Report for FishingForFishermen

Your Information

Firat / Last Name:	Naoya Takahashi
TopCoder handle:	fugusuki
Email address:	g440556d@gmail.com

Approach Used

Overview

My approach was combining k-Nearest Neighbor and Gradient Boosting Decision Trees (XGBoost library was used for GBDT). Because decision tree method is not good at using 2-dimensional complex position data, latitude and longitude itself were not used as input features of GBDT prediction but results of kNN prediction were used as features instead. The way of feature engineering, parameter tuning and model ensembling were not special. AIS static report messages were not used.

Entire parts are listed below (each step is corresponding to each line of run function [line 340] on the source code).

1. Raw messages were decoded to extract features and applied some feature engineering.
2. Fishing probability of each sample was predicted by GBDT without detailed positional features (i.e. latitude and longitude).
3. Fishing probability of each sample was predicted by kNN using only detailed positional features. Samples whose probability of fishing were low on part 2 were ignored because those samples might be unrelated to the boundary of fishing / not fishing.
4. kNN results were concatenated to the input data.
5. Fishing probability of each sample was predicted by GBDT using input data created by part 4.
6. Final submission file was created by averaging some result of part 5.

Detailed description of my algorithm (Approach ultimately chosen / Advantages and disadvantages of the approach chosen / Special guidance)

Feature engineering (Part 1 in Overview section; prepare function [line:10])

At first, raw messages were decoded with libais library. Continuous features were used as it is and categorical features were converted to be dummy features. Samples were separated in some clusters and region of the position were important feature. The region of the sample was appended as feature. Country from which the ship come could be obtained from MID (first 3 digits of MMSI) and were also important feature and appended. MMSI and MID themselves were dropped in order to avoid overfitting.

In addition, the samples were grouped by MMSI and aggregated to calculate some statistic values (e.g. sum, min, max, mean, max - min) of each feature. I did not test whether these handling were effective for this competition, but these might reveal the ship's characteristics and append ship's information to each signal sample.

Prediction by GBDT model (Part 2, 5; predict [line:247], create_model_xgb [line:235])

GBDT method and XGBoost library were used because they can fit the data well and are widely used for data mining problem. Input data was result of part 1 in part 2 and result of part 4 in part5. x (longitude), y (latitude), TIMESTAMP and aggregated values (min, max, mean) of them were ignored in both part. I think this handling had good effect for avoiding overfitting though provisional score did not significantly differ.

Probability of fishing were predicted for both training set and testing set using n-fold cross validation. Trainings were executed in multiple set of hyper-parameters (including tree count and sample weight) and all results were used in next part. I think sample weight variation was important because it can control importance balance of sample feature / ship feature.

Prediction by kNN model (Part 3; calc_knn [line:226], calc_knn_proc [line:172])

Some values between 10 and 300 were used as k. Mean of label and distance of farthest sample were calculated for each k. All results for different k were used in next part. Samples whose probability of fishing were low on part 2 were ignored because those samples might be unrelated to the boundary of fishing / not fishing.

kNN result concatenation (Part 4; concat_data [line:150])

All results themselves were concatenated. Samples were grouped by MMSI and several percentiles of all kNN results for each MMSI were appended. This may help characterize each ship.

Create final submission (Part 6; submit [line:311])

Final prediction was average of results manually selected considering cross-validation and provisional score.

What I tried other than that above (Approaches considered / Steps to approach ultimately chosen / Potential improvements)

I made many submissions in this competition because the testing set contains many samples in region where training data did not present and score for those samples could not be obtained by cross validation. Most of my submissions were for tuning hyper parameters of GBDT and selecting final submission.

I tried stacking (i.e. concatenating part 5 result to input data and again make GBDT prediction) but did not worked.

I did not try to find how to use AIS static report messages because it seemed to be too complicated but it might improve the prediction.

I only used GBDT only for final prediction. Combine other methods such as Neural Network may slightly improve the prediction.

Installation and usage (Open source resources / libraries)

Programing language

Python 3.5.2

Libraries

numpy 1.11.0 (http://www.numpy.org; 3-clause BSD License)
pandas 0.18.0 (http://pandas.pydata.org; 3-clause BSD License)
xgboost 0.4a30 (https://github.com/dmlc/xgboost; Apache 2.0 License)
libais 0.16 (https://pypi.python.org/pypi/libais; Apache 2.0 License)
scikit-learn 0.17.1 (http://scikit-learn.org/stable/; BSD License)

Commands

```
# After installation of Python 3 completed,
# run the following command to setup libraries.
pip install numpy==1.11.0 pandas==0.18.0 scikit-learn==0.17.1 xgboost==0.4a30 libais==0.16
# Change working directory to the directory that contains program.py.
# Put training and testing files in the ./input directory. Static report file is not needed.
# Then run the program by following command.
python program.py
# submit.csv that contains final result will be created at working directory.
```

About Dockerfile

My Dockerfile setup all libraries and files needed to run. Files are put in /work directory and working directory are set to /work. Please replace files in /work/input if you want to replace training or testing file.

To use the Dockerfile, please build a container from the Dockerfile or pull prebuilt image from fugusuki/fishing1 repository of the Docker Hub. The container run by following command.

```
docker run –it fugusuki/fishing1
```

In the container, simply following command can run the program.

```
python program.py
```

About memory usage

Please note that my algorithm uses about 2GB * knn_nthread [line: 171] (i.e. 40GB by default) of RAM. To reduce RAM usage, please set lower knn_nthread value.