CSCE 585: Machine Learning Systems: Homework #1

Shima Oruji

September 9, 2022

Contents

1 Name of the Company

For this homework, we decided to focus on ***Uber***. Uber uses a lot of machine learning models used in different parts of its application and business. For example, *pricing* is one of the famous applications of Uber that is enabled by machine learning. See reference #1 for high level information on the pricing model of Uber.

We must have noticed that sometimes, Uber offers different prices for the same route at different times. But how does it manage to calculate this price? It is all because of the machine learning-powered pricing algorithm. This algorithm is used to find the most reasonable prices based on that area's economy, weather, and current traffic conditions in addition to the distance of the trip. This looks like a regression problem where the model gets the aforementioned parameters as input and determines the offered price as output.

2 Number of Deployed Models and their Type

Unfortunately, we were not able to find the exact information for this part of the homework. However, we can do a high level estimation to figure it out. For the model type, I think they are using neural network models for this problem. Since they have access to massive amount of data, they can use them for training a deep learning model. The number of deployed models would heavily depend on the daily users of Uber. Let's assume that on average Uber is taking 5 billion rides (please see reference #2), which translates to 13.7 million rides/day. If we further break down the number, it becomes 160 rides/second. If we consider the busy times and peak usage, we Uber might receive 30 times of its normal requests, which makes the previous number 4,800 rides/second. So, the deployed system must be able to handle 4,800 price requests per second. Assuming that each of the deployed models can process 500 requests, Uber would need 10 parallel hosts to process this amount of price traffic. To have a cost effective deployment, we can change this number around-the-clock and depending on the demand forecast.

3 Used Ecosystem/Framework/Tools for Deployment

Based on reference #3, Uber uses Amazon Web Services (AWS) to develop and deploy their different application including their pricing model. There are different advanced frameworks available in AWS which Uber might use for the pricing application. The training pipeline can include S3 (for data storage), Sagemaker (for training jobs), lambda (for code execution), SNS (for invoking lambda), step functions (to orchestrate the tech workflow), and python (for code development). The deployment pipeline can include AWS code pipeline (for CI/CD), load balancer (for distributing the traffic among the hosts), lambdas (for executing the trained model), SNS, and Dynamodb (for caching the frequent similar requests). Note that lambda functions can automatically scale and are preferred choices in such applications. AWS cloud watch can also be used here for monitoring the health of the received requests and responses.

4 Load Balancing Between Models

As mentioned earlier, AWS load balancer can be used here. The load balancer can use a simple round robin scheme to distribute the traffic among different hosts. To handle the failure scenario in one/multiple number of the hosts, the load balancer can use more sophisticated algorithms such as consistent hashing, which is leveraged in most of the largescale and real-world applications.

5 Retiring Existing Models

The retraining of existing models could be periodical or based on their performance drop or even a hybrid criteria. In periodical criteria, the models are being trained and replaced on a periodical base (e.g., every hour, day, or week). In performance criteria, the performance of the existing models are continuously being evaluated during their execution and when their performance drops under a defined threshold, a new model starts being trained and deployed. In the hybrid method, an OR logic might exist for retraining an existing model and deploying a new one. In successful companies such as Uber, it is likely that they are using a hybrid algorithm. The performance criteria will be discussed in the next sections.

6 Updating the Models

I think when a new model is trained, it gets pushed through the code pipeline that they have. The new model is used with the old one in parallel (in a short A/B testing cycle) and when the performance improvement is confirmed for the new model, the old model gets deprecated and the new one processes the entire traffic.

7 Metrics for Monitoring Model's Performance

The online monitoring metrics of the model are very important as they are the deciding factors on the retirement of the existing models. In this specific example, the overall acceptance rate of the offered rides with the given prices is a reflecting metric on the success of the deployed model. If a model has a high ride acceptance rate, it means that both the customers and drivers are happy. If not, we can infer that the offered prices are either too high (which make the customer not to accept the rides) or too low (which make the drivers not to accept the rides).

8 Time for Model Deployment

In AWS, the time for deployment depends on the retraining scheme and the overall stages in the code pipeline. If we retrain the model in advance periodically and deploy it whenever the performance drops, the deployment could be very fast (less than 10 minutes to pass the unit tests and integration tests). However, this approach could be very expensive as we are retraining the model so many times without deployment. An alternate method could be start the retraining process once we received the performance drop trigger. This method is more cost efficient; however, the deployment time (1-5 hours depending on the training time) is much more than the previous method. Overall, I think a hybrid method is likely to be used in practice for Uber's pricing model.

9 Cost Decrease for Model Deployment

There are many factors that can be considered to decrease the deployment cost in our analyzed example. One important one was discussed in the previous section (the first approach has less deployment cost but the overall cost of the second approach is significantly less than the first one). One other factor that can reduce the deployment cost is to use effective caching algorithms. In this case, we can quickly respond to some of the similar requests and make fewer calls to the backend lambdas that are executing the deployed models (cost of AWS lambdas are calculated based on the number of calls to them). One other point is the number of parallel models that we deploy to process the received requests. As we discussed earlier, the pricing API of Uber must have 10 hosts during its peak time; however, if we always operate with 10 hosts, we will have low host usage during most of the time. Therefore, one effective strategy could be changing the number of deployed hosts around-the-clock and based on the demand forecasts that we have according to the historical data.