KDD CUP 2017 Travel Time Prediction
Predicting Travel Time – The Winning Solution of KDD CUP 2017

Team: Convolution
Task Understanding

Problem Definition

Task

- To estimate the average travel time from designated intersections to tollgates

Metrics

\[
MAPE = \frac{1}{R} \sum_{r=1}^{R} \left( \frac{1}{T} \sum_{t=1}^{T} \left| \frac{d_{rt} - p_{rt}}{d_{rt}} \right| \right)
\]

Data

- Vehicle trajectories along routes
- Weather data in the target area
- Road topology
• Data is **noisy and sparse**, easy to overfit to specific validation data
  • Hard to conduct offline experiments

• **Small abnormal values** have big impact on final metrics, especially considering MAPE evaluation

• Time sequence is hard to predict, especially the following pattern of the next six windows
More Effective Validation Methods

• Moving window based CV split
  - Use Last 2-8 days as validation, last 9-98 days as training
  - Use Last 3-9 days as validation, last 10-98 days as training

• Interval K-fold based CV split (know the impact of last week data for train)
  - Use second week as validation, other weeks as training
  - Use third week as validation, other weeks as training

• Use online leaderboard feedback to determine the weight of validation sets
  - totally 13 CV sets used, much more consistent with online feedback
- Offline training set
- Offline predict result
- Offline score

- Online training set
- Online predict result
- Online score

- Feature Extraction
- Model Tuning
- Ensemble

Workflow:
- Model
  - feedback
  - optimize
  - many times a day
  - One time a day
Features

Data Augmentation

Timeline

2 hours

2 hours

2 hours

2 hours

2 hours

2 hours

6 samples

Feature extraction \(\times\) targets

250k training samples in total
• Remove holiday days’ traffic

• CV based denoise
Aggregate information from 2 dimensions

- Basic Features
  - Time
  - Road

- Session Level Features
  - Last K cars statistics
  - Moving window statistics

- Long Term Features
  - Use exponential decay factor to conduct statistics
• More than 100 features are extracted

• Feature selection based on CV experiments and online feedback

• Multiple feature combinations are used for ensemble, solving feature conflict

Features

Feature Engineering

Full feature set

model1
Feature set 1

model2
Feature set 2

model3
Feature set 3
**Models**

- **XGBOOST**
  - Level-wise growth strategy
  - Stable

- **LightGBM**
  - Leaf-wise growth strategy
  - Good algo for category feature
  - Very Fast

**Model Tuning**

- Every fold has a stop round, using CV fold weight to determine the full data stop round

**Sample Weight**

- Use 1/label as sample weight, make the little value samples learnt better
**Neural Network**

**One Hot Embedding**
- Using embedding layer for basic feature
- Similar location’s embedding vector is close

**Early Interaction vs Later Interaction**
- The balance of learning between embedding part and statistics part

**MLP vs RNN**
- Multiple Layer Perception: Powerful Expression Ability, Learn feature interaction
- Recurrent Neural Network: Good to model sequence relationship, but unstable for generalization
• N-fold Validation model ensemble without retrain
• Global model and separate district model
• Use 2, 10, 20 minutes as moving window to construct samples

Data Level Ensemble

Feature Level Ensemble

• Different Exponential decay factor or smooth factor for statistics features
• General route feature and link feature bagging
• Long term statistics and short term statistics ensemble
Models

Ensemble

Model Level Ensemble

- Different ml models, including xgb, lightgbm, dnn, rnn
- Loss function changes, weighted norm-2 and fair loss to approximate norm-1 loss
- Label transform, like customized log transform, and the sample weight transform
- Model parameter, like nn structure, gbdt parameters

Result

- Choose 13 base models base on multiple weighted cv and leaderboard feedback
- Finally, we got 0.1748 mape in leaderboard
## Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Basic Feature, Session Level Feature, Long Term Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>XGBoost, LightGBM, Multiple Layer Perception</td>
</tr>
<tr>
<td>Ensemble</td>
<td>Weighted CV based, Log transform, Loss function etc.</td>
</tr>
</tbody>
</table>
Summary

- Pay more attention to data distribution, like data changes, noisy data
- Build scientific cross validation sets is very significant, and rely on CV much more than leaderboard
- Treating error composed of bias, variance and noise, try to decrease variance if bias is hard to decrease
- Think more and fine tuning less, or it will be easy to overfit to small data
- Understand the evaluation well, and design corresponding loss function
- Separate modeling is useful if data distribution differs a lot
Thanks

Q&A