Fraunhofer

IoT data analysis model

1. Introduction

In the logistic industry, IoT applications are considered as cheap solutions for variety of use cases, and could be easily combined with machine learning and artificial intelligent in order to develop smart maintenance and supervision models for the industrial assets. Especially, in the contact with cooperation partners in the logistic industry, we found that the behavior data of forklifts is full of value, which not only indicates directly the working status of a forklift (e.g. loading, unloading, normal driving, harmful vibration etc.), but is also possible to be further used for analyzing high level KPIs. For instance, the analyzing of forklift workload is valuable to arrange assets and forecast demand, as well as for predictive maintenance.

We develop an IoT solution in combination with machine learning algorithm for the logistic industry, aiming at analyzing daily behaviors of the forklifts. Those classified behavior data can be use by other IoT applications for further analysis, for example analysis functions of RIOTANA (Real-time IoT Analytics) by Fraunhofer ISST. We collect real motion sensor data from forklifts and develop machine learning algorithms to recognize the behaviors.

2. Dataset

2.1 Data Collection

The sensor data is collected in the real field test from a forklift at Fraunhofer IML. Here we use a Linde H25/500 diesel forklift. Two sensor sets are equipped to the forklift, one on the fork, and the other aside the driver seat. Each sensor set consists of a microcontroller (here ESP8266), an accelerator and a gyroscope. The dataset has 3-dimensional acceleration, 3-dimensional gyroscope as well as a timestamp provided by microcontroller. We receive 10 -15 records from each sensor sets per second, and the data will be further processed. The forklift ran around in garage for one hour with doing different daily behaviors. For example, the forklift running straightly, turning left, right, simulating a harmful vibration by running quickly over a groove, loading and unloading cargo. The whole process was filmed manually, and we labeled the dataset with different behaviors according to the video.

2.2 Data preprocessing

We down sample the raw data to three records per second by using the average value, so that each sensor dataset is down sampled to around 5000 records. In addition, we labeled the dataset manually with the five behaviors “Standing”, “Driving”, “Loading”, “Unloading” and “Shock”.

Label	Abbreviation
Standing	st
Driving	dr
Loading	lo
Unloading	ul
Shock	sh
Unknown	unknown

Research of the dataset show that the behaviors last from one second to more seconds, for example, a "Shock" can take one second, while "Loading" and "Unloading" usually take around 5 seconds. We apply sliding window to the time series data. The hyper-parameters of sliding window, window length and step size, are determined by a grid search of parameters in the analysis model. The experimented range and optimal value of parameters are shown in the following table.

	Range	Optimal
Window Length	[1, 10] second(s)	3 seconds
Step Size (Overlap)	10% - 90%	50%

The dataset is divided averagely according to the class distribution into training (80%) and testing sets (20%).

3. Data analysis

We developed two kinds of machine learning models for the behavior analysis, 1. Dynamic Time Warping (DTW) + K-Nearest-Neighbors (KNN). 2. Classical machine learning models (Multi-layer Perceptron, Support Vector Machine, Random Forest and Decision Tree). The main difference between the two kinds is the distance metric. In order to capture the temporal information from time series data, the distance metric is of vital important. DTW is a widely used metric to determine the similarity of two time series sequences. The DTW is used in a KNN model to determine the distance between neighbors. In the meanwhile, other classical classification models are trained and tested with expanded sliding window data. We applied feature expansion for the sliding windows, and used the statistical features of windows to capture the temporal information. In concrete, for each dimension of the raw data, we expand the feature in to a feature space with eight features (totally 48 features), which are mean, standard deviation, variance, skewness, kurtosis, maximum, minimum, and median. In order to focus on an essential important feature space, we reduce the dimensionality with

Principle Component analysis (PCA). The size of target feature space is also determined by grid search.

In the evaluation with subset of sensor dataset, the MLP model with three layers, each layer 100 neurons achieves 81% accuracy by predicting forklift behaviors, while the KNN+DTW with K=5 achieves 79% accuracy. Moreover, the classified forklift behaviors are then further used in RIOTANA Platform for KPIs and predictive maintenance.

	KNN+ DTW (K=1)	**KNN+ DTW (K=5)**	**KNN+ DTW (K=9)**	**DECISION TREE**	**SVM**	**MLP**
Accuracy	0.69	0.79	0.78	0.76	0.68	**0.81**

4. Usage

The tool is developed in python. In order to run the scripts, it is necessary to install all required packages with

pip install –r requirements.txt

In order to use the tool for IoT data analysis, the dataset should be prepared in advance. The data file should be in .csv format, with 8 columns:

<acc_x,acc_y,acc_z,gyro_x,gyro_y,gyro_z,timestamp,label>

which are 3-dimentional acceleration, 3-dimentional gyroscope, the sampling time as well as the forklift behavior category as label.

The input data path can either be the path to a single file, or a directory containing a couple of csv files, which are data collected in parallel from different sensors of a fork-lift.

There are a series of optional parameters of the model,

Parameter	Default value	description
recordsPerSecond	3	Resampling let every second contains the same amount of records
windowLength	10	Number of records contained in a sliding window
overlap	0.5	Sliding window overlap rate
testSize	0.2	Rate of data used for testing

Seed	42	Random seed for train- and test-set split
nComponents	10	Number of components after PCA
kFold	5	Cross validation folds
voting	True	Whether make a voting of the models

The tool can be start with

python main.py [OPTIONS] INPUT_PATH