Mathematical Problems of Computer Science 53, 29–38, 2020.

UDC 519.6

Estimating Time of Driver Arrival with Gradient Boosting Algorithms and Deep Neural Networks

Henrik T. Sergoyan

American University of Armenia
e-mail: henriksergoyan@gmail.com

Abstract

Customer experience and resource management determine the degree to which transportation service providers can compete in today's heavily saturated markets. The paper investigates and suggests a new methodology to optimize calculations for Estimated Time of Arrival (from now on ETA, meaning the time it will take for the driver to reach the designated location) based on the data provided by GG collected from rides made in 2018. GG is a transportation service providing company, and it currently uses The Open Source Routing Machine (OSRM) which exhibits significant errors in the prediction phase. This paper shows that implementing algorithms such as XGBoost, CatBoost, and Neural Networks for the said task will improve the accuracy of estimation. Paper discusses the benefits and drawbacks of each model and then considers the performance of the stacking algorithm that combines several models into one. Thus, using those techniques, final results showed that Mean Squared Error (MSE) was decreased by 54% compared to the current GG model.

Keywords: GG, Estimated Time of Driver Arrival, OSRM, XGBoost, CatBoost, Neural Networks.

1 Introduction

ETA calculation is a relatively significant product differentiator for firms competing in the automated ride service hailing industry. ETA computation is particularly difficult for cities with underdeveloped traffic and road management services such as Yerevan, thus requiring novel methodologies and datasets to design models that can be deployed for use by relatively fickle customers. GG is one of the major transportation service providers in Yerevan and this paper uses data provided by GG to predict ETA. While GG currently estimates the time of arrival, it does so with a significant error. This paper discusses optimization techniques to calculate estimated arrival time with more accurate and flexible algorithms.

1.1 Problem Setting and Description

GG has an application that enables drivers to receive orders to transport customers and goods. Having chosen a pick-up location through the app, the customer gets an ETA for the driver. The task is to improve ETA estimation so that it is more accurate than the baseline OSRM model, which GG currently employs. To do so, the paper uses several joint Machine Learning algorithms such as XGBoost, CatBoost, and Neural Networks. The dataset provided by GG consists of more than 2.4M orders occured (received) in 2018. The number of unique drivers exceeds 6000. In addition to spatio-temporal dataset, the hourly data about weather conditions were used to better capture possible traffic jams.

1.2 Structure of the Paper

The paper consists of five central sections. The first section provides the problem statement and description, the overall summary of the paper structure representing the current method GG uses for estimating the arrival time of the driver while describing its drawbacks. After giving full understanding about the reason for further investigations, the next 3 sections, give detailed information and clarification of alternative models and outline advantages and disadvantages of each of them showing the results run on the 1-year information containing data. Based on the preceding sections the fifth part of the paper shows comparisons of all 3 suggested models and interprets the most optimal solution of the problem based on 3 methods.

1.3 Description of OSRM

The Open Source Routing Machine is a routing engine that works to find the shortest paths in the road networks, and it supports Linux, FreeBSD, Windows and Mac OS X platforms. The primary function of the environment is to compute and output the shortest path between the origin and the destination points, on the basis of which the system can compute the estimated time within a couple of milliseconds for the transport to reach a particular destination. By implementing contraction hierarchies, pure route computation takes the minimal time of those calculations. The most significant part of the calculation is finding the route and transmitting the geometry over the network. The decision-making capacity of the system is based on routing algorithms with road network data from the OSM (OpenStreetMap) project (see [1]).

The OSRM optimization algorithm prioritizes the speed with which calculations are made. Data obtained from OSM consist of 3 main components; namely nodes, ways and relations between them. Nodes determine the geometric structure of the path. Ways are 2D polylines consisting of line segments. Several lines can share the same node if they intersect with each other. A relation relates nodes or ways to turn restrictions, routes, and other features [2].

In these graphs, nodes represent directions of an OSM segment and graph edges connect graph nodes by describing the transition from one specific point to another. Here is how it works:

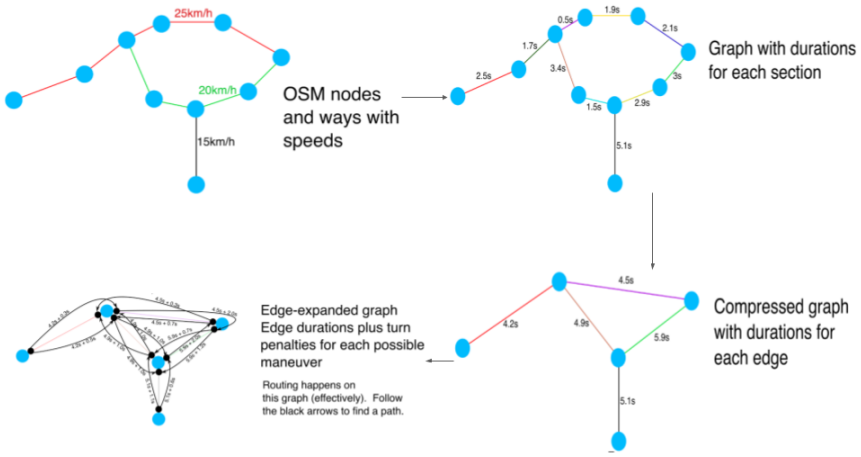

Fig. 1. OSRM visual implementation.

Even though this method seems to expedite the process of calculating ETA, the major drawback is that it does not capture the impact of weather conditions, traffic conditions and other real-life situations that depend on human factors. Thus, this paper concentrates on improving ETA calculation using information concerning weather and concrete data generated by drivers on actual routes.

2 Machine Learning Approach to the Problem

2.1 Related Work

Managing traffic among the urban population is becoming an increasingly significant issue for major municipalities. Thus, one of the fundamental problems to be solved is the efficient management of road traffic by minimizing congestion and assisting travelers with real-time information. Some urban areas currently use datasets that include information concerning GPS coordinates of origins and destinations, travel time, travel distance, pickup date, trip start-end times and the total fare to organize the road management. By analyzing data, route optimizing environments provide information for travelers related to the optimal routes, road conditions, and locations of incidents [3], [4].

Ishan Jindal, Tony (Zhiwei) Qin, Xuewen Che, Matthew Nokleby, Jieping Ye studied this topic and used the Unified Neural Network Approach to estimate Travel Time and Distance for a Taxi Trip. The data they used were structurally similar to those provided by GG, so this paper takes its results as an additional benchmark and aims to improve upon it [5].

Their research considers the waiting time at intersections for travel time estimations. The method used is a particular case of the path-based approach, where they add predictions of waiting time at intersections of sub-paths including the neighbor-based method by averaging the travel time for all the samples in the training data that have the same origin, destination, and time of day. Thus, the paper focuses on predicting travel time and distance from a source to a target as a function of the time of day based on NYC travel trip historical data. Having a large amount of training data, the authors build a unified neural network learning model, which jointly learns the travel time and distance between the origin and destination [5].

To understand what the travel time is, one needs to think of the time taken by the driver to move from the initial location to the final location including velocity changes or stops

made by him/her depending on some conditions. Travel distance is the path taken by the driver to reach from the primary point to the final location. Thus, the travel time depends on the origin and destination at a particular time of the day [4].

2.2 XGBoost: Extreme Gradient Boosting

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.), artificial neural networks tend to outperform all other algorithms or structures. However, when it comes to small-to-medium structured/tabular data, decision-tree-based algorithms are considered best-in-class (see [4]).

XGBoost algorithm was developed as a research project at the University of Washington. Tianqi Chen and Carlos Guestrin presented their paper at SIGKDD Conference in 2016 and caught the Machine Learning world by fire. Since its introduction, this algorithm has not only been credited with winning numerous Kaggle competitions but also for being the driving force under the hood for several cutting-edge industry applications. XGBoost is an ensemble learning method, and the main principle behind it is that a group of weak learners come together to form a keen learner. Bagging and boosting are two widely used ensemble learners [4], [6].

2.2.1 Bagging

While decision trees are one of the most easily interpretable models, they exhibit highly variable behavior. Therefore, several decision trees are being generated in parallel and form the base learners of bagging technique. Data sampled with replacement is fed to these learners for training. The final prediction is the averaged output from all the learners. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner [6].

2.2.2 Boosting

In boosting, the trees are built sequentially so that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals [6].

Using XGBoost, in this case, has several advantages. First of all, XGBoost has a feature of out-of-core computing that helps to handle massive datasets that do not fit into memory. Moreover, because of a block structure in its system design, XGBoost can make use of multiple cores on the CPU and GPU. Data are sorted and stored in in-memory units called blocks. That enables the data layout to be reused by subsequent iterations, instead of computing it again. Finally, XGBoost has an option to penalize complex models through both L1 and L2 regularizations. Regularization helps in preventing overfitting [6].

2.3 CatBoost: Categorical Boosting

CatBoost (Categorical Boosting) is a state-of-the-art open-source gradient boosting on decision trees library developed by Yandex. The algorithm is spawned from XGBoost, but it has its advantages compared to all other gradient boosting algorithms. Although XGBoost

became a game-changing algorithm in machine learning, it indeed, has its drawbacks. For example, XGBoost has a problem of dealing with categorical features, and therefore, one-hot encoding is used, which adds a new binary feature for each category. However, in the case of high cardinality features, such as User ID, Partner ID, which were present in the data, such a technique leads to an infeasibly large number of new features. CatBoost deals with this issue and is now considered the most innovative algorithm for processing categorical features. At the end of the paper, it is shown how well CatBoost deals with categorical features in the data by looking at the final accuracies of the models [7].

2.4 NN: Neural Networks

Neural Networks were used as the last method to predict the time of arrival of a driver. The main advantage of neural nets is that they can detect patterns that no other algorithms can discover. The logic behind it differs significantly from the aforementioned boosting algorithms [8].

- The network architecture includes an input layer, a hidden layer(s) and an output layer. It is also called MLP (Multi-Layer Perceptron) because of the multiple layers.

- The hidden layer can be seen as a distillation layer that concentrates some of the essential patterns from the inputs and passes them onto the next layer to view. It renders the network faster and more efficient by identifying only the critical information from the inputs and leaving out redundant information.

- The activation function serves for two notable purposes [8]:

    - captures non-linear relationships between the inputs;
    - helps to convert the input into a more useful output.

2.5 Feature Engineering and Final Results

To train the above-mentioned models, annual data generated by GG taxi orders were used. The data consists of more than 2M unique orders received in 2018. The features of the data were the driver ID, the coordinates where he takes the order, the coordinates of the order denoted by the user, the time-stamp when the order was created, the actual duration of the trip, and the estimated duration of the trip by OSRM. However, several extra features were added to the data, such as dummy variables of weekdays or weather information. It is important to mention that weather data contains hourly historic information about weather conditions, such as temperature, humidity, pressure, wind speed and facts about raining or snowing. The data contained outliers and missing values, and each of them happened because of failure of a mobile program. However, all this misleading information was identified and removed.

2.5.1 Training Process and Evaluation

The evaluation of the proposed model and all the considered baselines are on the mean absolute error (MAE). The task of each model was to minimize MAE.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - z_i|}{n}$$

Cross validation was used to prevent overfitting while training the XGBoost and CatBoost models. Moreover, as these algorithms require specifications of a lot of parameters, Grid Search was used to identify the best parameters for each model. Below are bar graphs depicting the importance of features used to train XGBoost and CatBoost algorithms. Both models indicate that the most important feature is the distance. However, the remaining features are represented as having varying levels of importance for both respective models. For example, CatBoost illustrates that the second most important feature is Partner ID, while the second most important feature for XGBoost was the hour of the day in which the trip occurred. It is worthwhile to mention that both models indicate that OSRM estimates ("from start to take") are important features. This makes considerable sense, since OSRM estimates include information concerning the waiting time at traffic lights, the number of turns, the angle of turns, and the mean velocity between turns.

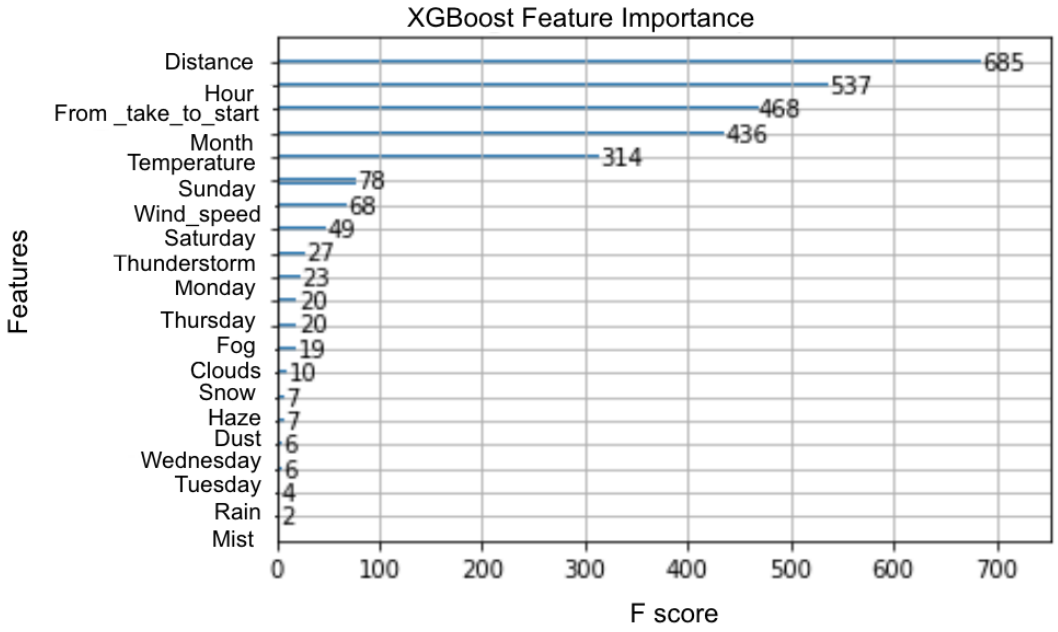

Fig. 2. Importance of Features for XGBoost algorithm.

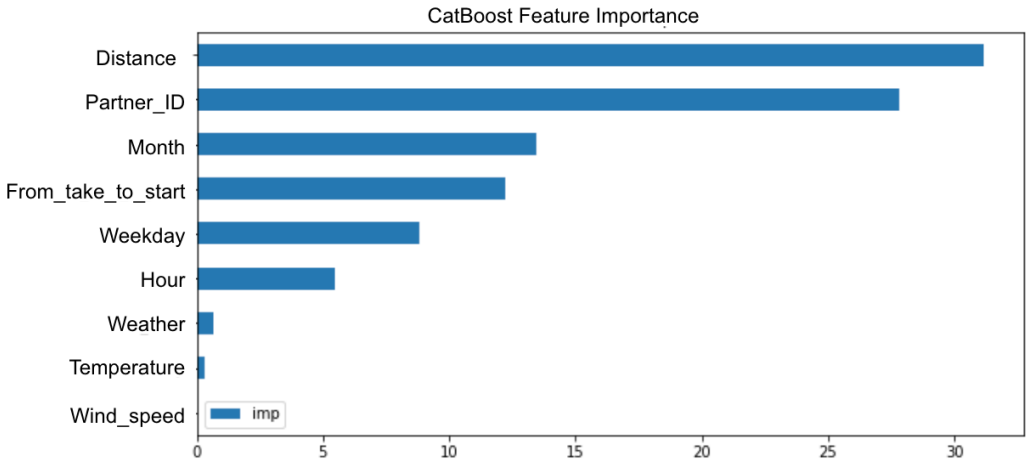

Fig. 3. Importance of Features for CATBoost algorithm.

Neural networks built with Keras library require some intuitive approach to identify the number of neurons, hidden layers, and types of activation functions for each layer. Based on experimentation, the optimal number of neurons and the most appropriate activation function to minimize MAE were determined.

As the last stroke of the brush, the predictions of those trained models and actual time were combined in one dataset and a linear regression model was trained on the said dataset. In machine learning, this process is usually called stacking, where it is expected that the ensemble of several models provides a better result than each of those models independently. However, this example clearly illustrates that this is not always the case. Although our stacked model outperforms XGBoost and Neural Networks, it still has a slightly higher MAE than CatBoost regressor. This fact can be interpreted in the following way: our three models mostly pick up and model the same information quite similarly and therefore, we did not get much out of an ensemble. Below the performance of each model for train and test sets is presented.

Table 1: MAE results for each method.

Methods	Train	Test
OSRM	205.796	205.787
XGBOOST	98.77	98.96
CATBOOST	92.34	94.97
NN	98.09	98.34
Stack	93.03	95.01

3 Conclusion

In this paper, we proposed a new approach that formulates the ETA calculation as a pure regression problem. As such, we train CatBoost, XGBoost, and Neural Networks and successfully improve upon benchmarks set by the OSRM architecture used by GG and the Unified Neural Network approach. The CatBoost model outperformed XGBoost and Neural Network. Thus, while OSRM exhibits errors of 205.796 seconds and 205.787 seconds for

test and train data respectively, CatBoost model exhibits errors of 92.34 and 94.97 seconds on train and test data respectively. We believe that by increasing the number and quality of features and using more powerful models we can register significant improvements in predictive performance.

References

[1] M. Dodge and R. Kitchin, "Crowdsourced cartography: Mapping experience and knowledge", *Environment and Planning A: Economy and Space*, vol. 45, no. 1, pp. 19–36, 2013.

[2] S. Huber and Ch. Rust, "Calculate travel time and distance with openStreetMap data using the OpenSource Routing Machine (OSRM)", *The State Journal*, vol. 16, no. 2, pp. 416–423, 2016.

[3] Y. Li and K. Fu and Zh. Wang and C. Shahabi and J. Ye and Y. Liu, "Multi-task representation learning for travel time estimation", *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1695–1704, 2018.

[4] Zh. Wang and K. Fu and J. Ye, "Learning to estimate the travel time", *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 858–866, 2018.

[5] I. Jindal and T. Qin and X. Chen and M. Nokleby and J. Ye, "A unified neural network approach for estimating travel time and distance for a Taxi Trip", arXiv:1710.04350v1, [stat.ML], 2017.

[6] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

[7] L. Prokhorenkova and G. Gusev and A. Vorobev and A. V. Dorogush and A. Gulin, "CatBoost: unbiased boosting with categorical features", *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6639–6649, 2018.

[8] D. Wang and J. Zhang and W. Cao and J. Li and Y. Zheng, "When will you arrive? Estimating travel time based on deep neural networks", *AAAI*, pp 2500–2507, 2018.

Submitted 10.12.2019, accepted 18.04.2020.

Վարորդի ժամանման ժամանակի մոտարկումը ուժեղացված գրադիենտի ալգորիթմների և խորը նեյրոնային ցանցերի միջոցով

Հենրիկ Տ. Սերգոյան

Երևանի պետական համալսարան
e-mail: henriksergoyan@gmail.com

Ամփոփում

Հաճախորդների սպասարկումը և ռեսուրսների կառավարումը որոշում են տրանսպորտային ծառայություններ մատուցող ընկերությունների մրցակցելու կարողության աստիճանը այսօրվա մեծ և հագեցած շուկաներում: Այս աշխատանքն ուսումնասիրում և առաջարկում է նոր մեթոդաբանություն վարորդի ժամանման տևողությունը հաշվարկելու համար՝ հիմնվելով GG-ի 2018 թվականի ուղևորություններից հավաքված տվյալների վրա: GG-ն տրանսպորտային ծառայություններ մատուցող ընկերություն է և ներկայումս օգտագործում է The Open Source Routing Machine (OSRM) համակարգը, որը կանխատեսման ժամանակ աշխատում է էական սխալներով: Այս աշխատանքում ցույց է տրված, որ նշված առաջադրանքի համար այնպիսի ալգորիթմների իրականացումը, ինչպիսիք են XGBoost, CatBoost և խորը նեյրոնային ցանցերը, կբարելավի ժամանակի ճշգրտությունը: Աշխատանքում քննարկվում են յուրաքանչյուր մոդելի առավելությունները և թերությունները և այնուհետեւ դիտարկվում է կուտակային ալգորիթմի տարբերակը, որը միավորում է մի քանի մոդելները մեկի մեջ: Այսպիսով, այդ տեխնիկայի կիրառմամբ ստացված վերջնական արդյունքները ցույց են տվել, որ միջին քառակուսային շեղումը GG-ում կիրառվող նախորդ մոդելի նկատմամբ նվազել է 54%-ով:

Բանալի բառեր՝ GG, վարորդի ժամանման տևողություն, OSRM, XGBoost, CATBoost, նեյրոնային ցանցեր:

Оценка времени прибытия водителя с помощью алгоритмов градиентного усиления и глубоких нейронных сетей

Генрих Т. Сергоян

Американский университет Армении
e-mail:henriksergoyan@gmail.com

Аннотация

Обслуживание клиентов и управление ресурсами определяют степень, в которой поставщики транспортных услуг могут конкурировать на сегодняшних сильно насыщенных рынках. В этой работе исследуется и предлагается новая методология оптимизации для расчетов предполагаемого времени прибытия водителя на основе прошлогодних данных, предоставленных GG, собранных из поездок водителей компании. GG является компанией, предоставляющей транспортные услуги, и в настоящее время она использует The Open Source Routing Machine (OSRM), который на этапе прогнозирования демонстрирует

довольно существенные ошибки. Эта работа показывает, что реализация алгоритмов, таких как XGBoost,CatBoost и Neural Networks для указанной задачи, повысит точность расчета. В статье обсуждаются преимущества и недостатки каждой модели, а затем рассматривается производительность алгоритма суммирования, который объединяет несколько моделей в одну. Таким образом, окончательные результаты, полученные с использованием этих методов, показали, что средняя квадратическая ошибка уменьшилась на 54% по сравнению с текущей моделью GG.

Ключевые слова: GG, расчетное время прибытия водителя, OSRM, XGBoost, CatBoost, Neural Networks