Project: Indycar Rank Prediction
Progress Report
10142019

1. Problem Definition

This project aims to predict the ranks of racing cars in a race.

Rank prediction is critical for the teams who will use simulations to pick up the best strategy in order to win. Rank prediction is also important to the fans. Before the race event each year, people will make all kinds of predictions of the race which including the winner(rank1 at the final lap) and rank for each team. It is part of the sports betting/gambling industry.

The problem can be defined as: given the historical and current status of the race $\{X_i; 0<i<t\}$, predicting the status of race in the future Y_{t+d} .

We explore to use a machine learning/deep learning approach to build the prediction model.

1.1. What is rank?

According to the protocols document, we have the following information on rank:

- "Race results are based on laps led and crossing order." (Results protocol definition)
- "Rank based on best time or race position, updated at each timeline during the race" ($O overall results)

We have verified that the following assumptions are correct:

1. Rank is calculated by the elapsed time when the car crosses the start-finish line. The order of the cars for the same lap number is its rank.
2. Indy500 has multiple sections (timelines), therefore, the rank of a car may change during one lap.

Although real-time position/rank is provided through the LED display on all the racing cars, there are no logged into the log file. I can use a lap-based or timeline/section-based rank only. In the following experiments, lap-based rank data are used.

Since rank can be calculated from the elapsed time, the problem of rank prediction equals to the problem of elapsed time or lap time prediction. We define the task as a time series prediction problem. Taking the lap time in each lap as one data point, a time series dataset is extracted from the log file. Given $\{X_i; 0<i<t\}$, to learn a model to predict X_{t+d}
Taking Indy500 as an example, 33 cars competed for 200 laps in 2018 and 18 of them finished. If the cars are treated as separated units, we get 33 time series for this event. Otherwise, if the cars are treated as one unit, we get one time series for this event with each data point x \in R^{33}.

1.2. How to evaluate?

It is difficult to predict the final result of a coming race. According to one sports betting website[1], the odd of the final winner of 2018, Will Power, is the highest one and is around 7/1 before Indy500 2019. It means that if one bets 1$ on Will Power, he can earn 7$ after the race. It also means that the betting company believes the probability of money spend on Will Power will be less than 1/(1+7)=0.125. An underestimated probability of correctly winner prediction will cause financial failure for the company. Roughly, we can regard 12.5% as the state-of-the-art upper bound of this type of prediction.

The problem of the final winner prediction task is it can NOT be evaluated with limited data. In order to draw a conclusion that a model achieves accuracy of 1/8, it should predict one time correctly on at least 8 different races. In practice, more data are needed to get a statistically effective result. To achieve 1 decimal accuracy, 1000 samples of test data are needed at least.

Alternatively, we can adjust the task to predict the rank in shorter future D laps instead of the final. In this way, we have more test data to evaluate.

1.3. Metrics

We use accuracy to evaluate the overall performance of the prediction of a model.
Accuracy = #(rank predicted is correct) / #(samples of ground truth)

Since not all the cars are equally important, top K accuracy can be used,
TopK_Accuracy = #(rank predicted is correct) / #(rank is in top K in ground truth)

From the perspective of a ranking problem evaluation, top K precision measures how accurate to predict the rank of a car will be in top K in the next D laps, as
TopK_Precision = #(rank predicted in top K is true)/ #(top K predictions)

Furthermore, metrics such as NDCG, DCG can be used to distinguish the different importance of the predictions on different ranks.

1.4. Dataset

Indycar provided log data for 14 different race events of the Indycar series 2018, including qualification races and the final.

In order to utilize more data to train and evaluate, we first tried to train a model with a subset of the events to predict the events not seen in the training set. However, it fails.

---

[1] https://www.sportsline.com/insiders/50883891/2019-indianapolis-500-sportsline-has-surprising-picks-and-predictions/

Then, we return back to train and predict for the same event with data of different years. In this case, the only available event is Indy500 which contains two years of logs, 2017 and 2018.

33 cars races in Indy500 for 200 laps. On average the lap time is about 40s. 16 cars finished in 2017 and 18 finished in 2018. Table 1 shows the elapsed time for the top 5 drivers. The difference of the time between the drivers are small, which also indicates that the rank prediction task could be hard.

Table 1. Ealpsed time of top 5 drivers in Indy500 2017 and 2018

Indy500	2017			2018		
rank	car#	elapsed time(s)	diff(s)	car#	elapsed time(s)	diff(s)
1	26	11583.3584		12	10782.6365	
2	3	11583.5595	0.2011	20	10785.7954	3.1589
3	19	11583.8862	0.3267	9	10787.2293	1.4339
4	8	11584.4949	0.6087	27	10787.8602	0.6309
5	10	11585.0056	0.5107	28	10789.3552	1.495

2. Baseline results

A naive model simplely predict the rank of the next D laps as the same rank as current lap, denoted as ***Simple***. Since rank of the cars are stable in most of the time, this navie model works well than expected. But, of course, it is useless. We have to find a model that works better than it.

We tried LSTM, SVM and RF to build prediction models, which are trained on 2017 Indy500 data and used to predict on 2018 Indy500 data. SVM and RF have to use a fixed time sliding widnow to prepare the training data, while LSTM is capable to deal with the whole time series as input. In the following experiments, we denote this LSTM model as ***LSTMTS.***

Table II. Model Evaluation on Indy500 dataset. (rank prediction of next D laps)

D=1	**model**	**accuracy**	**top1_accuracy**	**top5_accuracy**	**top5_precision**
0	lstmts	0.634328	0.785714	0.730612	0.876531
1	simple	0.761747	0.882653	0.823469	0.940816

D=5	**model**	**accuracy**	**top1_accuracy**	**top5_accuracy**	**top5_precision**
0	lstmts	0.372582	0.602041	0.455102	0.718367
1	simple	0.450802	0.69898	0.571429	0.788776

D=10	**model**	**accuracy**	**top1_accuracy**	**top5_accuracy**	**top5_precision**
0	lstmts	0.297402	0.571429	0.42551	0.663265

1	simple	0.296296	0.596939	0.426531	0.684694

As the results in Table II show that the lstmts model learned from data is even not as good as the naive simple model.

2.1. Model analysis

Taking an experiment of lstmts for example.
Training on 2017 data with 16 completed cars, testing on 2018 dataset with 18 completed cars. The other parameters include *"hidden units=16, use_flag_status=True, loss function=mse"*

The input <X,Y> is <laptime, laptime after D laps>. As in Fig.1, the loss curve of the training process shows that the model converges on the training ata, but the performance on the validation dataset is not much impacted by the 'training', especially in the case of large D. It indicates the model does not learn well from historical data in order to predict future for a new event.

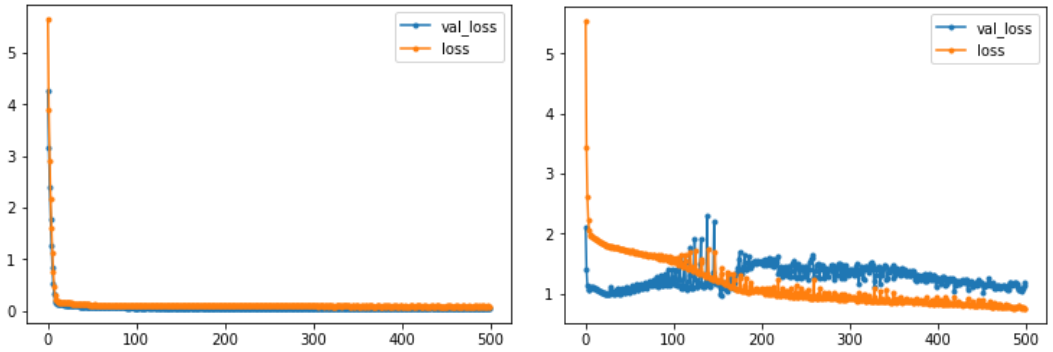

Fig.1 Convergence of loss. (left) D = 1, (right) D=10

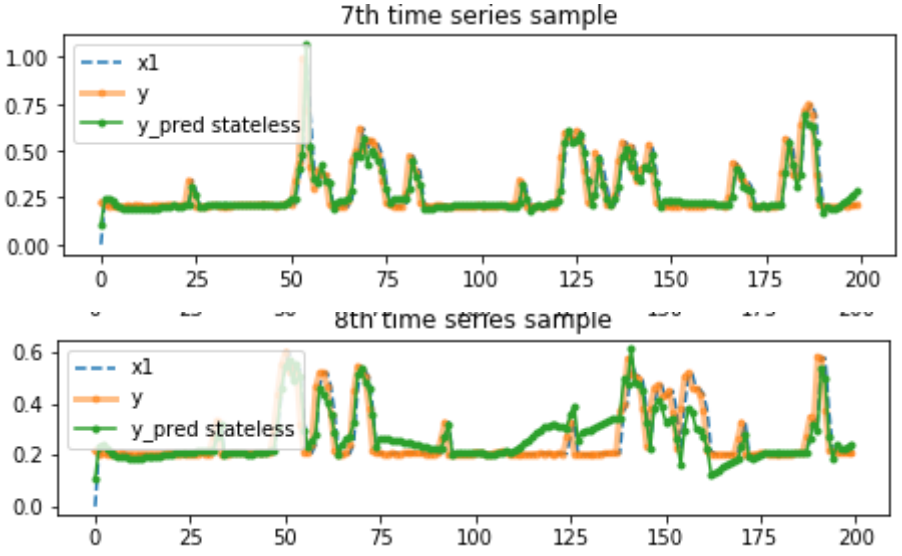

Fig.2 Precition on Training and Testing set wth D=1, use_flag_status=False. (upper) Training, (bottom) Testing

Fig.2 shows the overfitting of a model, which predicts ‘perfectly’ on the training set, but when it fails in predicting on a new event. As in Fig.2 (bottom), predictions near the third pitstop are messy. One possible reasone is that the model remembers there should be a spike after the second pit stop.

2.2. Ideas explored

idea	notes	result
use multiple events data	in order to increase the volume of training data set	not effective
use flag status	adjust the weights for the training samples according to the track_status assign small weights to the laps when yellow flag is raised	no significant difference
model parameter tuning	number of hidden units[16,32,64] loss function[mae, mse]	large model has the problem of overfitting

3. Summary

Experimental results show that the current models are not good enough to exceed the simple baseline. Beside exploring more modeling techiniques, the following issues are important for the work of next step.

1. Sufficient training data are needed in order to train a model which is able to capture the underline patterns within the data. Lap time correlates with the events of pitstop and crashes, and therefore, no obvious patterns are oberved, making the task difficult.
2. Define a task of prediction which we can extract more labeled data from the existing log files.