Machine Learning for Predicting Flight Ticket Prices

Supervisor: Boi Faltings, Igor Kulev
Student: Jun Lu
School of Computer and Communication Sciences, EPFL
1005, Lausanne, Switzerland
Email: jun.lu@epfl.ch

***Abstract*—Airline companies have the freedom to change the flight ticket prices at any moment. Travellers can save money if they choose to buy a ticket when its price is the lowest. The problem is how to determine when is the best time to buy flight ticket for the desired destination and period. Airline companies use many different variables to determine the flight ticket prices: indicator whether the travel is during the holidays, the number of free seats in the plane etc. Some of the variables are observed, but some of them are hidden. The goal of this project is to use machine learning techniques to model the behavior of flight ticket prices over the time. In order to build and evaluate the model, you will use data that contains historical flight ticket prices for particular routes.**

I. Introduction

Introduction

March 01, 2016

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Data Collection

III. Regression

A. Feature Extraction

The features extracted for training and testing are aggregated variables computed from the list of quotes observed on individual query days. For each query day, there are possibly eight airlines quoting flights for a specific origin-destination and departure date combination. For each query data, 5 features are computed, the *flight number*(encoded by dummy variables), the *minimum price so far*, the *maximum price so far*, the *query-to-departure* (number of days between the first query date(09.11.2015 in our case) and departure date), the *days-to-departure* (number of days between the query and departure date), and *current price*.

B. Data Description and Interpretation

Our data set consist of one set of input features, each one of them split into training dataset and testing dataset.

The training dataset consist of $\mathbf{N_{tr}}$=6000 data samples of one output variable $\mathbf{y}$ and input variable $\mathbf{X}$. The input variable $\mathbf{X}$ depends on the feature provided, as detailed shown above. The output variable is present is the minimum price for the specific origin-destination and departure date.

The testing dataset consist of $\mathbf{N_{te}}$=11453 , for which the output is unknown, and where we forged our predictions. For each output, i.e. the expected minimum price, if the current price is smaller than the output, then we predicted to buy, otherwise, we predicted to wait.

C. Regression Model Construction

1) Neural Networks: Regarding the neural network architecture, we explored several configurations in terms of number of nodes and number of hidden layers. We reached the conclusion that the neural network architecture that performed the best consisting of a single hidden-layer neural network of 6 neurons. Even though the increasing number of hidden layers enables to express the output as any function of the input, it also conveys the risk of the model overfit. Hence, a single hidden-layer model can reasonably well explain the input output relationship. Regarding the number of neurons of this layer, it is known that a low number of neurons in the hidden layers implies loss of important information, while a higher number of neurons can prevent convergence. We ran our models under 100 epochs, which conveys convergence while preventing to high computational cost.

Finally, as shown above, we defined three layers, namely the input layer, the hidden layer and the output layer. We set the output's activation function to be sigmoid function. So the output of the data is the probability of buying the ticket on that query date. The nonlinearity function used by other layer is $rectifier$, which is simply $max(0, x)$. It is the most popular choice of activation function these days. The neural net's weights are initialized from a uniform distribution with a cleverly chosen interval. That is, Lasagne figures out this interval for us, using "Glorot-style" initialization.

The update function will update the weights of our network after each batch. We'll use the $nesterov\_momentum$ gradient descent optimization method to do the job. There's a number of other methods that Lasagne implements, such as sgd, $adagrad$ and $rmsprop$. We chose $nesterov\_momentum$ because it has proven to work very well for a large number of problems. The core idea behind Nesterov Momentum is it looks at the point in the vicinity of where we are soon going to end up.

Because it is a regression problem. So we specified the mean squared error(MSE) as an objective function to minimize.

Then the data provided in X were split into a training and a validation set, using 20% of the samples for validation.

2) Decision Tree: The core idea of Decision Tree is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. It learns from data to approximate a decision with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

Decision Tree is simple to understand and to interpret and requires little data preparation. However, Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem. In our case, we used maximum depth of 3 to get the best performance. And we set the loss function to be MSE to minimize.

3) K Nearest Neighbors: Neighbors-based regression can be used in cases where the data labels are continuous rather than discrete variables. The label assigned to a query point is computed based the mean of the labels of its nearest neighbors.

The K-nearest neighbors regressor requires a setting for K, But what number works best? Additionally, we saw that there are many different distance functions we could have used: L1 norm, L2 norm. Also, the basic nearest neighbors regression uses uniform weights: that is, each point in the local neighborhood contributes uniformly to the classification of a query point. Under some circumstances, it can be advantageous to weight points such that nearby points contribute more to the regression than faraway points. In this case, we can assign weights proportional to the inverse of the distance from the query points.

Finally, we performed several combinations of this problem, getting the result that using K=6, distance metric of $L2\,norm$ and uniform weights will get the best performance.

4) Least Squares: fits a linear model with coefficients $w = (w_1, ..., w_p)$ to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation.

In our problem, although the Least Squares did not work very well, but it is useful for the Uniform Blending or Linear Blending Model described in the following sections.

5) Uniform Blending: If we have diverse hypotheses, i.e. many different algorithms of regression, then even simple uniform blending of these hypotheses can be better than any single hypothesis. Mathematically, it has the following formula:

$$G(x) = \frac{1}{T}\sum_{t=1}^{T} g_t(x) \tag{1}$$

where T algorithms $g_1, ..., g_T$ predict the same problem.

Theoretical Analysis of Uniform Blending: Let $f(x)$ be the true hypothesis, then:

$$\begin{aligned} avg((g_t(x) - f(x))^2) &= avg(g_t^2 - 2g_t f + f^2) \\ &= avg(g_t^2) - 2Gf + f^2 \\ &= avg(g_t^2) - G^2 + (G-f)^2 \\ &= avg(g_t^2 - 2g_t G + G^2) + (G-f)^2 \\ &= avg((g_t - G)^2) + (G-f)^2 \end{aligned} \tag{2}$$

which means:

$$\begin{aligned} avg(E_{out}(g_t)) &= avg(\mathbf{E}(g_t - G)^2) + E_{out}(G) \\ &\geq E_{out}(G) \end{aligned} \tag{3}$$

where $E_{out}(g)$ is the out-of-sample error.

From the analysis above, we can see that the uniform blending reduces variance for more stable performance.

IV. Classification

A. Feature Extraction

The input features are the same as the regression problem.

As for the output, we set the data of which the price is the the minimum price from the query date to the departure date to 1(namely Class 1 - to buy), otherwise, we set it to be 0(namely Class 2 - to wait).

B. Data Description and Interpretation

Same as regression problem.

C. Solving Imbalanced Data Set

Concerning a classification problem, an imbalanced data set leads to biased decisions towards the majority class and therefore an increase in the generalization error. As referred in the section of Data Description and Interpretation of the regression problem, the number of samples per class in our problem is not equally distributed. To address this problem, we considered three approaches.

- **Random Under Sampling** - Randomly select a subset the majority classes' data points so that the number of data points of each class is equal to the number of points of the minority class - Class 1.
- **Random Over Sampling** - Randomly add redundancy to the data set by duplicating data points of the minority classes so that the number of data points of each class is equal to the number of points of the majority class - Class 2.
- **Algorithmic Over Sampling** - Add redundancy to the data set by simulating the distribution of each minority class and consequently creating new data points so that the number of data points of each class is equal to the number of points of the majority class - Class 2.

Having these methods in mind, firstly, the *Random Under Sampling* would not be useful in out problem, because in our problem, the data set is very unbalanced, i.e. the buy entries is very sparse. If we use *Random Under Sampling*, we will lose many information. Secondly, if we use *Algorithmic Over Sampling*, it will add many noises into the data, because we do not know the hidden relationship between the features and the output. As a result, we prefered the second method, which is *Random Over Sampling*

D. Classification Model Construction

1) Neural Networks: The theory is described in the regression problem section, but the hyperparameters are different from it.

Finally, we defined three layers, namely the input layer, the hidden layer with 7 neurons and the output layer. We set the output's activation function to be sigmoid function. So the output of the data is the probability of buying the ticket on that query date. The nonlinearity function used by other layer is $rectifier$, which is simply $max(0, x)$. The neural net's weights are initialized from a uniform distribution with a cleverly chosen interval.

The update function will update the weights of our network after each batch. We used the $nesterov\_momentum$ gradient descent optimization method to do the job.

Although it is a classification problem, due to its probability property, it can also be seen as a regression problem. So we specified the mean squared error(MSE) as an objective function to minimize.

Then the data provided in X were spit into a training and a validation set, using 20% of the samples for validation.

V. Experiment Results

A. Performance Benchmarks

The naive purchase algorithm, called the *Random Purchase*, is to purchase a ticket randomly before the departure date. And the average price would be computed as the *Random Purchase Price*.

The lowest achievable cost is called the *Optimal Price* and it is the lowest price between the fist query date and the departure date.

B. Performance Metric

As long as we get the *Randome Purchase Price*, *Optimal Price*, and the *Predicted Price*, we can use the following performance metric to evaluate our results:

$$\textit{Performance} = \frac{\textit{Random Purchase Price} - \textit{Predicted Price}}{\textit{Random Purchase Price}}\% \tag{4}$$

$$\textit{Optimal Perfor} = \frac{\textit{Random Purchase Price} - \textit{Optimal Price}}{\textit{Random Purchase Price}}\% \tag{5}$$

$$\textit{Normalized Performance} = \frac{\textit{Performance}}{\textit{Optimal Performance}}\% \tag{6}$$

Having these metrics in mind, we uses the Normalized Performance to evaluate our results, because it normalizes every route and gives more intuition and it ranges from 0% to 100%, in which case, the higher the better.

C. Regression Performance Comparison

D. Classification Performance Comparison

VI. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] O. Etzioni, R. Tuchinda, C. A. Knoblock, and A. Yates, *To buy or not to buy: mining airfare data to minimize ticket purchase price*, in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2003, pp. 119128.

	Model				
Routes \ Method	Optimal	NN	Decison Tree	KNN	Random Purch.
BCN→BUD	decimal	1	1	1	1
BUD→BCN	decimal	1	1	1	1
CRL→OTP	decimal	1	1	1	1
MLH→SKP	decimal	1	1	1	1
MMX→SKP	decimal	1	1	1	1
OTP→CRL	decimal	1	1	1	1
SKP→MLH	decimal	1	1	1	1
SKP→MMX	decimal	1	1	1	1

TABLE I

PERCENTAGE-BASED PERFORMANCE COMPARISON OF VARIOUS DECISION THEORETIC APPROACHES ACROSS A COMBINATION OF 8 ROUTES.

One	Two	Three	Four	Five	Six	Seven	Eight	Nine	Ten	Eleven	Twelve	Thirteen	Fourteen	Fifteen
Fifteen	Fourteen	Thirteen	Twelve	Eleven	Ten	Nine	Eight	Seven	Six	Five	Four	Three	Two	One

TABLE II

HERE IS A CAPTION.