CS221 Project Proposal

Deep Queue-Learning: A Quest to Optimize Office Hours

Avoy Datta **ad9697**

Dian Ang Yap **dianang7**

Zheng Yan **yzh**

Note: We are planning to use the dataset we collect in this project in CS229 as well, with the same group members and same big-picture goal. Our current plans regarding how we partition our tasks between the two projects is described above, but may be slightly changed based on upcoming lectures. We want to ensure that we're putting material more suited for CS221 into the "CS221 part" of our project, and same with CS229.

Motivation

Among CS students at Stanford, the experience of queueing at office hours is practically universal. Unfortunately, the wait time is prone to high variance, resulting in students sometimes waiting 4-5 hours before receiving help. Not only are these instances stressful for students, they are stressful for TAs as well. Therefore, minimizing the wait times and optimizing queue lengths could benefit all parties. While these instances are not completely unpreventable, we strongly believe that they could be reduced — using the information that is gathered every day on Queuestatus. What if we had better tools to predict when students go to office hours, and how long each student is expected to take? If we had more processed information, then a lot of doors open for informed recommendations. In particular, if our goal is to minimize the expected service time overall, given some representation of TA constraints, we can even generate an optimal office hours schedule for instructors.

Scope, Desired Behavior, Challenges, and Intended Experiments

We plan to manually collect data of QueueStatus of Stanford CS classes, and utilize at least 3 years of data for each. We have constructed a pipeline for querying data from Queuestatus (through parsing JSON files using the curl package in R). The first part of our project is prediction: given a class, week, weekday, time, number of days before an assignment, assignment type, and closeness until midterm, we wish to predict the influx of students and expected service time per student. The unique part for CS229 in this project is experimentation of support vector machines, RNNs and regression trees for this in CS229, or complemented by neural networks for CS221. One major challenge to this part is feature selection and representation, i.e. removing, adding, or changing features for prediction. The second part of our project is optimization and recommendation (this is the part unique to CS221) - given a set of TA constraints for the week and expected student influx and expected service times for each "active" hour in the week (which are outputs from part 1, we assume to be correct), we wish to return a schedule that fits the constraints and correctly optimizes for lowest expected wait time. For this part, we hope to use uniform cost search. Defining our states and transitions will be the main challenge for this. The primary challenge for the "learning" phase of the model is feature extraction, identifying the most relevant features that impact queue wait times. One important factor that affects wait times for a given quarter such as efficiency and capability of course staff, but this is difficult to track reliably as the turnover rate for a lot of classes at QueueStatus is pretty high. We will thus try to use Bayes' Theorem with a prior to test the degree of independence between wait times and the course staff hired for a given quarter.

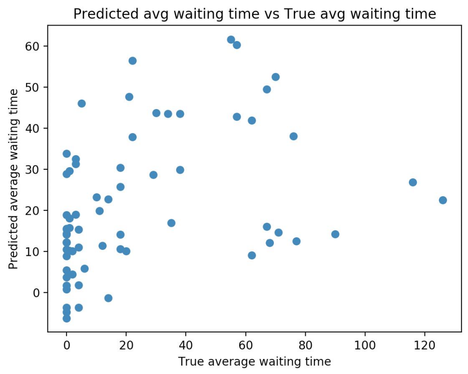
Preliminary data, I/O examples, and evaluation metrics

For the first part of our project (prediction), success will be if we predicted the influx of students and average service time with low error. One input example could be: [CS107, Week 3, 10 AM, 4 days until assignment due] and output would be something like [12 signups, 10 minutes per student]. For the second part of our project (recommendation), success will be if our system can return the optimized schedule correctly under all conditions (assuming part A goes well). One input example would be [TA 1: [(times available), total work hours], TA 2: [(times available), total work hours], array of dates with expected student signups and average serve times], and output would be [array of times to assign each TA to] that minimizes expected wait. For now, we have collected, parsed, and organized data from CS221 and CS107, for Autumn 2018 and Spring 2017.

Baselines and oracles

We experimented on CS107, Spring 2017 Queuestatus and used features such as day of week, number of sign ups, number of serves, average serve time and number of servers to predict average wait time. We randomly shuffled data with 0.9:0.1 train-test split and judged the models based on mean squared error between predicted times and actual wait times. Currently we just predict the average waiting time, which can be extended to other features on in future works in the project.

For baselines, we fitted a linear regression taking 5 features as input with an intercept term (in total, 6 features with $x_0 = 1$). Since the data was randomly shuffled for train/test split, mean squared error varies wildly between 226 and 21064. The average mean squared error of 10,000 different linear regression models gives a mean of 2553 minutes^2 for MSE.



For oracles, we manually estimated randomly selected outputs (with additional features, such as considering the average wait time before and after the hour we are predicting) and take the mean squared error between our predicted scores and true scores, which gives a final MSE of 329 min^2. Even for oracles, we see that the mean squared error prediction is not perfect, since the regression task based on mean squared error operates differently from a classification task. The difference between baseline and oracle can be attributed to the human ability to infer from the input data, using each of the features logically based on prior knowledge of how signups and number of TAs affect waiting time (and patterns from directly before and after each slot to be predicted).

Related work

Predicting CS106 Office Hours Queuing Times - Troccoli, Capoor & Troute:

The authors used custom feature extractors to come up with a model that could predict wait times at the LaIR. They made a few interesting observations regarding prediction tasks. The first was that multimodal classification with equi-depth buckets tended to outperform regression approaches in terms of accuracy, indicating that focusing part 1 as a classification problem might be more fruitful. Another observation was that adding layers to the neural network or nonlinearities did little to improve performance. We are thus also implementing a shallow model, putting emphasis on the feature engineering aspect. In contrast to this project's dataset, QueueStatus eclipses the LaIR framework in terms number of courses served, which allows us to generalize more. One limitation of QueueStatus is it collects less problem-specific information than LaIR does, so we will have to rely on other forms of feature extraction in our model.