The Optimized Physical Model for Real Rover Vehicle

Jun-young Kwak
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
junyoung.kwak@cs.cmu.edu

Abstract

This paper presents the way to select appropriate features for rover vehicle model and to reduce the errors (differences) between two different vehicle models. The fundamental idea for selecting features is to compare the error rates when the specific group of features is removed and to choose the best case. Based on selected features and learning algorithm, I find the optimal coefficients and adjust the system properties using those values. Results from a relevant simulation experiment provide foundations to support and illustrate the benefit of the devised ways. The paper concludes with several promising directions for future research.

1 Introduction

Over the past few years, rover vehicles for Mars exploration have been rapidly developed based on fundamental technologies. However, the measure and prediction of the parameter related to a wheeled ground robot while driving is still complicated. High levels of slip and friction can be observed on certain terrains and the specific condition, which can lead to significant errors of the vehicle's movement, inability to reach its goals, or, in the worst case, getting stuck without the possibility of recovery [1]. The reduction of cost for analyzing the vehicle model and setting up some experimental procedures is also getting an important issue. Current Mars rover vehicles have a lot of parameters and properties and we have to consider select relevant features in parameters to reduce the vehicles' complexity and get the optimal solution for exploration and navigation on the rough-terrain.

To safely perform tasks in unknown or complicated environments, learning algorithm can be applied to navigation on the rough-terrain using different vehicle models. Using an online learning method directly learns the mapping between two different vehicle models and data through the experiment. The system can be trained by simply driving through representative dataset. The optimal vehicle model that encodes structure of the real rover platform represents the coefficients like suspension, friction and max torque and similarity on the same terrain.

This paper provides the methods for selecting the optimal features in real dataset and reducing the execution errors between two different vehicle models using learning algorithm. Doing so provides extra information to the algorithm that allows it to operate in more than just a naive manner using whole features in dataset. These suggested techniques show improvements for the reducing whole error rates during learning and execution procedures.

1.1 Related work

Using structure to constrain problem. A limitation of the constrained method is that predictions are made independently in each small patch of terrain, without including any spatial context. This can cause problems when there is ambiguous feature data [3]. If the system considers each of these patches independently, it will give the same ground estimate for both and get at least one of them wrong. This work relaxes the strong assumption of independence between selected features through the inclusion of spatial correlations.

Learning to predict slip for rover. Slip can be defined as the difference between the different vehicles estimated by several factors. This work predicts the amount of slip an exploration rover by learning from examples of traversing similar terrain. Learning from examples allows the system to adapt to unknown terrains rather than using fixed heuristics or predefined rules [2]. This works also consider slip learning in a nonlinear regression framework. They describe a general framework to learn the functional relationship between visual information and the measured slip using training examples.

2 Approach

2.1 Assumptions

In this work, I use two different simulators — CMU Vortex simulator and NASA JPL ROAMs simulator (See Figure 1). These two simulators have their own vehicle models, but the other parts like path planning algorithm is the exactly same. In this paper, to simplify the problem I assume that the terrain condition and internal control part of each vehicle model are the same. Based on this assumption, I concentrate on finding factors making the execution errors between two simulators like tire friction, slippery, suspension and max torque. For the path planning algorithm, I used heuristically-guided RRT (hRRT) [4] algorithm to collect data and to test the vehicle models.

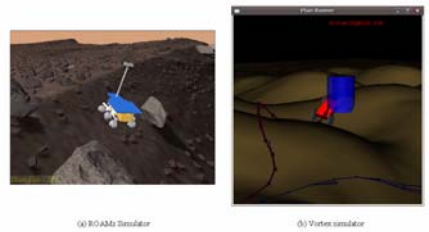


Figure 1: Vortex and ROAMs simulator

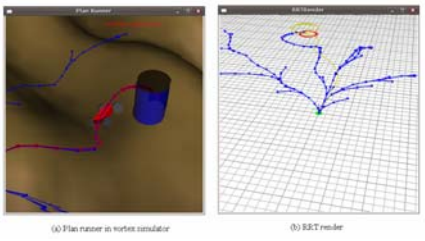


Figure 2: Collecting data using vortex simulator

2.2 Collecting data

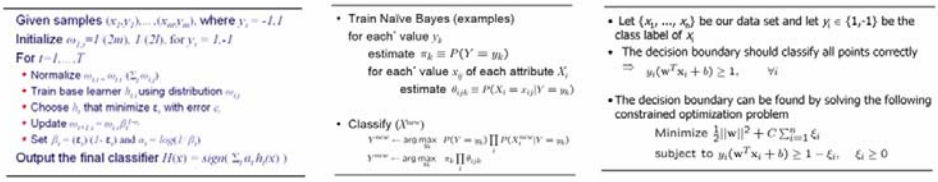
The data used in the paper was collected from two different simulators. In this work, there are two different dataset and one is obtained from Vortex simulator and another is obtained from ROAMs simulator. To obtain the data, I repeated generating the path and storing vehicle information on the planned path (See Figure 2). After repeating 20 experiments, I got 798 data and each data consists of 59 features. These features represent the physical vehicle model and include yaw of body, linear velocity, angular velocity, quaternion of each part, curvature, current car position and current planned position.

Features: Current car position(x, y, z), Current planned position(x, y, z), Yaw, Curvature, Actual velocity, Quaternion of body(x, y, z, w), Quaternion of each wheel(x, y, z, w) *4, Linear velocity of body(x, y, z), Linear velocity of each wheel(x, y, z) *4, Angular velocity of body(x, y, z), Angular velocity of each wheel(x, y, z) *4

2.3 Selecting the optimal features in dataset

2.3.1 Algorithm

In this part, I explain the way to find optimal features (or parameters) in dataset with AdaBoost algorithm (See Figure 3(a)). For selecting features, I use AdaBoost algorithm because that makes the error low and help to reduce the accidental cases. This algorithm provides the method for improving the accuracy of any learning algorithm and uses weak algorithms for single rules. It also combines weak rules into a strong learning algorithm by weighting weak learners. To apply AdaBoost algorithm to this problem, I used linear classifiers (Naïve Bayes classifier) as a weak learner.



(a) AdaBoost algorithm (b) Naïve Bayes classifier (c) Linear SVM with regularization

Figure 3: Used learning algorithm

I use the physical vehicle information as features and difference between planned positions and current vehicle positions as the target information.

(1) Features (X): The physical vehicle model (59 features)

➔ Yaw of body, linear velocity, angular velocity, quaternion of each part, curvature, current car position and current planned position

(2) Target (y): difference between the planned position and the current vehicle position (execution error)

- In this case, to simplify the problem, I assign the class to the target information. ($y_i \in \{-1,1\}$)

2.3.2 Evaluation

As the criteria to select features, I compare error rates for each case. Basically, data can be divided by some category and using this property I make different combinations of features. The test for each case is performed with AdaBoost learning algorithm and based on the error rate results I choose the feature sets having a high error rate. If the important feature set, i.e. that feature set have a big effect on the target data, is removed, the error rate for fitting and classifying become bigger. To check more accurately, each error rates are compared with the case which all features are used (base case).

- $D_i = w_1 (\varepsilon_{1,i} - b_1) + w_2 (\varepsilon_{2,i} - b_2)$, where D_i : weighted error difference, w_i : weight factor, $\varepsilon_{1,i}$: training error, $\varepsilon_{2,i}$: test error, b_i : base error
- If weighted error difference is bigger than threshold value, that feature set is selected. (This is because the big difference represents that that feature is more important one.)

2.4 Reducing the execution errors

Second part is to find the optimized learning algorithm to fit existing simulation model to optimal physical one for Mars rover vehicle using selected features. In this part, with two learning algorithms — Naïve Bayes classifier and SVM with regularization (See Figure 3(b), (c)), I compare the performance in reducing errors between two different model. Based on these experiments results, I find the optimal coefficients to fit the existing model better to real vehicle model. To learn the target data (execution error), slip of the rover vehicle is used. For the first step (see 2.4.1), I predict the amount of slip and compare the error using two different classifiers. Based on that result, the better classifier is selected and I find the optimal coefficient for the function between slip and features. For the second step (see 2.4.2), using the selected classifier I predict the target data (execution error) and find the optimal parameters for the function between execution error and slip.

2.4.1 Comparing two learning algorithms

2.4.1.1 Algorithms

For the first step, I compare two different learning algorithms — Naïve Bayes classifier and SVM with regularization (See Figure 3) to learn and predict the amount of slip using selected features in Part 2.3.

2.4.1.2 Finding the optimal coefficient for the function

Target (s): the amount of slip of vehicle

- The amount of slip s per wheel can be defined as a difference between the velocity measured by the wheel and the actual velocity: $s = wr - v$ [5] (w: angular wheel velocity, r: the wheel radius, v: actual linear velocity)
- In this case, to simplify the problem, I assign the class to the target information. ($s_i \in \{-1,1\}$)

Based on information I find a linear equation using non-parametric algorithm (Locally weighted linear regression).

$$s = \theta_0 + \theta_1 x_1 + \ldots + \theta x_i + \ldots + \theta x_n = \theta_0 + \sum_{i=1}^{n} \theta x_i$$

$$\text{subject to } J(\theta) = \frac{1}{2}\sum_{i=1}^{n} w_i (x_i^T \theta - y_i)^2$$

2.4.2 Predicting the execution error and finding the optimal parameter to reduce the error

For the second step, I predict the target data (execution error) with selected learning algorithm and find the optimal parameter for the function of the slip and execution error using locally weighted linear regression. (See 2.4.1.2)

Target (y): difference between the planned position and the current vehicle position (execution error)

➔ To assign the class to the target data, I use the same rule of Part 2.3.1.

3 Experimental results

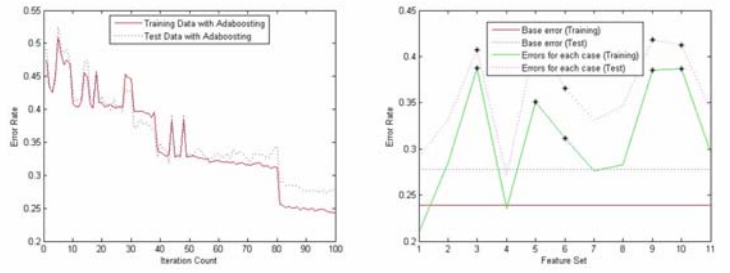
3.1 Experiment setup

This research is motivated mainly by planetary rovers, such as The Mars Exploration Rover (MER), so I used Vortex simulator and ROAMs simulator as testbed. Vehicle models in both simulators are made based on real rover platform like Rocky8 and FIDO. In this experiment, the vehicle speed in the simulator is set to 5km/h between planned nodes. I tested the experiments using MATLAB to analyze and plot the graph.

3.2 Selecting the optimal features in dataset

For AdaBoost algorithm, I chose 100 as an iteration number and did 10-fold cross-validation on the dataset. I tested different 12 cases with various combinations of features. Figure 4(a) is the plot of the error rate when all features in dataset are used. As shown in Figure 4, when AdaBoost algorithm with linear classifier is applied to this problem, the error rate is about 0.25 for the training data.

To select the feature set, I use the evaluation equation $D_i = w_1 (\varepsilon_{1,i} - b_1) + w_2 (\varepsilon_{2,i} - b_2)$, where $b_1 = 0.2392$, $b_2 = 0.2781$, $w_1 = 0.3$, $w_2 = 0.7$, threshold = 0.083. Figure 4(b) and table 1 represent the experimental results.



(a) Error rate when all features in dataset are used

(b) Error rate for each case. (* is the selected feature set)

Figure 4: Error rates for selecting features

Table 1: Selecting features based on error rate

CASE (REMOVE SPECIFIC FEATURES) *	ERROR RATE FOR TRAINING DATA	ERROR RATE FOR TEST DATA	WEIGHTED DIFFERENCE (D_i)
Nothing (Base Error)	0.2392	0.2781	
1. Current car position	0.2098	0.2916	6.3000e-4
2. Current planned position	0.2852	0.3323	0.0517
3. Yaw	0.3873	0.4070	0.1347
4. Curvature	0.2354	0.2715	-0.0058
5. Actual velocity	0.3510	0.4164	0.1303
6. Quaternion of body	0.3118	0.3657	0.0831
7. Quaternion of wheels (RMS error) *2	0.2766	0.3311	0.0483
8. Linear velocity of body	0.2829	0.3464	0.0609
9. Linear velocity of wheels (RMS error) *2	0.3852	0.4182	0.1419
10. Angular velocity of body	0.3871	0.4122	0.1382
11. Angular velocity of wheels (RMS error) *2	0.2957	0.3412	0.0611

As shown Table 1, I can choose five feature sets (cells filled by orange color) which are bigger then threshold value (0.083). This result means if those five features sets are removed, it is not enough to learn target data using remaining features.

- Selected Features: 21 among 59 features.

  ➔ yaw, actual velocity, quaternion of body (x, y, z, w), linear velocity of wheels (x, y, z) * 4, angular velocity of body (x, y, z)

3.3 Reducing the execution errors

3.3.1 Comparing two learning algorithms

(1) To compare two different two learning algorithms, I did 20-fold cross-validation on the dataset — Target(s): slip, X: selected features. Figure 5 is the plot of the accuracy when Naïve Bayes and SVM are applied to the dataset. For the SVM (regularization), I use $\lambda = 10^{-10}$, C=1.

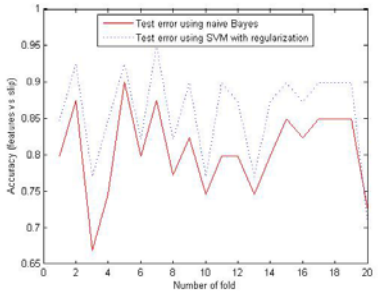


Figure 5: Accuracy with two learning algorithms

Table 2: Accuracy values for each dataset

NUMBER OF DATA FOLD	NAIVE BAYES	LINEAR SVM WITH REGULARIZATION
1	0.7974	0.8474
2	0.8744	0.9244
3	0.6692	0.7705
4	0.7462	0.8474
5	0.9000	0.9244
6	0.7974	0.8218
7	0.8744	**0.9500**
8	0.7718	0.8218
9	0.8231	0.8987
10	0.7462	0.7705
11	0.7974	0.8987
12	0.7974	0.8731
13	0.7462	0.7705
14	0.7974	0.8731
15	0.8487	0.8987
16	0.8231	0.8731
17	0.8487	0.8987
18	0.8487	0.8987
19	0.8487	0.8987
20	0.7246	0.7044
Average	0.8040	0.8582

*1 In Table 1, each CASE means removing each term in the table cell. For example, Nothing means removing nothing in features (i.e. using all features).

*2 In Table 1, RMS errors are root-mean-square errors of four wheels.

As shown in Figure 5 and Table 2, when linear SVM with regularization ($\lambda = 10^{-10}$, C=1) is used, I can get higher accuracy rates than Naïve Bayes classifier. Based on this result, I select linear SVM to predict the target data (execution error) using slip (Part 3.3.2).

(2) To find the function(s), I use locally weighted linear regression. Weight values can be obtained from the training result with SVM and as shown in Table 2, case 7 is the best case, and thus I use the weight value of that case. Step size is 10^{-5} and threshold is $0.5 * 10^{-4}$. Table 3 represents the optimal parameters for the function s (slip) of Part 2.4.1.3.

Table 3: The optimal parameters for the function s (slip)

FEATURES (X_i)	PARAMETERS (θ_i)
1	2.1653
2	-0.0840
3	-0.5357
4	1.9580
5	0.5659
6	-0.5700
7	0.4491
8	0.0183
9	0.0556
10	-0.0698
11	0.1411
12	-0.1040
13	-0.2276
14	-0.1866
15	0.1520
16	0.0883
17	-0.2251
18	-0.1965
19	-0.0654
20	0.1117
21	0.1085
22	0.1777

Table 5: The optimal parameters for the function y (execution error)

FEATURES (X_i)	PARAMETERS (θ_i)
1	1.0973
2	-0.0999

3.3.2 Predicting the execution error and finding the optimal parameter to reduce the error

(1) Using linear SVM with regularization ($\lambda = 10^{-10}$, C=1), I also did 20-fold cross-validation on the dataset — Target(y): execution error, X: slip(s). Figure 6 is the plot of the accuracy when linear SVM is applied to the slip dataset. To verify the result, I compared the test error based on slip with two random cases. Random case1 has the very similar features to the amount of slip $(-1 \le X \le 4)$ and random case2 has totally randomly generated data. (The data for two random cases are generated using MATLAB rand function.)

As shown in Figure 6 and Table 4, we can know that slip value is more correlated to target data (execution error) than other random case. This means if we can find the optimal model for slip, we can also find the optimal model for execution error using slip value.

(2) As Part 3.3.1, to find the function(y), I use locally weighted linear regression. Step size is 10^{-5} and threshold is 10^{-7}. Table 5 represents the optimal parameters for the function y (execution error).

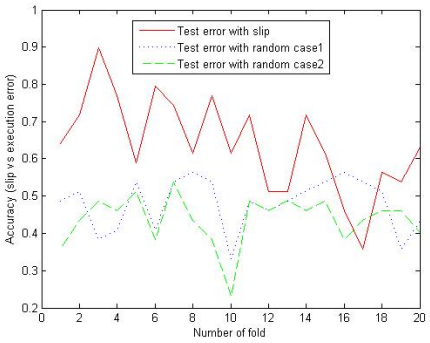


Figure 6: Accuracy rate for predicting the target value

Table 4: Accuracy values for each dataset

NUMBER OF DATA FOLD	SLIP	RANDOM CASE1	RANDOM CASE2
1	0.6410	0.4872	0.3590
2	0.7179	0.5128	0.4359
3	0.8974	0.3846	0.4872
4	0.7692	0.4103	0.4615
5	0.5897	0.5385	0.5128
6	0.7949	0.4103	0.3846
7	0.7436	0.5385	0.5385
8	0.6154	0.5641	0.4359
9	0.7692	0.5385	0.3846
10	0.6154	0.3333	0.2308
11	0.7179	0.4872	0.4872
12	0.5128	0.4615	0.4615
13	0.5128	0.4872	0.4897
14	0.7179	0.5154	0.4615
15	0.6154	0.5385	0.4872
16	0.4615	0.5641	0.3846
17	0.3590	0.5385	0.4359
18	0.5641	0.5128	0.4615
19	0.5385	0.3590	0.4615
20	0.6316	0.4316	0.4035
Average	0.6393	0.4807	0.4383

4 Discussion and conclusion

I have proposed a method to learn to select relevant features and predict slip and execution error using selected one. This paper makes a step forward in modeling and learning in complex environments using real data without involvement of detailed mechanical models. For selecting features, I used AdaBoost algorithm because that makes the error low and help to reduce the accidental cases. Using linear SVM with regularization, I also showed that slip prediction results can be directly incorporated into the execution error. Based on the results, we can conclude if we want to find the optimal model of vehicle to reduce the execution error, we can find the optimal slip model using the proposed method to reduce the number of parameters (features). Using the best fitted function (and its coefficients) and the slip model, we also can find the optimal model for execution error of vehicle. I demonstrated on real datasets how this method can be used to predict the slip and execution error and find the optimal model.

References

[1] A. Angelova et al. (2006), Learning to Predict Slip for Ground Robots, *IEEE International Conference on Robotics and Automation*.

[2] C. Wellington & A. Stentz. (2003), Learning Predictions of the Load-Bearing Surface for Autonomous Rough-Terrain Navigation in Vegetation, *International Conference on Field and Service Robotics*.

[3] B. Sofman, E. Lin, J. Bagnell, N. Vandapel & A. Stentz. (2006), Improving Robot Navigation Through Self-Supervised Online Learning, *Robotics: Science and Systems*.

[4] C. Urmson & R. Simmons, (2003), Approaches for Heuristically Biasing RRT Growth, *IEEE/RSJ International Conference on Intelligence Robots and Systems*.

[5] Wong J., (1993), Theory of Ground Vehicles, *John Wiley & Sons Inc.*.