Reinforcement Learning Project Report

Profit Maximization in Taxi Hailing Service

Abdulmueez Emiola, Ozioma Okonicha, Pavel Tishkin
a.emiola@innopolis.university,o.okonicha@innopolis.university,p.tishkin@innopolis.university

1 Introduction

Taxi is a popular means of transport that aims to ease the movement between different places for an affordable price. There are several competitors in the Russian market, such as Uber, Yandex Taxi. For taxi drivers it is vital to know, where their vehicle should be located and which requests they should get to maximize their profit.

In this report, we present an overview of our course project, and show the various aspects including the task description, problem statement, environment setup, data preprocessing, implementation, results, challenges faced and future work.

1.1 Task Description

The task involves designing and implementing a reinforcement learning algorithm that can learn an optimal strategy for taxi trips in order to maximize proft. Given historical data on taxi trips, we build an environment, from which an agent can sample taxi orders and decide which to take. The RL agent learns to distinguish orders worth taking over.

1.2 Problem Statement

Big technological corporations collect astronomical amounts of data, regarding the taxi drives. They can collect and store several metrics, such as the price for the drive, the amount of time it took, the distance of the ride. Big departments analyze this data and provide sophisticated algorithms to decide the price for the ride. This information is not available to the taxi drivers. They do not have means to analyze data and gain useful insights from it. The goal of this project is to provide some help to the drivers and develop an RL agent that can learn to maximize the daily profit, based on the historical data we obtained from NYC taxi drivers dataset.

2 Solution

Our solution is all about testing out five different reinforcement learning algorithms, analyzing them, their results and understanding which best solves the problem of profit maximization in taxi hailing services. Before being able to do that, we need our learning environment. Since we rae creating ours from scratch as opposed to using something from openai gym for example, we will explain in depth the variables of our environment below.

2.1 Environment and Variables

The environment consists of simulated taxi system that at each timestamp offers the driver a set of 1o best offers, a driver can take. We define it formally as:

- $\mathcal{S}$ - (hour, minute, current_zipcode)
- A - 10 orders, represented as (pick_zip, drop_zip), or waiting for an hour (-1, -1)
- R - calcultaed as a mean reward a taxi driver would get, if they travel from pic_zip to drop_zip. If the driver stays for an hour, they do not get any reward

States are transitioned the following way:

- Driver decides an order to take
- If driver is not on the same location as the order, time passes, as the driver goes to the pickup location
- The driver takes the passenger, and drives them to destination location, which consumes time
- The driver gets a reward, which is the mean reward a driver would get for taking such trip, based on the statistics we got from the NYC Taxi driving dataset

The environment consists of a simulated taxi system where the RL agent interacts with the environment by adjusting fares and observing the outcomes in terms of profit and customer satisfaction. The variables include pickup location, drop-off location, time of day, day of week, trip duration, trip distance, passenger count, fare amount, and total amount paid.

3 Exploratory Data Analysis

We conducted an extensive exploratory data analysis (EDA) to gain insights into the taxi trip data. At first we imported the necessary libraries and loaded the dataset, then we cleaned the data by removing leading or trailing spaces from column names. Finally we combined datasets based on shared columns and removing some incorrect coordinates and extreme values for different trip features.

Some feature engineering also needed to be done when defining zones based on pickup and dropoff coordinates. Then we extracted datetime features. Finally in order to group the locations into different zones we did KMeans clustering for five zones [1]. We can see this below:

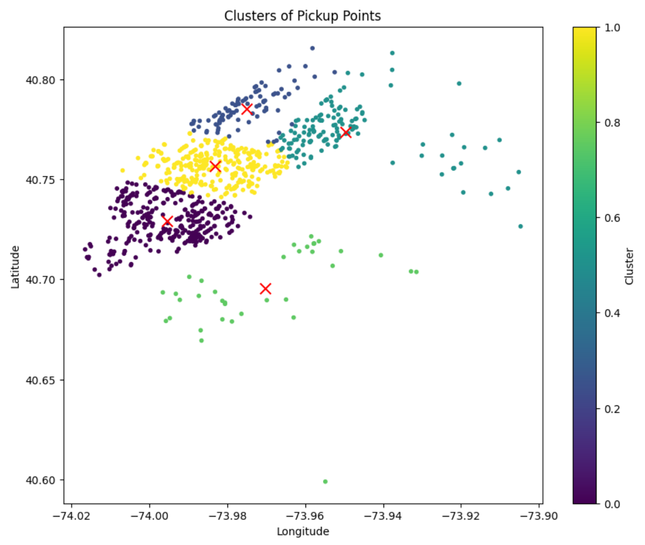


Figure 1. Kmeans clusters

4 Data Preprocessing

- **Sampling**: Initially, our dataset consisted of 12 csvs, with each csv consisting of approximately 15 million rows. To manage computational costs, we opted to subsample the dataset, selecting only one CSV corresponding to a single month.
- **Location Transformation**[2]: To enhance the location feature, we transformed both the pickup and dropoff coordinates (latitude and longitude) into zip codes. This allowed us to condense the location data considerably while preserving its essential meaning. However, generating zip codes from coordinates is resource-intensive. Consequently, we selected only a million rows for this transformation process.
- **Summary Statistics**: Furthermore, we grouped our data by the pickup hour, pickup, and dropoff location. This grouping process resulted in the creation of a new summary dataset comprising approximately 70,000 rows, distilled from our original dataset. The summary dataset included the following fields:
  - **Hour**: The hour of the day the pickup takes place.
  - **Pickup Location**: The zipcode the pickup takes place.
  - **Dropoff Location** : The zipcode the dropoff takes place.
  - **Number of transitions**: The number of ride between the pickup location and dropoff location in that hour.
  - **Mean Trip Distance**: The average distance of a trip between pickup location dropoff location in that hour.
  - **Mean Trip Duration**: The average time in seconds of a trip between pickup location dropoff location in that hour.
  - **Mean Trip Fare**: The average total fare of a trip between pickup location dropoff location in that hour. These value includes tips.

5 Implementation

We implemented different algorithms in order to practice what we learnt during the course and see what applying them in a more real-world environment will yield. Below we will explain in details each of the algorithms we implemented in the environment we already described above.

5.1 RandomAgent

To provide a baseline for our work, we started with a random agent. The agent chooses an action at random without paying attention to the current state.

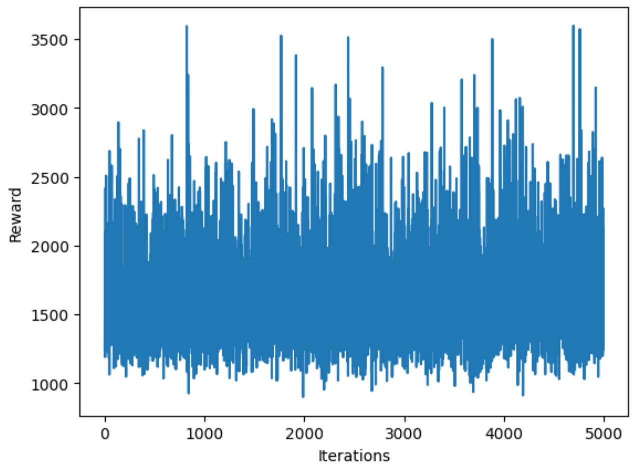


Figure 2. RandomAgent reward

5.2 SARSA Agent

SARSA (State-Action-Reward-State-Action) is an on-policy RL algorithm for finding optimal policy for a decision-making problem. In our own taxi simulation environment, the agent chooses a random starting point from the environments state space for initialization. It then uses an epsilon-greedy policy to select actions, updates q-values and repeats. For the SARSA we needed to perform state and action mapping in order to fit our defined environment.

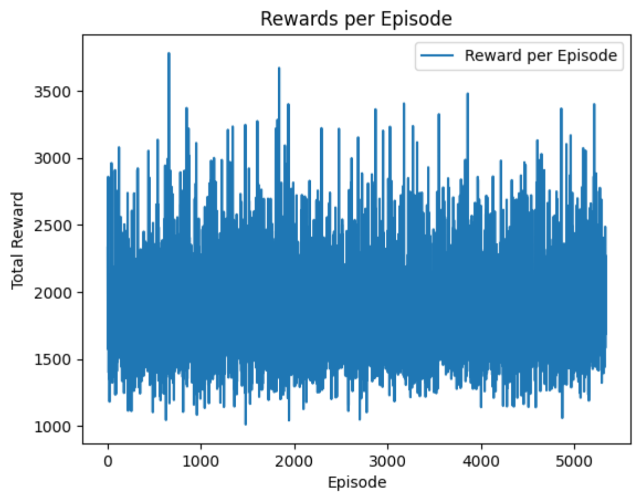


Figure 3. SARSA reward

5.3 Q-Learning

The next algorithm we applied is Q-learning. It is also an on-policy algorithm for learning the optimal decision making. Prerequisites to the algorithm are the same as for SARSA, however the updating process of Q-table is a bit different. It assumes greedy approach to next action in the next state. In our case, however, application of Q-Learning is a bit problematic, since we do not know the available actions beforehand. For reward approximation, we decided to assume all of the actions are available from the next state.

Below you can observe the training process and the rewards per episode for Q-Learning algorithm:

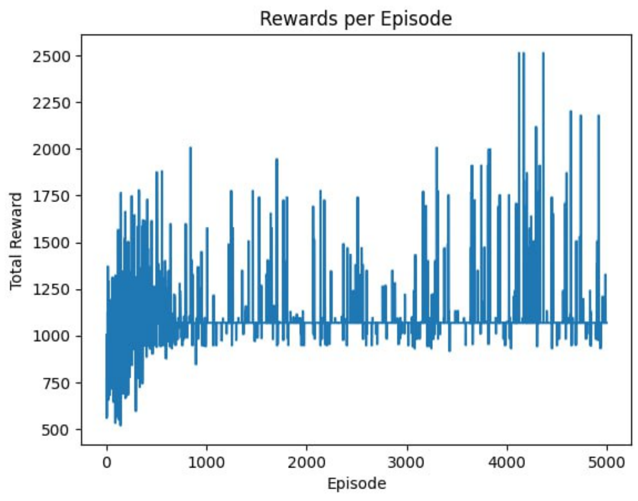
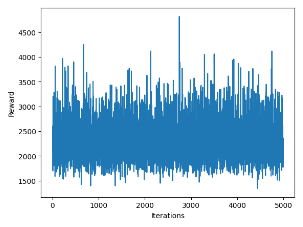


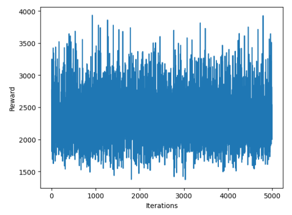
Figure 4. Q-Learning reward

5.4 Deep Q-Learning

Due to our state size, we resorted to using deep learning. To this end, we employed deep q learning. Q-Learning uses a deep neural network to approximate the different Q-values for each possible action at a state. In our q-learning algorithm, we employed experience replay to make efficient use of experiences and fixed targets to stabilize the algorithm. We also implemented **double deep q-learning**.



(a) Deep Q-Learning Network Reward



(b) Double Deep Q Learning Network

Figure 5. Comparison of Rewards

6 Results

The trained RL agents demonstrated the superiority of the deep learning methods, since they obtained highest mean values. As we can observe, most of the methods have high variance, with the exception of the Q-Learning algorithm. Other than that, we can see that Deep Q-Learning and Double Deep Q-Learning methods, while retaining high variance, managed to outperform the baseline to some degree. Q Learning is the only method, consistently underperforming throughout the training process. Maybe something went wrong with the environment, we trained it, but this may be the consequences of our bold assumption that all the actions are available to the model in the training phase.

Agent	Reward Mean	Reward STD
Deep Q Learning	2320.93	381.50
Double Deep Q Learning	2314.11	386.19
Q Learning	1074.54	112.92
SARSA	1861.225	358.469
Random Agent	1632.196	356.37

Table 1. Reward properties for each model

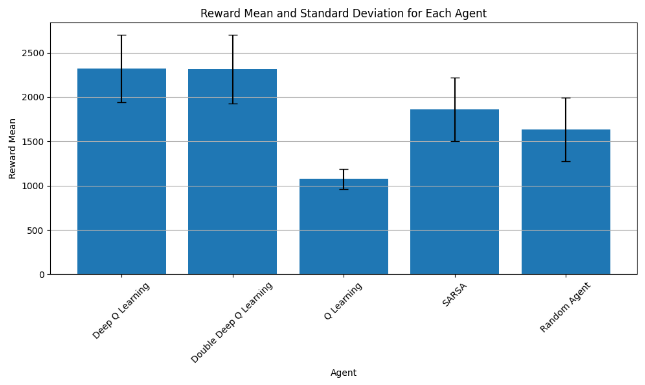


Figure 6. Agent reward comparison

7 Challenges

One of the greatest challenges with the project was the necessity to build an environment from scratch. As such, we had to make a lot of assumptions and simplify the real world scenario of taxi driver work. Some misleading and incorrect assumptions about the process could significantly reduce the quality of the end system. To address this problem, we ensured that our decisions about simplification were based on other research on the topic.

The other challenge is the size of our state-action space. It made us difficult to analyze the problem, using simple Q-learning and SARSA. To address the problem, we had to introduce several optimizations to the data storage and Q-values retrieval.

Throughout the project, we encountered several challenges such as data preprocessing complexities, algorithm convergence issues, and tuning hyperparameters for optimal performance. However, we addressed these challenges through rigorous experimentation, algorithm adjustments, and collaboration within the team.

8 Future Work

In the future, we plan to improve the model of the environment by providing a better heuristic for generating the possible action space (taxi requests at a particular time).

We plan to also improve our summary dataset by using a deep clustering neural network for generating the training dataset. This will help us to better learn important features that we couldn't capture via classical approaches and can also be used to decide the duration and reward for taking an action.

9 Conclusion

Through the application of deep reinforcement learning techniques, we succeeded in realizing a notable 40% enhancement in the profits generated by a random agent. This outcome aligns closely with our overarching goal of augmenting taxi driver profits, underscoring the efficacy of our approach.

References

[1] Rui Chen et al. "Reinforcement Learning for Taxi Driver Repositioning Problem in NYC". In: *Data Science Institute, Columbia University* (2020). URL: https://datascience.columbia.edu/wp-content/uploads/2020/12/35_Didi_Reinforcement-Learning-for-Taxi-Driver-Repositioning-Problem-in-NYC.pdf.

[2] Benjamin Lampert Jingshu Wang. "Improving Taxi Revenue With Reinforcement Learning". In: *cs229.stanford* ().