Capstone Project Proposal

Machine Learning Engineer Nanodegree

Version 2.0

Mike Leske

August, 16 2018

Proposal

Domain Background

New York City is world-famous for its amount of yellow cabs. With the rise of market disrupters like Uber and Lyft it is essential to upfront be aware of the expected ride cost to select the 'best' option is terms of monetary benefit for the user. Kaggle currently hosts a competition together with its partners Google and Coursera aiming to predict the taxi fare. [1]

Problem Statement

The problem statement of this Kaggle competition is defined as "predicting the fare amount (inclusive of tolls) for a taxi ride in New York City given the pickup and dropoff locations". [1] The predicted fare amount is to be provided as US Dollar and can serve as an estimate for a future taxi rides.

A comparable problem statement arose with the publication of the Boston House Prices dataset [2] which aims to predict the value of a property given a set of provided characteristics, like number of rooms, neighbourhood, etc.

Datasets and Inputs

The dataset [3] to be used for this competition is provided by Kaggle and its partners Google and Coursera. It contains two essential files: 1) a train.csv file with a training set of 55 million taxi rides and 2) a test.csv file with a test set of approx. 10,000 taxis rides. For each observation (individual taxi rides) the following parameters are provided: key, pickup_datetime, pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude, passenger_count. While the official training set (train.csv) contains the final taxi ride fare, the official test set used for participation in the Kaggle competition does not provide this final insight.

Solution Statement

The New York City Taxi Fare Prediction Kaggle competition presents itself as a classical regression type of problem, which is a well-studied domain with several potential solutions available ranging from classical Machine Learning (ML) algorithms to more sophisticated Deep Learning (DL) architectures based on Artificial Neural Networks (ANN). As described in the Datasets and Input section, the provided training data is labelled with typical taxi ride variables like pickup and dropoff coordinates and the number of passengers as well as the “label” in terms of cost of the ride (‘fare_amount’). Therefore, the solution to the prediction problem can be targeted using Supervised Learning algorithms.

The following specific algorithms/solutions will be evaluated and compared:

- Multivariate Linear Regression in Python using the scikit-learn library
- Gradient Boosting (XGB) in Python using the xgboost and scikit-learn libraries
- Deep Learning ANN in Python

Benchmark Model

When the Kaggle competition was launched a starter solution using a Simple Linear Model was provided to establish a baseline performance. [4] This solution calculates the taxi travel distance from the provided pickup and dropoff longitude/latitude values and subsequently executes numpy matrix multiplication operations.

This Simple Linear Model results in a competition evaluation metric (root-mean-sqared-error, see section Evaluation Metrics) of 5.74.

The solutions to be implemented as part of this capstone project will be measured and compared against this baseline performance value. The Kaggle Leaderboard will effectively be used for the benchmark comparison after the results on the test data have been calculated by a potential solution algorithm and been uploaded to Kaggle for evaluation.

Evaluation Metrics

As the problem statement points into the direction of a regression analysis, the competition evaluation is based on the root mean-squared error (RMSE) [5][6]. RMSE is well-suited for this problem statement as it measures the difference between the predicted taxi fares and the real taxi ride fares provided with the training dataset. In this sense the taxi ride fare is the regression's dependent variable.

RMSE is calculated by:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

A smaller RMSE means the predicted values are close(r) to the real fares of the training set than the predictions of a model with a resulting larger RMSE.

Therefore, the solution for this Kaggle competition seeks to minimize the RMSE.

Project Design

The initial phase of the project targets 'data preprocessing' and includes at least a 'data cleaning' and a 'feature engineering' step.

The 'data cleaning' step includes, but may not be limited to:

- Drop observations with missing data
- Identify and remove training observations with unreasonable 'fare_amount', e.g. negative or extremely large
- Identify and remove extremely short taxi rides, e.g. taxi rides shorter than 0.05 miles seem unexpectedly short
- Identify and remove extremely long taxi rides as they might reflect a diverging cost model (e.g. fixed price)
- Restrict training data to observations with reasonable GPS locations (some isolated observations have longitude and latitude coordinates swapped)

The 'feature engineering' step includes, but may not be limited to:

- Calculate actual ride distance from pickup/dropoff longitudes/latitudes
- Extract additional temporal insights from pickup timestamp, e.g. year, day in week and hour to detect hidden patterns like rush hours
- One-Hot-Encode categorical columns

Additional considerations:

- While 'data cleaning' is only applied to the test dataset, the 'feature engineering' will be applied to both training and test datasets
- As the training and test datasets contains isolated taxi ride observations no time-series considerations are planned
- The training set is very large and may impose memory limitations. Therefore, training with smaller chunks of the training set may be necessary, e.g. chunks of 10 million training observations at a time.

Next, the different ML & DL algorithms as outlined in the Solution Statement will be implemented in Python using the respective libraries. Hyper-parameters for the ML/DL solutions are subject to experimentation during the project execution.

The Multivariate Linear Regression in Python using the scikit-learn library will strictly fit the training data and subsequently predict the dependent variable for the test data.

The Gradient Boosting (XGB) in Python using the xgboost and scikit-learn libraries allows to influence the algorithm by setting hyper-parameters such as learning_rate (eta), the depths of the resulting boosting trees and the number of boosting rounds. The scikit-learn train_test_split function will be used to split the training set into training (90%) and validation (10%) sets.

Deep Learning ANN in Python will be implemented using the fastai high-level API (using Pytorch as Deep Learning framework) and offers the technique of an "embedding matrix" to represent hidden patterns in categorical data. Continuous variable will be scaled into the interval [-1, 1] to avoid bias towards variables with larger actual values. Hyper-parameter tuning will involve selecting an appropriate learning_rate, the number of hidden neural layers, the size of the hidden neural layers and the number of training iterations. As with Gradient Boosting, the training data will be split into 90% actual training data and 10% validation data.

In order to compare the different solution approaches, the training process based on the validation training set provided.

As a last step, Kaggle competition submission files will be created based on the trained solutions using the provided test set (test.csv). The results for this step can only be obtained via the Kaggle Leaderboard.

Links

[1] https://www.kaggle.com/c/new-york-city-taxi-fare-prediction
[2] https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html
[3] https://www.kaggle.com/c/new-york-city-taxi-fare-prediction/data
[4] https://www.kaggle.com/dster/nyc-taxi-fare-starter-kernel-simple-linear-model
[5] https://www.kaggle.com/c/new-york-city-taxi-fare-prediction#evaluation
[6] https://en.wikipedia.org/wiki/Root-mean-square_deviation