Apply quantile regression to do average-travel-time prediction

Blackswan Team

June 7, 2017

Contents

1 Introduction

The objective of this competition is to estimate the average travel time from designated intersections to tollgates of every 20-minutes time window.Our final models are quantile regression produced by 3,4-hidden-layers neural network model.
Given the evaluation metric MAPE=$\frac{y-\hat{y}}{y}$ and the classical regression loss function adopted by most machine learning pakage:

- MSE(gaussian distribution) $loss = \frac{1}{2}(y - \hat{y})^2$.
- MAE(laplace(quantile) distribution): $loss = |y - \hat{y}|$.

we tried two different approach in this competition:

Log-transform : if we take a log-transform of target y, we can do a simple mathmatical approximiation:

$$|log\hat{y} - logy| = |log\frac{(\hat{y})}{(y)}| = |log(1 + \frac{\hat{y} - y}{y})| \approx |\frac{\hat{y} - y}{y}|$$

in this case, the measure MSE, MAE become relatively measure close to the spirit of MAPE;So we can try to do log-transform of the target value-model training-do exponential transform.

Quantile-regression : the quantile regression minimize the absolute error between target and prediction $|y-\hat{y}|$, given the true target is unknown to us, a quantile point a little small than 0.5 is also close to the spirit of MAPE;

Finally, we choose **quantile regression** directly which acheive better result both on the validation set and stage1's leadboard.

1.1 Framework of code files

All of task1's code is writeen by R;

- *installPackages.R*, We provide the script to install all the packages we used in the competition;
- *basicPreprocess.R*, Read the file and preprocess the trajectories data and weather data; save the result into the file "basicPreprocess.rda"(R's binary file, like pkl to python);
- *advancedPreprocess.R* and *ATTHelper.R*: advanced data preprocessing include "data augmentation", "outlier preprocessing" and feature engineering; More details will be provided in the second part.
- *dataImputation.R* and *ATTHelper.R*: there is a lot of missing data in the Task1, so we adopt an imputation approach base on the random forest model. More details will be provided in the second part.
- *extendBaseline.R* and *DLHelper.R*: **the core model**:feed-forward artificial neural network; We apply quantile regression with 3,4 hidden-layers neural network as our final model. We will provide more details in the third part.
- *BaselineB*1.*R*, *BaselineC*1.*R*, *BaselineC*3.*R*: for B1,C1,C3 this 3 intersectionTollgate pair with high missing rate, we training with single intersectionTollgate, then do bagging with main result.

2 Data preprocessing and Feature engine

Our data preprocessing mainly include 4 part: "outlier preprocessing", "data augmentation", "feature engineering" and " missing data imputation".

2.1 Outlier preprocessing

There are some travel_time data point which is extraordinary large. Our first attempt is simply calculate the 99.95 quantile point α_i of every intersection-tollgate pair, and remove those points large than α_i.This apporach did not acheive better result in stage1's leadbord. Finally, we just pick an threshold of 450 and manually modify some special day's threshold and remove those points

large than the threshold.
The code is illustrated in line17-25 of file "*advancedPreprocess.R*".

2.2 Data augmentation

One challenge of the competition is "small dataset". A simple and useful approach is to move time window $w(t)$ to $w(t-\pi)$, moving time window will allow different vehicle data in the 20-minute time window and produce more different and reliable data.Finally we choose the data set that is (original, original-5, original+5, original-3, original+3), which is about 5 times large than the original dataset.

	morning	afternoon
orignal	6:00-7:40–8:00-9:40	15:00-16:40– 17:00-18:40
+5	6:05-7:45–8:05-9:45	15:05-16:45– 17:05-18:45
-5	5:55-7:35–7:55-9:35	14:55-16:35– 16:555-18:35

Table 1: time-window moving

The data augmentation function: *extendMLDataFunc* in file "*ATTHelper.R*";
augmentation preprocess: line60-99 of file "*advancedPreprocess.R*".

2.3 Feature engineering

Here is the final feature list table:

feature	description	code
intersectionTollgate	one-hot encoding	line31 of file "*ATTHelper.R*"
hourWindow	morning or afternoon	line16 of file "*ATTHelper.R*"
MWInterval	1:6 numeric	line31 of file "*ATTHelper.R*"
intersectionTollgate	one-hot encoding	line31 of file "*ATTHelper.R*"
ATTLag1-6	6 average-travel-time lag points	line64-69 in file "*ATTHelper.R*"
ATT120SdLag	trave_time Sd of last 2 hours	line42 in file "*ATTHelper.R*"
ATT60MeanLag1	mean trave_time of last 1 hours	line45 in file "*ATTHelper.R*"
ATT60NumLag1	vehicle number of last 1 hours	line47 in file "*ATTHelper.R*"
isHoliday	Including 9.31 afternoon	line26-27 in file "*dataImputation.R*"
weather	precipitation(0-1 encode),rel_humidity	line17-24 in file "*dataImputation.R*"

Table 2: feature and code

2.4 Missing data imputation

Another challenge of this task is highly missing rate of data points. Finally we apply the methodology of proximity imputation which inherits from randomforest[1].

The basic idea of proximity imputation is to do a quick replacement of missing data and then iteratively improve the missing imputation using proximity matrix. The proximities originally formed a NxN matrix. After a tree is grown, put all of the data, both training and oob, down the tree. If cases k and n are in the same terminal node increase their proximity by one. At the end, normalize the proximities by dividing by the number of trees.
In our application, we train with the formula and set tree number=50
$ATT = intersectionTollgate + MWInterval + hourWindow + ATTLag1 + ATTLag2 + ATTLag3 + ATTLag4 + ATTLag5 + ATTLag6 + isHoliday$
The data imputation function: *ATTImputationFunc* for original data and *extendImputationFunc* for extend data in file "*ATTHelper.R*"; imputation preprocess: file "*dataImputation.R*".

3 NN Model Training and Parameter Turning

Our final submit involves 4 NN models which include:

- NN model with all intersection-tollgate dataset(extendBaseline.R).
- NN model with data "intersection-tollgate==B1"(baselineB1.R).
- NN model with data "intersection-tollgate==C1"(baselineC1.R).
- NN model with data "intersection-tollgate==C3"(baselineC3.R).

We analysis the missing-rate of 6 tollagate-direction pairs. A bagging of two model result may reduce variance

	intersectionTollgate	missATTLag1	missATTLag2	missATTLag3	missATTLag4
1:	A2	8	8	18	6
2:	A3	31	25	17	22
3:	B1	223	206	119	97
4:	B3	61	66	90	133
5:	C1	143	200	196	182
6:	C3	267	211	190	338

, which also acheive better score both in validation the stage1's leadboard. After some testing, we finally decide to use H2O.AI's open-source product h2o.deeplearning[2]. Our basic training framework include

- Take the last week as validation set, others as training set(line13-21 in extendBaseline.R).
- Training NN with adadelta[4];Momentum training can acheive better single model result, as we train the model with CPU services, we find in the same training time(about 3 4 hours with 32 CPU cores), adadelta always acheive better result than momentum training.(line67-79 in extendBaseline.R)
- combined randomized strategy and early-stopping to do grid search(line56-79 in extendBaseline.R).
- Do evaluation and Bagging of best K manually selected models(function adaGridEvalFunc in *DLHelper.R*, line81-95 in extendBaseline.R).

parameters	list
activation	"Rectifier","Tanh"
distirbution	lapalce,quantile regression with quantile_alpha=0.5
l1,l2 regularization	c(0, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7)
input_dropout_opt	c(0,0.05,0.1,0.2)
max_w2_opt	10,20,30,40,50
adadelta-epsilon	c(1e-4,1e-6,1e-8,1e-10)
adadelta-rho	c(0.9,0.95,0.99,0.999)

Table 3: grid search parameters list

3.1 NN Manually Turning Guide

The h2o.deeplearning model Turning is a little tricky and involve serveral manual turning. The section is to help others to reproduce similar even better score in leardboard, but not reproducible given the offical document instruction[3].Everytime the top K best models on validation set may also be different with different neuron framework.

- `seed`: Specifies the random seed controls sampling and initialization. Reproducible results are only expected with single-threaded operations (i.e. running on one node, turning off load balancing, and providing a small dataset that fits in one chunk). In general, the multi-threaded asynchronous updates to the model parameters will result in intentional race conditions and non-reproducible results. The default is a randomly generated number.

1. First I usually run a simple 3-layers model with varible importance as a "warm-up" and observe the variable importance of features(line37-51 in file "*extendBaseline.R*").
2. Speficy the proper total grid search running time:*max_runtime_secs*(line55 in file "*extendBaseline.R*") and single model max running time *max_runtime_secs* = 300(line77 in file "*extendBaseline.R*"), it usaully depend on the computing power,training data size, candidate cartesian parameter list. Tt takes about 3 4 hours to finish grid search in my 32 cores CPU server. It will takes less time with deepwater with GPU servers(https://github.com/h2oai/deepwater). It is very important to finish the randomized grid search with sufficint time and computing resource.
3. Apply function *adaGridEvalFunc* to illustrate all model parameters and validation result(line89-90 in file "*extendBaseline.R*").
4. Select top K(K=15) models, observer the validation result of bagging topK(K=2:15), manually remove single model that generate worse baggining validation result, through function *DLSummaryFunc*, than select

best J models through manual pick-up(line93-95 in file "*extendBaseline.R*", DLSummaryFunc in ATTHelper.R).

References

[1] Breiman's RF page, R randomForest package, R mice package

[2] DeepLearning_Vignette.pdf, h2o.ai offical website

[3] DeepLearning_Vignette: Appendix A: Complete parameter list:seed,

[4] MD Zeiler, *Adadelta: An adaptive learning rate method*, 2012.