Robot Floor Surface Recognition

Ran Yi
ryi1@stevens.edu
Stevens Institution of Technology
Hoboken, New Jersy

Daoyuan Chen
dchen2@stevens.edu
Stevens Institution of Technology
Hoboken, New Jersy

ABSTRACT

The goal of this project is trying to find the best way to do preprocessing and develop models using machine learning algorithms for robots to recognize the type of floor they are driving on. All the data used to train the models is collected from 10 inertial measurement units mounted on the robot. The report will provide a way to do the analysis of the data preprocessing. Using Exploratory data analysis (EDA) to find the data structure of the data and also do the data preprocessing to improve the model performance. And also compare the performance of different models in predicting the label.

KEYWORDS

EDA, Machine Learning, Data preprocessing

1 INTRODUCTION

Nowadays have witnessed the increasing appearances of robots in our daily life. A sweeping robot will free you from hours of housework and ensure you a clean floor every-time you come back from work. A drone can help you take pictures from unreachable angles. If you live far from your workplace and have to spend hours each day driving through heavy traffic, won't you dream of a reliable self-driving car which will free you from staying highly focused behind the wheel? In the industry, robots have significantly improved the efficiency, quantity and quality of production while driving the cost of production down. Without robots in the production line of automobile factories, many wouldn't have afforded a car nowadays. One-day delivery wouldn't have become an option if there were no robots working in the warehouses. Robots have become irreplaceable roles in some fields like military operations and scientific researches, where robots can perform a lot of tasks which are considered dangerous or impossible for our human beings.

In the near future, we are expecting to expand the application of robots to a wider area. Thus, it's essential for robots to quickly adapt to their surrounding environment. One of the environments robots need to learn is the type of floor they are traversing on. Why is floor type recognition important? Many people may ask such question since they think learning the positions of nearby obstacles is enough for an autonomous navigation system. Wrong. They totally underestimate the complexity of robot navigation. Here is an example, imaging a robot is carrying a glass of water from your living room with a wooden floor to your bedroom with a carpet floor. Since you don't want to splash the water out of the glass, it's better for robot to drive much slower on rough surface such as carpet than on smooth surface such as wood. Obviously, figuring out the type of floor helps the robot apply different navigation strategies. A proper navigation strategy can avoid damage to robot, extend the lifespan of robot and improve work efficiency. In this project, we will have a detailed discussion on how to help robots recognize different floor types.

2 BACKGROUND

One method to classify the floor type is to analyze the images captured by the cameras on the robot. Certain floor type has certain range of colors and patterns. For example, most wood floors have yellow color and thin-strip with annual ring pattern. Most tile floors have a square pattern. A carpet floor shows an uneven color distribution and a granular pattern. A concrete floor is usually grey.

Image based floor classification has been proven quite reliable under specific situation[1].

However, this method will not work on floors with decorations or in a very dark environment. A better way to classify the floor type is to analyze the movement of the robot while driving on different surfaces under the same power. Each floor has its unique coefficient of friction and roughness. For example, If we drive the same robot on a carpet floor and on a wood floor, we will obtain a movement curve with a lower linear acceleration and a larger vibration on vertical axis on carpet than on floor. Many more factors like angle speed and angle acceleration will also be taken into account during the classification task. The sensor to get all the features is called inertial measurement unit (IMU), a combination of accelerometers and gyroscopes, sometimes also magnetometers. In a navigation system, the data reported by the IMU is fed into a processor which calculates altitude, velocity and position. Then IMU integrates angular rate from the gyroscope to calculate angular position. This is fused with the gravity vector measured by the accelerometers in a Kalman filter to estimate attitude. The attitude estimate is used to transform acceleration measurements into an inertial reference frame (hence the term inertial navigation) where they are integrated once to get linear velocity, and twice to get linear position[2].

3 APPROACH

In our project, we will be training multiple machine learning algorithms on a dataset. Our objective is to successfully predict the floor type in most cases by our machine learning models based on the input of those sensors. Since this is a multi-class classification problem, We will try to build models by multiple machine learning methods which are suitable for multi-class classification like Decision Tree, Support Vector Machine, Random Forest, AdaBoost, Bagging, Extra Trees and CNN. Then apply k-fold cross validation to assess model performance by evaluating the accuracy scores on both training and validation dataset.

4 DESCRIPTION OF THE DATSET

The dataset that our project use is collected by Heikki Huttunen and Francesco Lomio from the Department of Signal Processing and Damoon Mohamadi, Kaan Celikbilek, Pedram Ghazi and Reza Ghabcheloo from the Department of Automation and Mechanical Engineering both from Tampere University, Finland. In this dataset, there are 3810 test series, in each series robot drives on 1 of the 9 types of floor (Carpet, Concrete, Fine_concrete, Hard_tiles, Hard_tiles_large_space, Soft_pvc, Soft_tiles, Tiled, Wood). There are 128 measurements per test series, 10 features (orientation_X, orientation_Y, orientation_Z, orientation_W, angular_velocity_X, angular_velocity_Y, angular_velocity_Z, linear_acceleration_X, linear_acceleration_Y, linear_acceleration_Z) are collected from Inertial Measurement Units (IMU sensors) mounted on robot when it was driving on one of the nine floor types in each measurement.

5 EDA

5.1 target value distribution

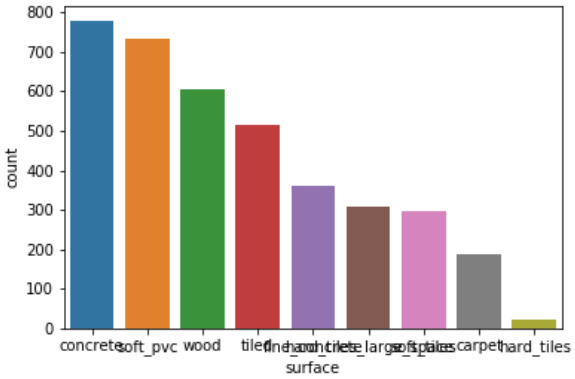

Figure 1: Use the bar-chart to plot the distribution of the target value

From this graph (Figure-1), we can know the target value is not that balance,many machine learning algorithms are designed to maximize overall accuracy by default, sometimes it will completely ignore the minority class in favor of the majority class. So we need to imply resample to our dataset as the professor mentioned. Or we can just use tree based algorithm, this can prevent the imbalance dataset's effect.

5.2 distribution of the data

From the result we got (Figure 2), most of the data are normally distributed, like linear acceleration and angular velocity, this is consistent with common sense, all these variables cannot be too large or too small, and these columns can use directly, because of the normal distribution. However, only the linear_acceleration_z is not centered on 0, which is negative value for the center. For the direction value, they don't follow the normal distribution. orientation X,Y are distributed in range (-1,1) but the orientation Z,W are distributed in range(1.5,-1.5). Usually, we should remove some

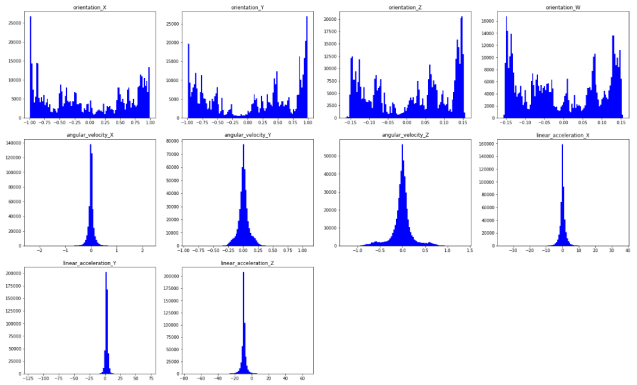

Figure 2: use one series data to see the distribution of the raw data

outliers from the data, and this will improve the normally distribution of the data, but in our case, we cannot do it, because the shape of the distribution is not that kind of close to bell-shape. X,Y,Z,W orientation data are not symmetrical or bell shaped distributed.

5.3 time series plot

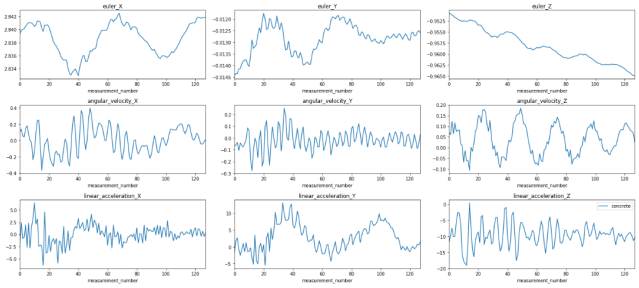

Figure 3: use one series data to see the change of the data in a time series

From the plot above(Figure-3), we can see some value range a lot, but others may stay stable. so we decide to get the maximal, minimal, median, mean, standard deviation, quantile and also the ratio between maximal and minimal. this is the standard label for the time series analysis, we just use them to preprocessing. As I mentioned in the presentation, the focus of this project is not about time series, we just want to use the time series analysis to help us do the preprocessing part.

5.4 correlation matrix

From the correlation matrix (Figure-4), we can summarize a large amount of data where the goal is to see patterns, and the pattern is some varibales are highly correlated, but nost of them can use directly, because they are not that highly correlated. This means our data is ready to use, and the check of the correlation matrix, is the basic step before putting the data into model. We do not need to remove some columns because the correlation between them is not that high level.

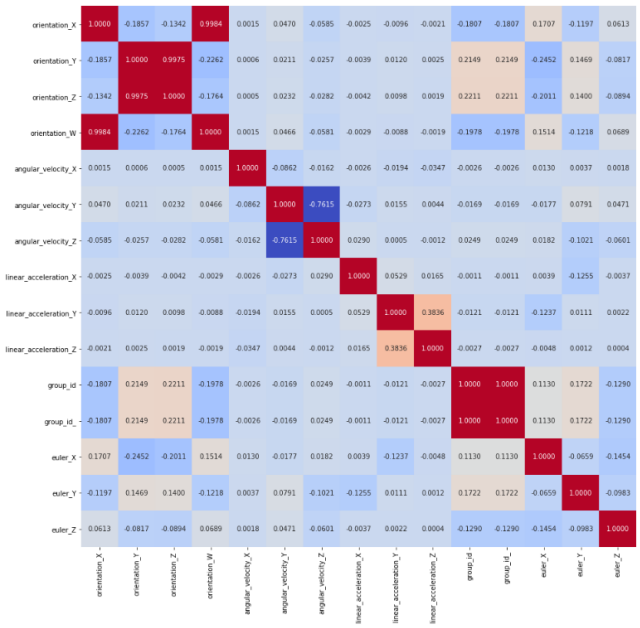

Figure 4: correlation matrix with number after the initial preprocessing (change the axis to Euler axis)

5.5 boxplot

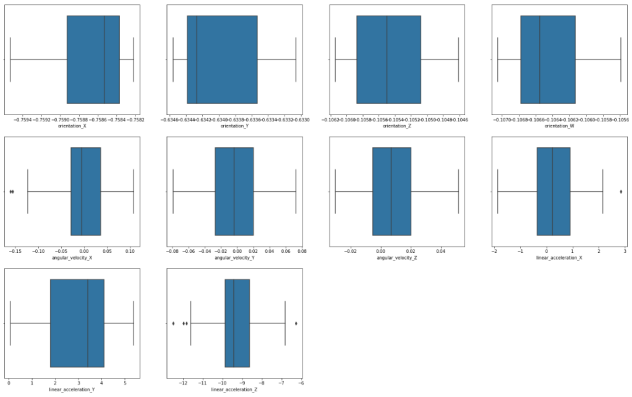

Figure 5: boxplot for every columns, usually, data scientist use this to find the outliers of the data.

From the boxplot (Figure-5), we can see there is some outliers in our dataset, but not too much, most of them occurs in orientation columns, which have showed in dsitribution plot. This boxplot confirm the way we do the preprocessing part, which have been mentioned before. For example, in most columns, there are not very much outliers, but in two columns,linear_acceleration_Z and linear_acceleration_X, there are some obvious points stand outside, we call that outliers. In previous slides, we have learned the effect of the outliers, so we should remove them. But in this project, the outliers tell us, we should take the range of the data into the model. The reason why we should not remove them is because all these outliers points are not seperated from each other, they are connected by the time series and a shared label.

6 PREPROCESSING OF THE DATASET

6.1 Quaternion to Euler Angles Conversion

This is the common part of all model, we used the Accel, Gyro features as it is. Using raw orientation features did not make sense to me, so we used the first order difference of the orientation features (after converting them to Euler angles), so that any vibrations caused by various surfaces are captured.

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_0 q_1 + q_2 q_3)}{1-2(q_1^2+q_2^2)} \\ \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \arctan \frac{2(q_0 q_3 + q_1 q_2)}{1-2(q_2^2+q_3^2)} \end{bmatrix}$$

Figure 6: Orientation$_X$, $Orientation_Y$, $Orientation_Z$, $Orientation_W$ => $Pitch$, $Roll$, Yaw

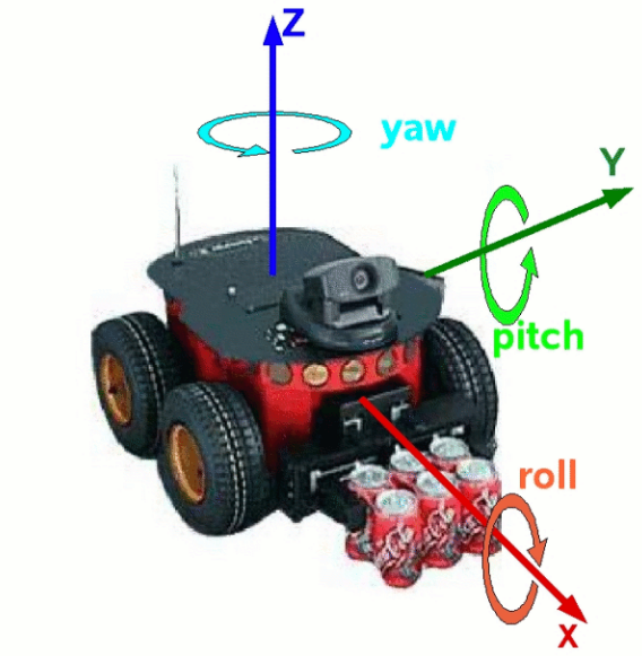

Figure 7: a clear presentation of the pitch, roll, yaw

6.2 preprocessing of CNN

After the first step of preprocessing, we reshape the feature dataset to a nested numpy array of a size (3810, 128, 9, 1), but keep all the information , not removing any values in that dataset. Then we normalize feature values to ensure inputs are zero-mean and

unit standard deviation. Meanwhile, we perform a label encode to our target labels since targets we are going to fit into the model cannot be strings. The corresponding integers to the targets are as follow: 0: 'carpet' 1: 'concrete' 2: 'fine_concrete' 3: 'hard_tiles' 4: 'hard_tiles_large_space' 5: 'soft_pvc' 6: 'soft_tiles' 7: 'tiled' 8: 'wood'

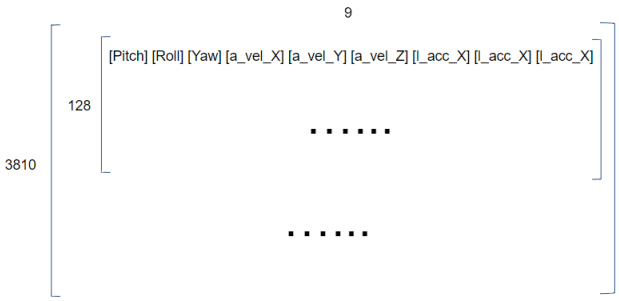

Figure 8: A nested numpy array to be fitted into 2D Convolutional Neural Network

6.3 preprocessing of other models

We add the following feature into the dataset, such as total_angular_velocity, total_linear_acceleration, the ratio between total_angular_velocity and total_linear_acceleration, total_angle, the ratio between total_angle and total_angular_velocity, and the ratio between total_angle and total_linear_acceleration to each series of the data, and find out the mean, minimal value, maximal value, standard deviation, the ration between minimal and maximal value, mean of the absolute value of change of the value, absolute value of the minimal value and absolute value of the maximal value. explain the new feature in equation: total_angular_velocity =

$$\sqrt{angular\_velocity\_X^2 + angular\_velocity\_Y^2 + angular\_velocity\_Z^2}$$

total_linear_acceleration =

$$\sqrt{(linear\_acceleration\_X^2 + linear\_acceleration\_Y^2 + linear\_accelerati}$$

the ratio between total_angular_velocity and total_linear_acceleration

$$= \frac{total\_linear\_acceleration}{total\_angular\_velocity}$$

total_angle

$$= \sqrt{{euler_x}^2 + {euler_y}^2 + {euler_z}^2}$$

the ratio between total_angle and total_angular_velocity

$$= \frac{total\_angle}{total\_angular\_velocity}$$

the ratio between total_angle and total_linear_acceleration

$$= \frac{total\_angle}{total\_linear\_velocity}$$

After all above preprocessing, we use this data to train model like Decision Tree, Support Vector Machine, Random Forest, AdaBoost, Bagging, Extra Trees. We believe after the preprocessing, the performace can be better.

7 EXPERIMENTAL DESIGN

Because our data-set is different from the common data-set, there are 128*9 columns in one series, each series share one label for the floor type. If use common methods to preprocessing the data and train the model, the performance of the model will suffer from the increasing of the dimensionality, which we have mentioned in the class, the curse of dimensionality.So we should lower the dimensionality of the raw data. So we try some methods with EDA, and use the analysis from the EDA to do the preprocessing of the data. This will help us with the dimensionality reduction. And as our previous experience tell us, the neural network will have more stable performance on the prediction, compared to other algorithm, so in our experiment, we use the simple preprocessed raw data to train our Neural network, but use other more advanced preprocessed data to train other model. We expected to see whether our data preprocessing method are useful or not.

8 EXPERIMENTAL RESULTS

8.1 the performance of tree based model

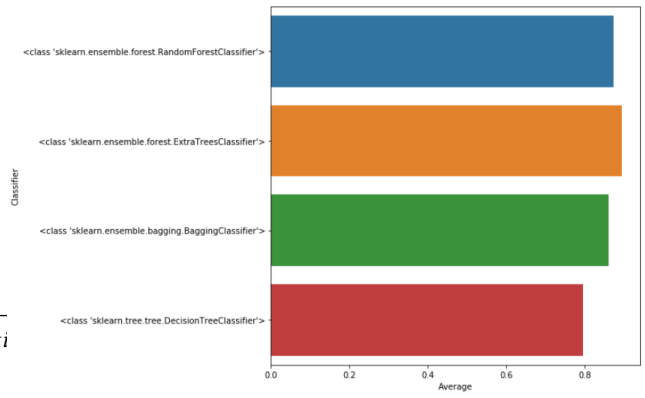

Figure 9: comparison between several models

The result (Figure-9)shows here is as we expected, the tree based model have a good performance, they all have really high accuracy of prediction, which is higher than 90%.

8.2 the performance of other kinds of model

The result (Figure-10) shows here is also as we expected, the imbalance data will have a bad performance on prediction, like KNN. The Adaboost, we expected it to have a more stable performance because of the overfitting-avoid methods, but it also failed. The accuracy of the model is as low as 40%, far below the performance of the tree-based model.

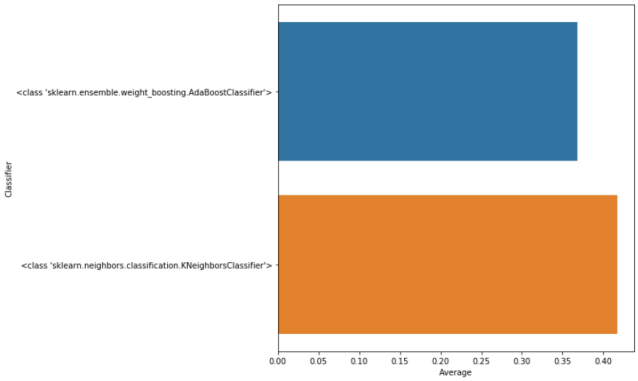

Figure 10: comparison between several models

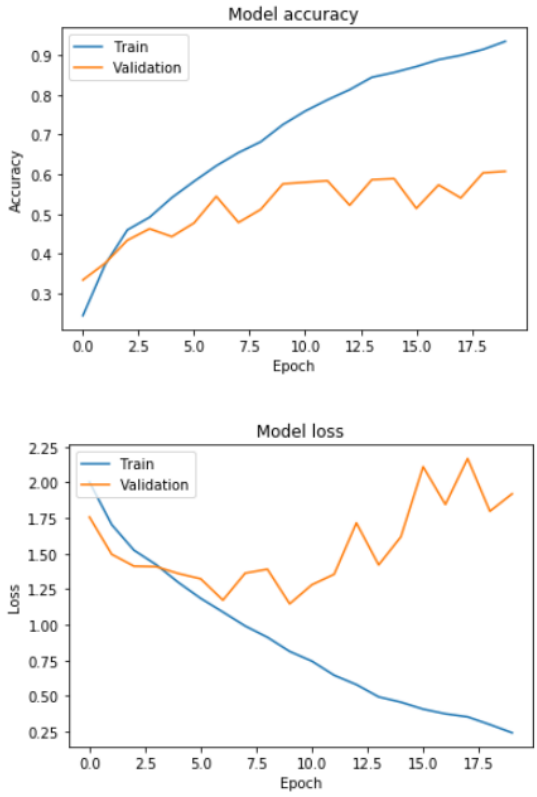

Figure 11: the model accuracy and loss after each epochs

8.3 the performance of CNN

As the figure shows here (Figure-11), loss starts to rebound around epoch 10. We apply early stop at epoch 10, when accuracy: 0.7251 and validation accuracy: 0.5761. The predictive accuracy is low.

9 ANALYSIS OF THE RESULTS

From the performance of three groups, we can get the rank of model: tree-based model»neural network>other model(in our project is adaboost and knn). But in neural network, we do not try too much data preprocessing, only combine 128 rows data into a matrix, which is to say, only change the shape of input matrix. But for other model, we try to preropcessing as we mentioned above. In adaboost, the performance is really low evern though we do the preprocessing, after we do some researching, the reason maybe many outliers stay in our dataset, that have a bad effect on the performance. The imbalanced training samples is the main reason why CNN and KNN has a mediocre performance. From previous experience, neural network will always give us the best predicting performance, and tree-based model will have more chance to overfitting. But in this imbalance dataset, the results show here are the opposite. we think this is because the preprocessing. As now, we can know the data preprocessing is really important. Based on the previous analysis, we should choose models trained by Random Forest, Extra Trees and Bagging to be the floor type classifier of the robot as they achieved over 90% accuracy.

10 FUTURE WORK

In the future, we can also apply classifiers to the visual information and integrate with these ones which based on movement information to get a higher predictive accuracy. Further improvement on data preprocessing and model parameters may increase the accuracy score as well.

For now, the methods we used in EDA part is pretty simple, maybe in the future, we should learn more about how to do EDA more efficient, this will lead to a better way to do the data preprocessing also.

11 CONTRIBUTIONS

As a group working on this collaborated project. Ran Yi did the part of EDA, model for tree-based, KNN and adaboost. Daoyuan Chen did the part of CNN.

ACKNOWLEDGMENTS

We would like to express our most sincere gratitude to our professor (Yue Ning) for providing us such a good course, providing us with interesting and useful reading materials (like the classic and standard textbook, Pattern Recognition and Machine Learning)and online learning resources in our spare time, helping us in the final project. Machine learning and Artificial Intelligence have gained their popularity in recent years, this course, as the machine learning master's core course, is a very important and basic step for knocking at the gate to the machine learning area. Applying the lessons learned in class to a real project, but also providing full support for our project work. And always guide us. Without her encouragement, support and guidance, we will not be able to get too much from this course. This course is very important in the work. We hope that in the future, We will have the opportunity to apply the knowledge We have learned in class to work.

REFERENCE

[1] Chary R., Lakshmi D.R., Sunitha K. Feature extraction methods for color image similarity. Adv. Comput. An Int. J. 2012 doi: 10.5121/acij.2012.3215.

[2] Nilsson, J. O.; Gupta, A. K.; HÃďndel, P. (October 2014). "Foot-mounted inertial navigation made easy". 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN): 24âĂŞ29. doi:10.1109/IPIN.2014.7275464.