ]
For t in T:
. *I*(t) = arg max (l belongs to L) { p(v(t-1), l, t) + simulate (v(t)=l , t+1))}*
. *solution = solution.append(I*(t))*
Return solution

simulate (v(t)=l, t+1):
. *Return Max(for all l') { P(v(t), l', t+1)+simulate(v(t+1)= l',t+2)}*

Here,
P(v(t-1), l, t) predicts the total demand captured from location l at time t with location v(t-1) visited at time instance t-1.
simulate (v(t) =l,t+1) this is a Dynamic programic algorithm to go through states v(t-1),t for time t= t+1 to T

This algorithm did not perform well as the ML model predictions were not good.
This solution gave total captured demand less than 520 which way less than optimal solution value of 777 and also less than mathematical optimization solution.

Method 3

Since the Dynamic programing methods didn't work well we wanted to increase the state space of DP model with also providing the demand captured in v(t-1)
The demand captured w(t-1) in time t-1 is classified into 5 groups,
0 if no demand captured,
1 if demand captured is less than $\frac{1}{4}$ of max demand captured from that customer
2 if demand captured is between $\frac{1}{4}$ and $\frac{1}{2}$ of max demand captured from that customer
3 if demand captured is between $\frac{1}{2}$ and $\frac{3}{4}$ of max demand captured from that customer
4 if demand captured is greater than $\frac{3}{4}$ of max demand captured from that customer

ML Training

The input X would have:

- w(t-1) of custome
- time period t
- l(t) location visited at time period t (if customer is visited or not)

The Output Y is:

- Demand captured from a customer at time t

The bi-GRU model of 12 episodes data with w(t-1) information available gave us MSE of 0.42 which is better compared to model with only v(t-1) inofrmation.

Now to find the sequence of locations we deploy Dynamic Programming

Pseudo Code

solution = [ ]
For t in T:
. *I*(t) = arg max (l belongs to L) { p(w(t-1), l, t) + simulate (w(t) , t+1))}*
. *solution = solution.append(I*(t))*
Return solution

simulate (w(t), t+1):
. *Return max(for all l') { P(w(t), l', t+1) + simulate (w(t+1) ,t+2) }*

Here,
P(w(t-1), l, t) predicts the total demand captured from location l at time t with w(t-1) percent

demand captured at time instance t-1.
simulate (w(t)l,t+1) this is a Dynamic programic algorithm to go through states w(t-1),t for time t= t+1 to T
w(t) is classified as 0,1,2,3,4 based on P(w(t-1),l,t)

Even with this model there wasn't any significant improvement in solution prediction. The solution gave a total demand captured of 547 which is lower than mathematical optimization solution.

Method 4

Here we assume that we only have the demand captured from location in previous time periods and we don't have individual demands captured from customers. (No customers info). This method is very similar to method 1 but we don't have individual customer information.

Initially, given Bitcoin's generally favorable price trends, our idea was to develop strategies capable of capturing market trend movements. Consequently, we formulated models utilizing indicators such as the RSI, MACD, and another model based on the Ichimoku framework.

ML training

Input X for ML model:

- The previous demand captured from every location in past time periods
- The sequence of locations we have visited till the current period
- The location we are visiting in the current period

Output Y of ML model:

- the demand captured from current location at the current period

Pseudo Code

To find the Sequence of locations:
T = total time period in an episode
l(t) = location to visit at time period t
L is set of all locations
l*(t) is the optimal location to visit at time t

Solution −− > [ ]
Captured −− > [ ] # (will be updating it with locations visited and demand captured from

location)
For t in T:
. *l*(t) <-- arg max (l in L) { P(captured, l, t) + simulate(t+1,captured, l(t)=l) }*
. *solution <-- solution.append(I*(t))*
. *update captured based on I**
Return solution

Here,
P(captured, l, t) is the prediction of total demand captured from the location l at time t. This ML model directly predicts total demand from location but in method 1 it is the summation of demand predicted for all customers in a location
simulate(t=t+1,captured , l(t)=l) is a function that runs a greedy algorithm from time period t= t+1 to T with the assumption that we visited location l at time t. This function returns the total sum of demand captured from time t+1 to T using the greedy method.

This algorithm didnt perform as good as method 1 because we didnt have customer information available so the individual customer characteristic wasn't learned by the model. This method gave a solution with total demand captured value equal to 535, which is also lower compared to the mathematical optimization solution.

Overall with all these methods the methods 1 with informaataion of all previous demand captured of customer known the model performed best. The ML methods are not performing well when there isn't sufficient information in data to understand the characteristic's of the customer.

There can still many models be build with different assumptions to predict the optimal sequence of location.