In-flight Wifi Pricing Optimization

California State University, Fullerton
In collaboration with Panasonic and Black Swan Data
Romain Drai, Tin Huynh, James Luo, Shan Jiang, Jessica Shin, Foon Wong, Tingting Xu
May 1, 2019

Abstract

Introduction

Dynamic pricing is a pricing policy in which a firm adjusts the price of their product as a function of its perceived demand in order to maximize profit. A common challenge when applying machine learning techniques for pricing optimization is the lack of past observations. The specific problem of inflight WiFi pricing optimization is no exception. While Panasonic collects millions of transaction per year from its wide network of partner airlines, the actual number pricing policies for learning consumer behavior are much more limited if we consider regional, product and hardware differences. Fortunately, if we strategically generalize some of the variables, we can create opportunities for learning.

From literature research, we choose to adopt a two-staged framework to tackle this problem. First, a demand, or price-response model is built to understand the effect of the different factors. Then, a dynamic pricing model is built to find the optimal pricing. The appeal aspect of this framework is that the problem is broken down into two independent stages. For this project, we can build upon any prior work done on the demand model; conversely, improvements to the demand model can be made without changing the pricing solver. Finally, it provides an evaluation scheme for validation and future improvements.

Data Preprocessing

The raw data describes pricing and usage per session. Given a per-megabyte cost of data, we are able to calculate the profit-per-session of each session of data usage. Importantly, cost is linked to data usage, and needs to be modeled.

Show model for cost noted: will add graphs

From the raw data, we extract relevant features of each session, including whether the session was data capped or time capped. We impute values for missing data based on the data type. For total passenger count of each flight, of which approximately 3% are missing, we impute with the average value for the given airline and aircraft type. For all boolean features describing amenities such as in-flight entertainment, TV available, and assorted luxury items, of which less than 1% are missing, we impute by defaulting to false. For night flights, which is a categorical feature for which roughly 10% are missing, we also impute by defaulting to false. For international flight, of which approximately 3% are of unknown status, we impute by examining the origin and destination countries. Finally, close to 0.1% of flights indicate more passengers than seats available, which is nonsensical. In these cases we replace the value with the average number of passengers for that aircraft type.

At this point we have a cleaned dataset of flights with identifiable features and imputed values. We then split this dataset into three distinct sets of data based on whether the flight was data capped, time capped, or otherwise.

Show distribution of three datasets here Noted

Using these three sets of data, we can then extrapolate the profit-per-person per session for different types of cap and usage, and thereby define a metric for which we can measure profitability

Demand model

We have found a wide variety of demand model specifications. Parametric models from the generalized linear model family are well studied and their success depends on whether their assumptions are met. In addition, their gradient are relatively easy to obtain, allowing optimising problems to be solved. On the other hand, powerful machine learning techniques such as random forest or neural networks can also be employed, but additional optimization problem can become challenging.

The goal for the demand model is to capture the most important factors that influence demand.

$$Q_t = \psi_t(P_1, P_2, ..., P_N),$$

```
where $Q$ is the total sales and $\psi(.)$ is the demand function at time $t$. $\psi(.)$ can take on a
```

Previously, Panasonic data science team have built a successful demand model based on random forest. They have found that the take rate/sales per flight for their products depend greatly on covariates such as the number of passengers, flight route, red-eye, etc. To estimate demand difference, for example between regular vs red-eye flights, we can check model prediction by holding other factors constant.

Pricing optimization model

While a demand model can provide prediction on demand and easily solve for pricing that maximizes revenue, total profit needs to be separately calculated. The optimization problem can be formulated as:

```
$$
\underset{P_1,\dots,P_N}{\max}   \Pi = \sum_{i=1}^N [\psi_t (P_1, P_2, \dots, P_N)(P_t - C_t)]
$$
```

```
where $\Pi$ is the total profit across a planning horizon, as a function of a et of pricing $P_1, \dots
```

For cost estimation, we will assume a linear relationship between customer data usage and true cost. This functional relationship base on actual business situation and can be included as a functional parameter. In addition, capacity and pricing limiting constraints can be imposed.

In our research, we came across multiple techniques such as kernel estimation, greedy iteratively reweighted least squares (GILS) and control variance pricing (CVP). However, for this price optimization step, we propose to use evolutionary algorithms (EA). EAs are a family of population based optimization techniques. They use the concept of natural selection and random variation to evolve better solutions to the problem. EAs are advantageous for two reasons. First, they do not depend on the functional form of the demand model and can thus be used to optimize a wide variety of complex models. Second, they have the ability of simultaneously searching over a wide range of possible solutions, and are thus more likely to converge toward a global maximum.

Outline of EA

An EA begins by randomly generating a population of pricing policies. Our pricing policy would be a combination of prices and the relevant flight factors. Each policy would be evaluated on its profitability using the demand model. Then, subsets of good policies are selected and is used to generate new policies. This process is repeated until a termination criterion is met. Popular examples of EAs include the genetic algorithm (GA) which relies on crossover and mutation operators. A crossover operator randomly selects two policies from the subset of best policies and combines them in order to create an even better solution. Similarly, the mutation operator introduces additional variation by slightly modifying a policy in order to search unexplored parts of the sample space. Another type of EA are Estimation of Distribution Algorithms. EDAs take a more probabilistic approach. They build a probabilistic model of solutions and sample the model to generate the new child population. The two EDAs we are considering using are: Population Based Incremental Learning (PBLT) and Distribution Estimation using Markov random fields (DEUMd) The EDAs that we are considering can be classified as univariate EDA where the variables in the problem space are considered independent from each other. For the problem of internet pricing optimization, it is suitable when we are only considering pricing change. We have also found extensions of EDA that can take into consideration of multivariate interactions, which can be useful if we are interested in optimizing both pricing and data cap limit.

Outline of evolutionary algorithm workflow

An EA begins by randomly generating a population of pricing policies. Our pricing policy would be a combination of prices and the relevant flight factors. Each policy would be evaluated on its profitability using the demand model. Then, subsets of good policies are selected and is used to generate new policies. This process is repeated until a termination criterion is met.

Popular examples of EAs include the genetic algorithm (GA) which relies on crossover and mutation operators. A crossover operator randomly selects two policies from the subset of best policies and combines them in order to create an even better solution. Similarly, the mutation operator introduces additional variation by slightly modifying a policy in order to search unexplored parts of the sample space.

Genetic algorithm

1. Generate a population P consisting of M solutions
2. Build a breeding pool by selecting promising solutions from P
3. Perform crossover on the breeding pool to generate a population of new solutions
4. Perform mutation on the new solutions
5. Replace P by the new solution and go to step 2. Repeat this process until the termination criteria is met

Estimation of distribution algorithms (EDA)

Another type of EA are Estimation of Distribution Algorithms. EDAs take a more probabilistic approach. They build a probabilistic model of solutions and sample the model to generate the new child population. The two EDAs we are considering using are:

```
* Population Based Incremental Learning (PBLT)
```

- Distribution Estimation using Markov random fields (DEUMd)

The steps of the PBLT algorithm are shown below:

```
1. Initialize a probability vector $p = p_1, ..., p_n$ with each $p_i = 1/2$, where $p_i$ represent the p
```

2. Generate a population P consisting of M solutions by sampling probabilities in p.
3. Select set D from P consisting of N promising solutions where $N < M$
    4. Estimate the marginal probabilities of $x_i = 1$ as

    $$p(x_i = 1) = \frac{\sum_{x \in D} x_i}{N}$$

    5. Update each p_i in p using $p_i = p_i + \lambda(p(x_i = 1) - p_i)$ where $\lambda \in [0, 1]$ is the learning rate parameter
4. Go to step 2 and repeat until termination criteria are met

The EDAs that we are considering can be classified as univariate EDA where the variables in the problem space are considered independent from each other. For the problem of internet pricing optimization, it is suitable when we are only considering pricing change. We have also found extensions of EDA that can take into consideration of multivariate interactions, which can be useful if we are interested in optimizing both pricing and data cap limit.

Evaluation

Although optimal pricing can be solved using EA based techniques, the accuracy of the demand model is equally critical and they must be evaluated together.

An proposed evaluation scheme for this problem:

```
1. Data simulation
```

i. Split existing data using decision trees into different scenarios
ii. Under each scenario, generate 60 pricing representing 60 weeks of data
iii. Estimate the corresponding demand with the demand model

2. Optimization and demand model evaluation

i. Generate true optimal pricing by using the data generating model as input
ii. Re-fit demand models using simulated data. Solve for optimal pricing under the simulated demand models

3. Evaluate performance of the overall optimization by comparing the difference in prices (RMSE) obtained from 2-i and 2-ii.

##Gradient Boosting/lightGBM

http://jmarkhou.com/lgbqr/ This is a pretty good reference. Try to read the references that guy uses

llightGBM is a gradient boosting framework that uses tree-based learning algorithm. It was designed for distributed training and it splits the tree leaf wise with the best fit. The framework is fast and lower memory usages. The gradient boosting has two primary method: bagging and boosting. Bagging involves the training of independent models and combines their prediction. When we want to reduce the variance of decision tree, we used bagging and random forest is one of the example. If single model has low performance, bagging will not get a better bias, but boosting could generate a combined model with lower error. Both are good at reducing variance and provide higher stability, however, if the single model is overfitting, bagging would be the best option.

When we consider decision trees, we start with an F_0(initial fit) The constant value that minimize the loss function L is:

```
$$
F_0(x)=argmin_{\rho}\sum_{i=1}^nL(y_i,\rho)
```

$$ In the case of optimizing the MSE, we take the mean of the target values $F_0(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$

```
Calculate pseudo residual with initial guess of $F_0$
```

```
$$
r_{i1}=-\dfrac{\partial L(y_i,F_{0}(x_i))}{\partial F_{0}(x_i)}
```

$$

```
Now, we can fit the decision tree $h_1(x)$ to the residuals .
```

In order to minimize the loss for each leaf, we apply gradient descent by stepping in the direction of average gradient for the leaf nodes from the decision tree $h_1(x)$ yielding a new boosted fit of the data: $F_1(x) = F_0(x) + \lambda_1 \rho_1 h_1(x)$ where λ_1 is learning rate

Gradient Boosting Algorithm Let M be a number of boosting rounds and L be a differential loss function L:

$$F_0(x) = arg_r min \sum_{i=1}^{n} L(y_i, \gamma)$$

```
For m=1 to M
```

Calculate the pseudo residuals

$$\overset{\sim}{y_i} = -\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}$$

Fit decision tree $h_m(x)$ to $\overset{\sim}{y_i}$ Compute the step multiplier ρ_m for each leaf of $h_m(x)$ Let $F_m(x) = F_{m-1}(x) + \lambda_m \rho_m h_m(x)$ where λ_m is the learning rate for iteration m.

References

- Shakya, Kern, Owusu, and Chin. "Neural Network Demand Models and Evolutionary Optimisers for Dynamic Pricing." Knowledge-Based Systems 29 (2012): 44-53. Web.
- Bauer, and Jannach. "Optimal Pricing in E-commerce Based on Sparse and Noisy Data." Decision Support Systems 106 (2018): 53-63. Web.
- Owusu, G., Voudouris, Kern, Garyfalos, Anim-Ansah, and Virginas. "On Optimising Resource Planning in BT Plc with FOS." 2006 International Conference on Service Systems and Service Management 1 (2006): 541-46. Web.
- Shakya, Chin, and Owusu. "An AI-based System for Pricing Diverse Products and Services." Knowledge-Based Systems 23.4 (2010): 357-62. Web.
- Friedman "Greedy Function Approximation:A Gradient Boosting Machine." (2001): Web.