5th SASTech Iran |Mashhad May 12 - 17 | 2011
5th Symposium on Advances in Science & Technology

Failure Prediction Using Robot Execution Data

Tahereh Koohi*, Elham Mirzaie and Ghamarnaz Tadaion*****
***Student of Islamic Azad University of Mashhad, tahere.koohi@gmail.com**
**** Student of Islamic Azad University of Mashhad, elham_mirzaie_mshdiau@yahoo.com**
Professor of Islamic Azad University of Mashhad, tadayon@mshdiau.ac.ir
Name of the Presenter: Tahereh Koohi

Abstract

Robust execution of robotic tasks is a difficult learning problem. Whereas correctly functioning sensors' statements are consistent, partially corrupted or otherwise incomplete measurements will lead to inconsistencies within the robot's learning model of the environment. So, methods of prediction (classification) of robot failure detection with erroneous or incomplete data deserve more attention. Studies have shown that the techniques combining to classification has become an effective tool in increasing the efficiency and accuracy the classifieds. The primary goal of the evaluation was to analyze the impact of erroneous data on predictive robot fault detection accuracy .And then show category, increases performance classification despite noise with combined categories. In this regard have used the data set associated with torque obtained from a humanoid robot. In this paper the performance of base-level classifiers and meta-level classifiers is compared. Bagged Naïve Bayes is found to perform consistently well across different settings.

Key words: fault detection, incomplete data, and Meta classifiers.

1. Introduction

The term "fault detection" is commonly referred to as the detection of an abnormal condition that may prevent a functional unit performing a required function Chow et al. (1984), Merrill et al. (1988), Valavavins et al (1991). This paper is concerned with predicting the ability of a mobile robot to reliably monitor the execution of its plans and detect failures, given that some features or course of actions are missing.

Twala et al. (2009) has formulated this prediction as a classification problem that works with decision tree for handling erroneous or incomplete data with imputation. In this paper the performance of base-level classifiers and meta-level classifiers is compared. Bagged Naïve Bayes is found to perform consistently well across different settings, and so, we have shown the superiority of Bagged Naïve Bayes from twala et al.'s approach.

Robot failure execution dataset was used to train a set of classifiers, which included decision trees (C4.5), decision tables, naive Bayes classifier and nearest-neighbor algorithm found in the Weka Machine Learning Toolkit Witten et al. (1999). Naive Bayes classifiers showed the

best performance, predicting failures with an overall accuracy of 93.1818%. In addition to analyzing the performance of base-level classifiers Bao et al. (2004), we have studied the effectiveness of meta-level classifiers (such as boosting Freund et al. (1996), bagging Breiman et al. (1996), plurality voting, stacking using ODTs, and stacking using MDTs Todorovski et al. (2003) in improving failure prediction accuracy. We have tried to answer the following question:
Which are the best classifiers for predicting failures; is combining classifiers a good idea?
In the following sections, we describe our data collection methodology and our approach to predict failure from robot failure execution data, followed by results.

2. Data collection:

Each dataset obtained from the UCI repository of ML Blake et al. defines a different learning problem as shown in table1 and further described next.

Dataset	Instances	Number of classes (a (and their distribution)
LP1	88	4 (1 = 24%; 2 = 19%; 3 = 18%; 4 = 39%)
LP2	47	5 (1 = 43%; 2 = 13%; 3 = 15%; 4 = 11% 5 = 19%)
LP3	47	4 (1 = 43%; 2 = 19%; 3 = 32%; 4 = 6%)
LP4	117	3 (1 = 21%; 2 = 62%; 3 = 18%);
LP5	164	5 (1 = 27% ; 2 = 16% ; 3 = 13% ; 4 = 29% ; 5 =16%)

LP1= failures in approach to grasp position
LP2= failures in transfer of a part
LP3= position of part after a transfer failure
LP4= failures in approach to ungrasp position
LP5= failures in motion with part

Table 1: Each of these datasets defines a different learning problem

Just first dataset is used in this simulation study. In order to improve classification accuracy, a set of five feature transformation strategies (based on statistical summary features, discrete

Fourier transform, etc.) was defined and evaluated. This enabled an average improvement of 20% in accuracy. The most accessible reference is Lopes et al. (1998). All features are numeric although they are integer valued only. Each feature represents a force or a torque measured after failure detection; each failure instance is characterized in terms of 15 force/torque samples collected at regular time intervals starting immediately after failure detection; the total observation window for each failure instance was of 315 ms. each example is described as follows:

Class

Fx1 Fy1 Fz1 Tx1 Ty1 Tz1
Fx2 Fy2 Fz2 Tx2 Ty2 Tz2
.....
Fx15 Fy15 Fz15 Tx15 Ty15 Tz15
Where Fx1 ... Fx15 is the evolution of force Fx in the observation window, the same for Fy, Fz and the torques; there is a total of 90 features.

Each robot behavior is in the dataeset contains several classes like normal، collision, near collision, and are blocked. For example two trace of the typical robot can be seen in Figure 1 Seo et al. (2006). Also for check some time series samples we have drawn them in a diagram (Figure 2).

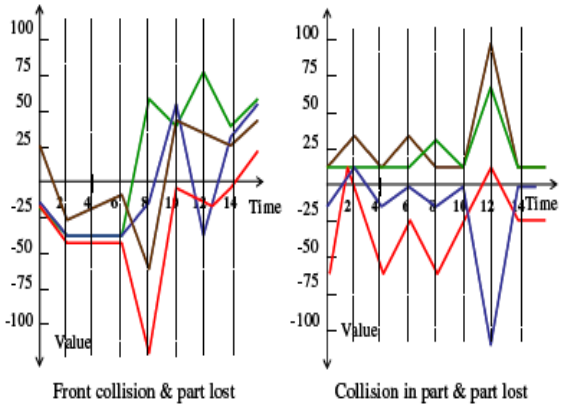

Figure 1: Two trace of the typical robot

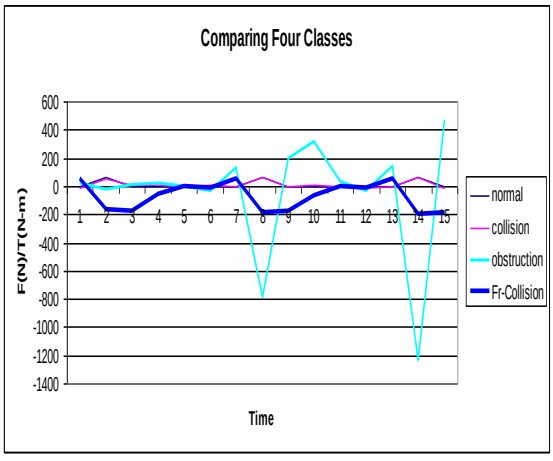

Figure 2: Comparison of Classes with some samples.

Since the distribution of missing values among attributes and the missing data mechanism were two of the most important dimensions of this study, one suite of data were created corresponding to MCAR. Therefore we have two settings in this study, one with missing values that across all specifications random completely (MCAR[1]) is distributed, and another have used missing-free.

3. Data Interpretation

We evaluated the performance of the following base-level classifiers, available in the Weka toolkit:

• Decision Tables
• Decision Trees (C4.5)
• SVM
• Naive Bayes.

We also evaluated the performance of some of the state of-the-art meta-level classifiers. Although the overall performance of meta-level classifiers is known to be better than that of base-level classifiers, base-level-classifiers are known to outperform meta-level-classifiers on several data sets. One of the goals of this work was to find out if combining classifiers is indeed the right thing to do for failure prediction from incomplete data, which to the best of our knowledge, has not been studied earlier. Meta-level classifiers can be clustered into three frameworks:

- Voting (used in bagging and boosting), stacking Wolpert (1992) and cascading Gama (2000). In voting, each base-level classifier gives a vote for its prediction. The class receiving the most votes is the final prediction. In stacking, a learning algorithm is used to learn how to combine the predictions of the base-level classifiers. The induced meta-level classifier is then used to obtain the final prediction from the predictions of the base-level classifiers. The state-of-the-art methods in stacking are stacking with class probability distributions using Meta Decision Trees (MDTs), stacking with class probability distributions using Ordinary Decision Trees (ODTs) and stacking using multi-response linear regression Seewald (2002). Cascading is an iterative process of combining classifiers: at each iteration, the training data set is extended with the predictions obtained in the previous iteration. Cascading in general gives sub-optimal results compared to the other two schemes.

To have a near exhaustive set of classifiers, we chose the following set of classifiers: Boosting, Bagging, Plurality Voting, Stacking with Ordinary-Decision trees (ODTs) and Stacking with Meta-Decision trees (MDTs).

- Boosting Meir (2003) is used to improve the classification accuracy of any given base-level classifier.

Boosting applies a single learning algorithm repeatedly and combines the hypothesis learned each time (using voting), such that the final classification accuracy is improved.
It does so by assigning a certain weight to each example in the training set, and then modifying the weight after each iteration depending on whether the example was correctly or incorrectly classified by the current hypothesis.

[1] - Missing Completely at Random

Thus final hypothesis learned can be given as

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x),$$

where α_t denotes the coefficient with which the hypothesis h_t is combined. Both α_t and h_t are learned during the Boosting procedure. (Boosting is available in the Weka toolkit.)

- Bagging is another simple meta-level classifier that uses just one base-level classifier at a time. It works by training each classifier on a random redistribution of the training set. Thus, each classifier's training set is generated by randomly drawing, with replacement, N instances from the original training set. Here N is the size of the original training set, itself. Many of the original examples may be repeated in the resulting training set while others may be left out. The final bagged estimator, $h_{bag}(.)$ is the expected value of the prediction over each of the trained hypotheses. If $h_k(.)$ is the hypothesis learned for training sample k,

$$h_{bag}(.) = \frac{1}{M} \sum_{k=1}^{M} h_k(.).$$

- Plurality Voting selects the class that has been predicted by a majority of the base-level classifiers as the final predicted class. There is a refinement of the plurality vote algorithm for the case where class probability distributions are predicted by the base-level classifiers. In this case, the probability distribution vectors returned by the base-level classifiers are summed to obtain the class probability distribution of the meta-level voting classifier:

$$P_{ML}(x) = \frac{1}{|C|} \sum_{c \in C} P_c(x).$$

- Stacking with ODTs is a meta-level classifier that uses the results of the base-level classifiers to predict which class the given instance belongs to. The inputs to the ODTs are the outputs of the base-level classifiers i.e. class probability distributions (CPDs) —$Pc_j(c_i|x)$, as predicted over all possible class values C_i, by each of the base-level classifiers C_j. The output of the stacked ODT is the class prediction for the given test instance.
- Stacking with MDTs learns a meta-level decision tree whose leaves consist of each of the base level classifiers. Thus, instead of specifying which class the given test instance belongs to, as in a stacked ODT, an MDT specifies which classifier should be used to optimally classify the instance. The MDTs are also induced by a meta-level data set that consists of the CPDs —$Pc_j(c_i|x)$.All the above meta-level classifiers are available in the Weka toolkit.

4. Simulation and presented experimental results:

Setting 1: Data collected with missing values that across all specifications random completely (MCAR) is distributed and cross-validated.
Setting 2: Dataset is used missing-free and cross-validated.
We did a 10-fold cross-validation for each of the classifiers in each of the above settings. In a 10-fold cross validation, the data is randomly divided into ten equal-sized pieces. Each piece is used as the test set with training done on remaining 90% of the data. The test results are then averaged over the ten cases.

Table 2 shows the classifier accuracies for the two settings respectively. It can be seen that Bagged NB performs the best in the first and second settings. In general, meta-level classifiers perform better than base level classifiers. The scatter-plot in Figure 3 shows the correlation in the performance of each classifier on Missing free data and MCAR model. Bagged NB has the best performance (95.24%).

Classifier	Accuracy (%)	
	MCAR model (5%)	Missing free
Naïve Bayes(NB)	95.24%	93.18%
Boosted NB	94.05%	92.05%
Bagged NB	95.24%	94.32%
SVM	*	40.91%
Boosted SVM	*	69.32%
Bagged SVM	*	55.68%
Decision Table(DT)	63.10%	79.55%
Boosted DT	66.67%	75%
Bagged DT	77.38%	77.27%
Decision Tree(DTr)	78.57%	73.86%
Boosted DTr	78.57%	72.73%
Bagged DTr	82.14%	80.68%
Plurality Voting	89.29%	86.36%
Stacking (MDTs)	80.95%	80.68%
Stacking (ODTs)	72.62%	71.59%

Table 2: Accuracy of classifiers for the two different settings

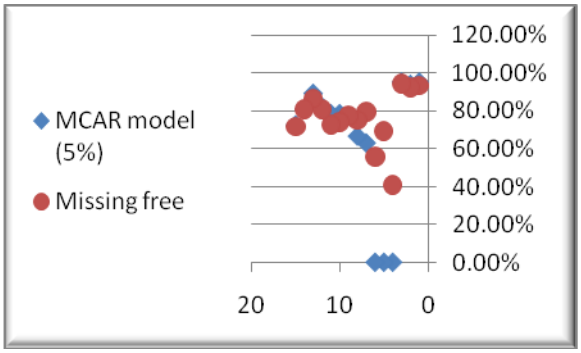

Figure3:Performance correlation for Missing free data and MCAR model

5. Conclusions and Future work

Robot failure prediction can be formulated as a classification problem. In this paper the performance of base-level classifiers and meta-level classifiers is compared. Bagged Naïve Bayes is found to perform consistently well across different settings, and so, we have shown the superiority of Bagged Naïve Bayes from all (95.24%). An interesting extension would be to check if the error and incompleteness on data be more than 5% what classifier would better and what preprocessing will be need.

References

Bao L. and Intille S.S. (2004), "Activity recognition from user annotated acceleration data. In Proceedings of the 2nd International," Conference on Pervasive Computing, 1–17.

Blake C.L. and Merz C.J. "UCI Repository of Machine Learning databases," [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Breiman L. (1996), "Bagging predictors," Machine Learning, 123–140.

Todorovski L. and Dzeroski S. (2003),"Combining classifiers with meta decision trees," Machine learning 223–249.

Chow E.Y. and Willsky A.S. (1984),"Analytical Redundancy and the Design of Robust Failure Detection Systems," IEEE Transactions on Automatic Control, 29 (7): 603-614.

Freund Y. and Schapire R.E. (1996),"Experiments with a new boosting algorithm," In International Conference on Machine Learning, 148–156.

Lopes S. and Camarinha-Matos L.M. (1998), "Feature Transformation Strategies for a Robot Learning Problem, Feature Extraction, Construction and Selection. A Data Mining Perspective", H. Liu and H. Motoda (edrs.), Kluwer Academic Publishers.

Merrill W.C., DeLaat J.C. and Bruton W.M. (1988), "Advanced Detection, Isolation, and Accommodation of Sensor Failures – Real Time Evaluation," Journal of Guidance, Control, and Dynamics, 11 (6): 517-526.

Meir R. and Ratsch G. (2003),"An introduction to boosting and leveraging," 118–183.

Seewald K. (2002), "How to make stacking better and faster while also taking care of an unknown weakness," In Proceedings of the Nineteenth International Conference on Machine Learning, 554–561. Morgan Kaufmann Publishers Inc.

Seo S., Kang J., Lee D., and Ryu K. H. (2006), "Multivariate Stream Data Classification Using Simple Text Classifiers," In Proceedings of DEXA 2006, LNCS 4080, pp. 420 – 429, 2006. Springer-Verlag Berlin Heidelberg.

Twala B. (2009), "Robot Execution Failure Prediction Using Incomplete Data," In Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics December 19 -23, Guilin, China.

Valavavins K.P., Jacobson C.A., and Gold B.H. (1991), "Integration Control and Failure Detection with Application to the Robot payload Variation Problem," Journal of Intelligent and Robotic Systems, 4: 145-173.

Witten I. and Frank E. (1999), "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations," Morgan Kauffman.

Wolpert D. H. (1992), "Stacked generalization," Neural Networks 241–259.

Gama J. and Brazdil P. (2000),"Cascade generalization," Machine Learning 315–343.