CPSC 340 Machine Learning Take-Home Final Exam (Fall 2020)

Skeleton for Question 2 Answer

1 Team

Team Members	*William Choi (x4a9, 36643138)* *Fong Yuan (b0w2b, 23575442)* *Siddarth Mittal (n0q1b)* *Vanessa Bayubaskoro (w3z0b, 45602166)*
Kaggle Team Name	*Lorem Ipsum*

2 Introduction (3 *points*)

A few sentences describing the autonomous driving prediction problem.

In this problem, we are given the position of ego vehicle and their (up to nine) nearest agents spanning 1 second into the past. These agents can be either cars or pedestrians. Our job is to predict where the ego vehicle will be 3s into the future using the past information regarding itself and other agents.

3 Summary (12 *points*)

For this problem we choose to use a linear regression model with L1 regularization (lasso) and hyperparameters lammy=0.01 and maxEvals=3000. We decided to train a model at each time slice, for every 100ms interval, resulting in 30 models. For each model we used a feature vector created from extrapolating the ego vehicle data sets.

In order to make sure the data for each feature vector was consistent, we had to first rearrange the data such that the agent itself was at the front for each ego vehicle. This was because not every agent was located in the same column for its own data set and the agent position at time 0 always has a position of 0,0. After fitting and making predictions for each time slice, we recombine the result into one prediction for each ego vehicle.

For features, we used the position of the ego vehicle, the other agents for each ego vehicle as well as the velocity and acceleration. We found that adding velocity and acceleration to our feature set greatly improved our score.

4 Experiments (15 *points*)

Several paragraphs describing the experiments you ran in the process of developing your Kaggle competition final entry, including how you went about data prepossessing, feature engineering, model, hyper-parameter tuning, evaluation, and so forth.

At the start, we just wanted a basic baseline to work with. So we started with an auto regressive model using only the 22 features associated with each ego vehicle. This didn't work out too well since we had a restricted features set. So the next step was to combine all 220 features across all vehicles associated with the ego vehicle by creating feature vectors of each ego vehicle. Along the way, we also reordered the agent to be at the front of each feature vector to make the dataset consistent between all feature vectors. We found that this model performed similar if not worse to only having features for ego vehicle. Our guess was that auto regression doesn't generalize well with a large feature vector or perhaps there was large correlation between feature vectors. Auto regression also only grabbed the first position in our yfiles and basically needs to guess the rest of the time slices, so it was having to predict alot.

So our next step was to switch our model out with a linear regression model, starting with L1 regularization (Lasso). To address the issue of having to predict so far into the future, we instead decided to fit our model to every time slice of 100ms intervals. That is, we ended up fitting 30 different models for each time slice respectively. This ended up giving us a strong validation error to start with. Since we had a solid baseline model to work with, we then experimented with different models to see if we could get stronger results. Using L2 regularization obtained validation errors that were close, but ultimately still worse than L1 regularization. MLPRegression and SVM both provided much worse validation errors as well. In the end, we chose to stick with L1 linear regression and move on to feature selection and hyperparameter tuning.

Once we had decided on the model, we decided to try and add potentially relevant features for our model to use. With the position of each agent as well as a time slice, we calculated the velocity and acceleration and added it into our feature set. This increased the score of our model significantly. We also tried adding what the type of agent (car or pedestrian) into our feature set, but this did not show much improvement. We believe that perhaps with a large neural network, it might have helped our score, but we did not have the chance to test it.

Hyperparameter choice was fairly straight forward at this point since all we had was a linear regression model. We tested with varying lammys and max iterations and found that large and very small lammys were unfavourable in regards to the actual test error. Very small lammys with large iterations contributed to lower validation error, but actually gave us worse test errors; likely due to overfitting on the validation set. So we settled with lammy = 0.01.

5 Results (5 *points*)

Team Name	Kaggle Score
Lorem Ipsum	*1.94308 (sklearn L1 - 0.43)*

6 Conclusion (5 *points*)

Several paragraphs describing what you learned in attempting to solve this problem, what you might have changed to make the solution more valuable, etc.

We learned that autoregression models with large number of features did not work super well for this problem. This might be due to some multicollinearity between the features. From the several linear models we tried next, we learned that Lasso regression gave us the best accuracy. Although, linear model with L2 regularization gave similar results to Lasso, we chose Lasso due to the fact that it also does feature selection for a big enough parameter value. In the end, we saw that adding some interaction terms like velocity and acceleration improved the Lasso regression model significantly.

If we had more time, we would do some statistical analysis on all of the features to make sure that we do not have features that are multicollinear to each other. Next, we also just want to keep features that are

actually predictive. We can do some feature selection before we train and fit the data to the model. The goal is to keep the model simple (less number of features), while still outputting accurate results. We've already introduced interaction terms like velocity and acceleration, but we might also try some other interaction terms to see if that would make our Lasso Regression model perform better. Side note, using SKLearns implementation gave us extremely good results (0.40923) on the test data on Kaggle, but we could not seem to get our own implementation of L1 regularization to perform as well..

7 Code

Include all the code you have written for the autonomous driving prediction problem.
Python files included in the zip.