Fault Detection and Analysis of Control Software for a Mobile Robot

Jiliang Lin, Jingping Jiang
College of Electrical Engineering, Zhejiang University,
Zheda Road, Hangzhou, Zhejiang, 310027, China
LinJL@zju.edu.cn

Abstract

In certain circumstances mobile robots are unreachable from human being, for example Mars exploration rover. So robots should detect and handle faults of control software themselves. This paper is intended to detect faults of control software by computers. Support vector machine (SVM) based classification is applied to fault diagnostics of control software for a mobile robot. Both training and testing data are sampled by simulating several fault software strategies and recording the operation parameters of the robot. The correct classification percentages for different situations are discussed.

1 Introduction

Mobile robots are widely used in industry, aerospace and military areas. In industry, faults of software in mobile robots may cause downtimes of entire production lines with the consequence of production losses and high costs just like faults of hardware do. In aerospace and military sectors, it may also threaten our securities. Hence detection and analysis of control software for mobile robots are of an important meaning for their applications.

Fault is unavoidable when developing control software for mobile robots. Debugging control software can be a difficult and time-consuming problem, which depends on the experience of its developers. An experienced developer who has come through similar symptoms before can quickly identify the problems, while inexperienced one will find it a daunting task and may cost long time to fix them. Fault detection and diagnosis may assist the developers to debug the control software and may save much time for a novice. A kit of mobile control software may be developed by groups of developers. Fault detection and diagnosis may also help them to locate the errors if some developers may not know the modules from others very well.

Faults of control software for a mobile robot may be encountered infrequently in its operation period. Acquisition the fault feature is more difficult and more risky than that from a normal one. Support Vector Machines (SVM) are new tools for pattern recognition that have a good performance despite of insufficient training samples. In this paper, SVMs are applied to detect faults of control software for a mobile robot.

2 Related Works

Many efforts have been focused on automatic fault detection and identification of software. Bose et al [1] use data mining approaches to diagnosis software faults. They think that a system call execution trace can capture some dynamic states of the system and it is helpful in identifying software faults. They do not have data for the fault specific system calls so they experiment their approach on the host based intrusion datasets. They apply the One Class Support Vector Machine classification technique and Latent Semantic Analysis approach on different Unix programs. Most intrusion traces are classified correctly and the performance they get is almost as good as Hidden Markov Models (HMMs) approach while the computing complexity is much lower than the latter.

Warrender et al [2] compare four methods for characterizing normal behaviors and detecting intrusions based on system call in privileged processes. Although no one method of those four gives the best results on all programs have been tested, the paper reports that HMMs generally recognized as one of the most powerful data modeling methods in existence which give the best accuracy on average. But the computational cost is so high that they spend two months to train a model. As [2] mentions that applying SVM is available because it has almost the same performance as HMMs but lower cost than HMMs.

Hunt [3] presented a system called WATSON which automatically constructs of an explanation of faults and a repair strategy from a library of particle repair models. WATSON is developed in SmallTalk, and it handles SmallTalk exceptions by exploiting Case Based Reasoning technology.

Brun et al [4] use support vector machine and decision

Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)
0-7695-2528-8/06 $20.00 © 2006 IEEE

tree learning tools to classify the fault-revealing properties lead a programmer to an error. They propose machine learning models of program properties known to result from errors, and apply these models to program properties of user-written code to classify and rank those properties. Given a set of properties, the models select a subset of properties that are most likely to reveal an error. Their experience suggests that most of those properties do lead to an error.

Goel et al [5] use multiple model estimation and neural network to detect and identify faults in hardware of a mobile robot. The faults they detected include flat tires, gyros failing and encodes failing. In this paper faults of control software for a mobile robot are detected and identified.

3 SVM classification

Support Vector Machines (SVM) were first introduced by Vapnik [6]. They are developed based on the structural risk minimization inductive principle from statistical learning theory and from optimal classification hyperplane under linear separatable condition. Let n-dimensional input $x_i(i = 1...n)$ belong to class I or class II and associate labels $y_i = 1$ for class I and $y_i = -1$ for class II. Figure 1 shows the basic idea of SVM. The circles and stars present two classes of training data. Classification line H shatters these data correctly. H_1 and H_2 which ran cross the nearest points are lines parallel with H. The distance between H_1 and H_2 is the margin. The points on H_1 and H_2 are called support vectors. The optimal classification hyperplane not only shatters two kinds of data without mistake which ensures the minimal experiment risk (which is zero), but also makes the maximum margin which means the minimal confidence interval. This method minimizes the structure risk which composition of experiment risk and confidence interval, so the actual risk is minimal. In high dimension, the optimal line becomes optimal hyperplane.

For non-linear separable cases, SVM maps the lower dimension data to be classified onto a higher dimension feature space, where the data can be linearly classified. The mapping functions are called Kernel Functions. The following four kernels are used to test their predict accuracies:

1. Linear:$K(x_i, x_j) = x_i^T x_j$
2. Polynomial:$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
3. Radial Basis Function (RBF):$K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \gamma > 0$
4. Sigmoid:$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Here γ, r, and d are kernel parameters.

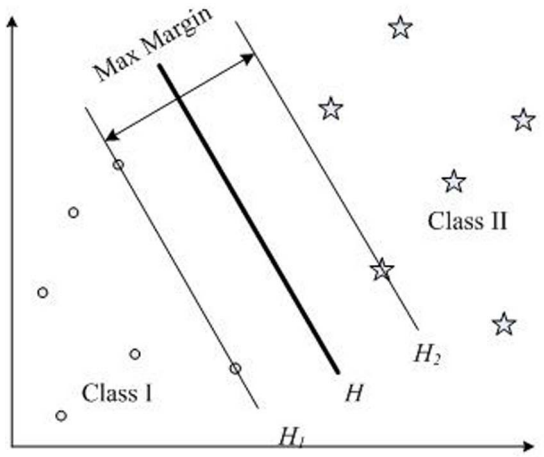

Figure 1. Optimal hyperplane

4 Multi-Classification

Several fault classes may exist in a suit of control software for a robot. So a method to classify multi-class is needed. Currently, applying SVM into multi-class classification is still under study. One method is called "all together" to classify multi-class which only one SVM is used and it outputs several results. This method concerns complicated optimism problems. It is time consuming and the classification accuracy is low when sample sets are large. Another way is to construct several binary classifiers and combine them together to distinguish multi-class. It includes "one-against-one" method and "one-against-all" method. In this study we use "one-against-one" method and a majority voting approach which are efficient and easy implemented. $k(k-1)/2$ classifiers are constructed, here k presents how many fault classes to be classified. Each of them which trains data from two different classes is considered to be a voting where votes can be cast for all data points x-in the end point is designated to be in a class with maximum number of votes [7].

5 Experiment Data

Some possible software faults are simulated on a mobile robot Pioneer 3. Pioneer 3 can move arbitrarily with its two drive wheels and a caster. It has position encoders, sonar sensors, robotic pan-tilt-zoom camera and precision range-finding laser for auto-navigation (Fig 2). We program the control software of Pioneer 3 with the following intended faults so that it wanders in a large office.

We measure the data of velocity, angel velocity from encodes and distances from obstacles in 16 directions from

Table 1. Codes and descriptions for control software faults

Code	State Desciption
NS	Normal state
SL	Left wheel is slower (Left flat tire)
SR	Right wheel is slower (Right flat tire)
GS	Getting stuck
MR	Move round in a small area
PC	Program collapse

sonar sensors each second. These data constitute 18-dimensional input $x_i(i = 1, \ldots, 18)$ of the support vector machine. Available detective range of the sonar sensors is less than 5 meters. The sensors return only five meters even if a distance between the robot and an obstacle is longer than five. We also try the laser sensor which is much more precise than sonar.

We simulate five faults and a normal state of the control software which are presented in Table 1. Normally, Pioneer 3 wanders in the office arbitrarily. When it approaches obstacles, it slows down and turns to avoid it in a safety space. We may program the velocity of left wheel slower than the right one by mistake. This can also simulate left flat tire. On the contrary slower the right wheel to simulate right flat tire. The robot run fast without slowdown will bump or get stuck in the obstacles. We limit the turn angel of the robot when the robot approaches a corner or run in a narrow walkway so that the robot will get stuck. If we prolong the safety space, the robot may turn round continuously in a small area. We simulate program collapse on a single-chip-controller AT89C51. We tamper with the executable binary control software intended and sample its output. Fig 3 and Fig 4 are actual photos and sketch map of Pioneer 3 wandering in an office.

6 Results

The robot is placed in the office arbitrarily and wanders 10 times in each fault case. Each time the robot wanders for 5 minutes. Then we get 60 sets of sample data and each set has 300 vectors. We combine the data of same fault together so we finally have 6 sets of data. Each set is separated into two parts, one part for training support vector machine and the other for testing. In normal cases, the robot wanders arbitrarily. If there is a wall in front of it, it slows down and gets off the obstacles in the same angle that it approaches. If there are obstacles in front, left and right sides of it, the robot stops and goes back. There is always a positive rotate velocity when the robot wanders in the condition of left wheels being slower than the right one. In this condition,

Figure 2. Pioneer 3 with all sensors

Figure 3. Pioneer 3 is wandering in an office

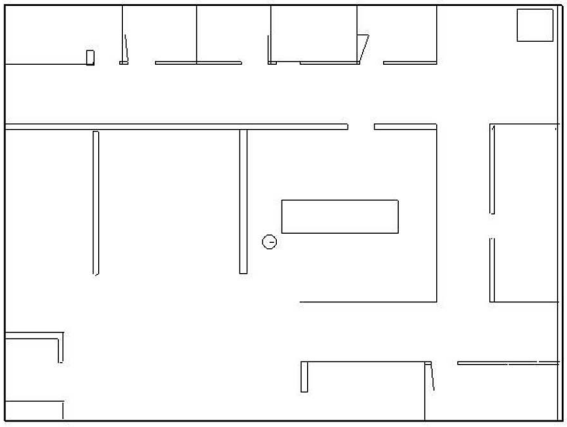

Figure 4. The sketch map simulates the wandering of Pioneer 3 in an office.

the robot walks along the wall instead of getting off the wall. If the right wheel move slower than the left one, it has the same behavior except the rotate velocity is always negative. When the robot goes into a narrow space, it makes its rounds for a long period of time and can not go out itself. In this case the control software should label it as fault condition and assume some other control strategy. Thus the distances to the walls are small and they change again and again in a small range. If the robot moves too fast, it may rush into a narrow corridor and get stuck. The variety of speed, rotate velocity and the distances will stop changing while the operation parameters are just like normal before the stuck. If the program collapses, the operation parameters are almost all zero except some of them change into random values casually.

We design 15 2-class SVM classifiers because we have 6 classes of sample data, one normal class and five fault classes. Here we use SVM tools developed by Chang et al [8]. After comparing four kernels mentioned above, we find that the RBF kernel is the most convenience while it performs as good as others. So RBF kernels are assumed in all these SVMs. Pairs of parameters of RBF kernel are tried and the one with the best cross-validation accuracy is picked for each SVM. Table 2 shows the correct classification percentages between different classes for testing sets. Getting stuck is difficult to distinguish from normal state, move round in a small area and program collapse. Table 3 shows the percentages of support vectors for training sets. Comparing with Table 2 we found that the less support vectors, the higher classification percentage between two classes of fault sample data.

We combine 15 classifiers together and use the majority voting approach mentioned in [7]. One hundred of samples from each fault class are tested with this combining classifier. The first line of data in Table 4 shows the correct classification percentages in the condition that no noise exists. Noisy signal is added to the testing data which is 1% in amplitude and its mean value is zero. We test again and find that the correct classification percentage degrade from 85% to 63%. After applying noise filtering [9], the classification percentage increases into 77%.

Table 2. Classification percentages for testing sets

Percentage	SL	SR	GS	MR	PC
NS	100	100	62	90	100
SL	-	100	100	85	100
SR	-	-	100	85	100
GS	-	-	-	68	32
MR	-	-	-	-	100

Table 3. Percentages of support vectors for training sets

Percentage	SL	SR	GS	MR	PC
NS	5	5	58	18	10
SL	-	3	5	25	4
SR	-	-	5	25	4
GS	-	-	-	41	73
MR	-	-	-	-	12

We train the classifier in the office and put the robot in an unknown environment. Same data for each fault class are sampled. We try the classifier on these new data and find that the classification percentage descends slightly. This means that the classifier can also applied even if the robot goes into unknown surroundings. We also try experiments by replacing the sonar sensors into laser, and increasing the speed of the robot. Table 5 shows that the classification percentages become very low, which means the classifier can not be applied in these two conditions.

7 Conclusions

This paper applies SVM to diagnosis the faults of control software for a robot. A normal robot wandering condition and 5 fault ones are simulated. The operation parameters of velocity, rotate velocity, distances to obstacles from sonar sensors are acquired as the input vectors of the SVM for both training and testing. A majority voting approach is applied to deal with multi-class classification.

The experiment shows that the classification with multi-SVMs can diagnose the faults of control software for a mo-

Table 4. Correct classification results of combined classifiers with and without noise

Percentage	NS	SL	SR	GS	MR	PC	Total
Without noise	89	100	100	61	78	82	85
With noise	82	95	95	18	65	21	63
With noise filter	85	96	97	57	71	57	77

Table 5. Correct classification results of combined classifiers, while test conditions changed

Percentage	NS	SL	SR	GS	MR	PC	Total
Same environment	89	100	100	61	78	82	85
New environment	82	100	100	60	62	82	81
Replace sensors	23	73	74	54	28	28	47
Higher speed	15	42	41	33	18	81	38

bile robot very well especially the sample data sets without noise. The classification accuracy in this condition is 86%, while it is reduced to 52% by the sampling noise. With noise filtering the fault detection rate increases to 64%. In all conditions, the distinguish rate for left flat tire and right flat tire are all above 90% because the robot always has a positive or negative rotate velocity in the whole wandering process. Getting stuck is difficult to detected, but we think this may be improved if we have sampled the voltage and current of the driving motor.

The classification structure can be used in new unknown environments. But if the wandering speed is different from the training condition, or if the sonar sensors have been replaced by laser sensor, then the trained fault detection SVMs do not work well any more, it needs to be retrained. Other methods are under study to compare with SVM classification technique proposed in this paper.

References

[1] Bose, R. P. J. C., Srinivasan, S. H.: Data mining approaches to software fault diagnosis. Proceedings of the 2005 IEEE 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications, pp: 1-8, 2005

[2] Warrender. C., Forrest, S., Pearlmutter, B.: Detecting intrusions using system calls: alternative data models. In Proceedings of the IEEE Symposium on Security and Privacy, May 9-12 1999.

[3] Hunt, J.: Case based diagnosis and repair of software fault. Expert systems Vol. 14. No.1. pp 15-23, 1997.

[4] Brun, Y., Ernst, M.D.: Finding latent code errors via machine learning over program executions. Proceedings of the 26th IEEE International Conference on Software Engineering (ICSE'04), 2004.

[5] Goel, P., Dedeoglu, G., Roumeliotis, S. I., Sukhatme, G. S.: Fault detection and identification in a mobile robot using multiple model estimation and neural network. Proceedings of the 2000 IEEE International Conference on Robotics and Automation, pp: 2302-2309, 2000

[6] Vapnik, V. N.: The nature of statistical learning theory. Springer, 1999.

[7] Poyhonen, S., Negrea, M., Arkkio, A., Hyotyniemi, H., Koivo, H.: Fault diagnostics of an electrical machine with multiple support vector classifiers. Proceddings of the 2002 IEEE International Symposium on Intelligent Control, pp: 373-378, 2002.

[8] Chang, C.-C., Lin, C.-J.: LIBSVM: a Library for Support Vector Machines.

[9] Valiviita, S., Ovaska, S.J., Vainio, O.: Polynomial predictive filtering in control instrumentation: a review. IEEE transactions on Industrial Electronics. Vol. 46, No. 5, pp. 876-888, 1999.