Prediction of Hard Drive Failures via Rule Discovery from AutoSupport Data

Vipul Agrawal AGRVIPUL@CSA.IISC.ERNET.IN
Intern, NetApp Advanced Technology Group, and
Dept. of Computer Science and Automation
Indian Institute of Science
Bangalore, India

Chiranjib Bhattacharyya CHIRU@CSA.IISC.ERNET.IN
Dept. of Computer Science and Automation
Indian Institute of Science
Bangalore, India

Thirumale Niranjan NIRANJAN.THIRUMALE@NETAPP.COM
NetApp Advanced Technology Group
Bangalore, India

Sai Susarla SAI.SUSARLA@NETAPP.COM
NetApp Advanced Technology Group
Bangalore, India

Editor:

Abstract

The ability to accurately predict an impending hard disk failure is important for reliable storage system design. The facility provided by most hard drive manufacturers, called S.M.A.R.T. (self-monitoring, analysis and reporting technology), has been shown by current research to have poor predictive value. The problem of finding alternatives to S.M.A.R.T. for predicting disk failure is an area of active research. In this paper, we present a rule discovery methodology, and show that it is possible to construct decision support systems that can detect such failures using information recorded in the AutoSupport database. We demonstrate the effectiveness of our system by evaluating it on disks that were returned to NetApp from the field. Our evaluation shows that our system can be tuned either to have a high failure detection rate (i.e., classify a bad disk as bad) or to have a low false alarm rate (i.e., not classify a good disk as bad). Further, our rule-based classifier generates rules that are intuitive and easy to understand, unlike black box techniques.

1. Introduction

Predicting the impending failure of hard disks in the field accurately can help storage systems or data center administrators to take corrective actions before the failure to avoid loss of data and performance degradation. The problem of predicting failures in hard disks is an old one, but there have not been any significant breakthroughs. A promising line of attack is to use machine learning techniques [9, 11, 13, 14]. The problem poses many hard challenges. First, we do not know which factors should be taken into consideration while studying disk failures. Second, monitoring data contains proprietary information and thus is difficult to acquire and not freely available.

Most of the hard drive vendors support self-monitoring, analysis and reporting technology (S.M.A.R.T.), which measures drive characteristics such as temperature, spin-up time, data error rates, etc. However, it has been found that S.M.A.R.T. parameters alone are not enough to reliably predict individual drive failures [15]. Also, S.M.A.R.T. is implemented differently by every drive manufacturer. Although it is an industry standard among most major hard drive manufacturers, the implementation is manufacturer-specific and much proprietary information is held by individual manufacturers as to their specific approach. Thus, S.M.A.R.T. is not correctly implemented on most systems due to lack of standards for handling of S.M.A.R.T. data. So, we incorporated many different types of disk events and errors generated by various layers in the storage and file system stack, referred to as "disk events" in the rest of the paper. These include read and write errors, checksum errors, RAID-level errors, and others. We also took into account the disk model, manufacturer, disk size and the interface. Not only do the types of disk events vary widely across different manufacturers, there are significant variations in the occurrence of events in different models by the same manufacturer [1, 2]. Also, the types of disk events varied on the type of interface, i.e., FC or SATA, used in the disks.

In addition to predicting the failure of hard drives, it is also necessary to keep the false alarm rate (FAR) to a minimum. High FAR can create an unnecessary panic for the owner as well as escalating the support and warranty costs incurred by storage array and hard drive vendors. To avoid this, NetApp offers a facility called the Maintenance Center that performs tests on the Controller itself. Maintenance Center offlines the disk and runs several diagnostic tests after a disk becomes erratic or its performance degrades significantly. However, these tests are intrusive and time-consuming, and are best kept to a minimum and applied as a last resort. Using the approach described in this paper, we can develop a system that can serve as a pre-filtering mechanism to drastically reduce the number of disks actually subjected to time-consuming diagnostics.

The basic problem is that of *classification*. An algorithm is trained with data to recognize the characteristics of two classes – "good disks" and "failing disks". In this paper, we describe an application of an existing machine learning approach called MLRules algorithm [7], which is based on a rule induction principle that can be effectively employed to detect impending disk failures. The MLRules algorithm generates an ensemble of classifiers to give an overall prediction of whether a disk will fail based on the hard disk events and warnings logged during the lifetime of the hard disk. It uses decision rules as its base classifier. A simple decision rule classifier contains a set of logical statements or conditions which, when satisfied, votes for a particular class. The MLRules algorithm sequentially generates decision rules with an associated weight factor, by greedily minimizing the negative log-likelihood to estimate the conditional class probability distribution. The details of the algorithm are presented in Appendix Section 6. Before generating new rules, it recalculates the class-conditional probability of instances according to the ensemble classifier constructed until that point.

For any technology to be readily accepted by the hard disk manufacturers and vendors, it must have two properties. It should be easily implementable, and the results must be trustworthy. Rule learning is thus a viable option. Rules can be learnt and calculated with low memory and computational requirements and can be used to develop an efficient stand-alone system for predicting hard disk failures with very good accuracy and low false alarm rate, unlike many other machine learning techniques. Rule-based techniques, since they are intuitively comprehensible, have special significance in case of hard disk failures. First, the interpretation can help in pruning insignificant rules from the classification model. Second, the rules can provide an insight into the disk events

that can be helpful in predicting failures, which could help in getting to the real cause of failures in hard disks.

Section 2 offers a survey of existing literature in this area. In Section 3, we give a brief background on rule-based learning. We describe our system and its experimental evaluation in Section 4. Finally, we outline our conclusions and avenues for future work in Section 5.

2. Related Work

Hard drive manufacturers have been developing self-monitoring technology in their products for predicting failures before it actually occurs, to allow users or storage systems enough time to backup their data. Currently, the Self-Monitoring and Reporting Technology (S.M.A.R.T.) system is used by nearly all hard drive manufacturers. It collects data such as temperature, spin-up time, data error rates, etc., during normal operation and uses them to set a failure prediction flag. The S.M.A.R.T. flag is a one-bit signal that is generated to warn users of impending drive failure. Nearly all hard drive manufacturers use a very naive threshold algorithm which triggers a S.M.A.R.T. flag when any attribute exceeds a predefined value. These thresholds are set so as to avoid false alarms at the expense of predictive accuracy. The thresholds may not be derived by statistical inference and based only on prior observations.

Past research [15] has found very little correlation between failure rates and either elevated temperature or activity levels. They show that some S.M.A.R.T. parameters (scan errors, reallocation counts, offline reallocation counts, and probational counts) can be helpful in predicting disk failure. However, due to the lack of occurrence of predictive S.M.A.R.T. signals on a large fraction of failed drives, they concluded that S.M.A.R.T. alone cannot be used to form an accurate predictive failure model.

Bairavasundaram et al. [1, 2] found that SATA disks and their adapters develop checksum mismatches an order of magnitude more often than FC disks. They also observed that the probability of developing checksum mismatches varies significantly across different disk models even within the same disk class. Also, the fraction of disks with latent sector errors varies significantly across manufacturers and disk models. They observed that as disk size increases, the fraction of disks with latent sector errors increases across all disk models. We use these observations to partition our dataset by model and interface, for better rule discovery.

Other approaches [9] have applied Bayesian methods to predict disk drive failures based on measurements of drive internal conditions. They used two methods. The first method posed it as the problem of anomaly detection. They applied a mixture model of naive Bayes clusters that is trained using expectation-maximization (NBEM). The second method was a naive Bayes classifier, a supervised learning approach. The dataset consisted of 1943 drives of which only 9 were marked as "will-fail". It achieved failed-disk detection rates of 35-40% for NBEM and 55% for naive Bayes classifier with low FAR.

Hughes et al. [11] proposed Wilcoxon rank sum statistical tests to improve failure warning accuracy and lower false alarm rates. It used Multivariate rank sum along with ORed rank sum and ORed threshold. Experimental data sets were obtained from drive design testing of 3744 drives of two different drive models with each set containing 2-3 months of reliability design test data. There were 36 verified drive failures. The highest accuracies achieved were modest (40%-60%) at 5% FAR.

Murray et al. [13] compared the performance of support vector machines(SVMs) [3], unsupervised clustering, and non-parametric statistical tests (rank-sum and reverse arrangements). They also proposed an algorithm based on the multiple-instance learning framework and the naive Bayesian classifier (mi-NB) [14]. Data from 369 drives was collected, and each drive was labeled good or failed. Drives labeled as good were from a reliability demonstration test, run in a controlled environment by the manufacturer whereas the ones labeled as failed were returned to the manufacturer from users after a failure. There were only 191 failed disks in the dataset. It showed about 50% detection rates at 0% false alarm rates.

Our results, presented here, are much better than the ones seen in the above referenced literature.

3. A Review of Rule-based Learning

Decision rules are logical statements of the form, *if condition then response*. Rules can be treated as simple classifiers that votes for some class when the conditions are satisfied and abstains from vote otherwise. They can be considered as a form of decision stump. Example:

if
  feature1 $\leq$ a and
  feature2 $\geq$ b and
  feature3 = c
then class = +1

There are many rule learning procedures [7, 8, 6, 17] to learn a compendium of rules from empirical data. A comparative review of these approaches are beyond the scope of the paper. Instead we present a short description of a general procedure followed in most rule learning algorithms known as sequential covering procedure or separate-and-conquer approach. In this technique, a rule is learned which covers a part of the training examples. Then, all the covered examples are removed from the training set and the process is repeated until no examples remain. The sequential covering algorithm is shown in Algorithm 1.

In this paper, we use a state of the art rule learning algorithm called MLRules Algorithm [7].

Algorithm 1 Sequential Covering Procedure

For each class C:
  Initialize to the set of all examples E
  While E contains examples of class C
    Create a rule R that predicts class C
    Until R is 100% accurate or no more attributes do:
      For each attribute A $\notin$ R and each value 'v'
        Consider adding condition A=v to R
        Select A-v pair to maximize accuracy and
        covering
      Add A=v to R
    Remove examples covered by R from E

4. Deriving Rules from Disk events

In this section we discuss the application of MLRules algorithm for computing a rule based classifier from disk events.

4.1 Data Collection

Netapp storage controllers have a built-in, low overhead mechanism called Autosupport to log important system events back to a central repository. These messages, when enabled, report a variety of system events including disk errors. Due to the sheer size of such a database, which contains logs for about four thousand different events, we were forced to use the data selectively. Learning which subset is best for prediction is itself a difficult problem and is beyond the scope of this paper. We selected a few disk events based on prior knowledge such as media errors, checksum errors, parity errors etc. We also carried out a rigorous search on other potential events and selected those that had a significant number of occurrences for the hard disks in consideration. In view of observations [1, 2], we also added the details of disks such as disk model, disk size, manufacturer and the interface. The NetApp Autosupport Database has also been used in disk failure [12], latent sector error [1] and data corruption studies [2].

The data collection started with taking all the disks present in the RMA database, which is the database that contains information about the disks that were returned from the field because they were flagged as faulty either by ONTAP alone or also by Maintenance Center. Using this database, we got the disk serial number and the RMA test results. Many of these disks were actually found to have no problems at all (NTF or "no trouble found"). We used the RMA test results to assign the class label for each disk. There were 650K disks in the RMA database. All these disks were searched for in the ASUP database. Unfortunately, there were records of only around 128K disks. This could be due to a recent restriction on keeping only last 2 years of data in the ASUP database. Another large portion of disks had to be removed since there were duplicate entries for a single disk serial number. The disk serial number is not unique for a particular disk. Instead, another number named 'Device ID' is unique across NetApp, but the RMA database only contains disk serial numbers. We were left with about 86K disks.

We queried around 1000 tables related to disk events found in ASUP database. Out of these, only 75 were found to have a significant number of event logs present for the given set of disks. We used these 75 events as the final set of events to get data for. In spite of considering such a large set of events, many disks were found to have not even a single event log for any of them. We removed such disks from consideration. We were left with about 40K disks. We then limited our disks to FC or SATA. The total number of disks in the resulting dataset was 28877.

4.2 Experiments

For our evaluation, we extracted the disk events data of 28877 disks returned from the field due to complaints from the customer. Out of these, 25048 were found to be good after the in-house tests and 3828 failed the test. In spite of having nearly 75 different types of disk events, most of the disks showed relatively very few events, leaving our data set quite sparse. Most of the prevalent machine learning techniques failed to give even modest results. We trained the classifier based on the known failure data of some of the disks and used the rules to predict other disk failures in the set. For

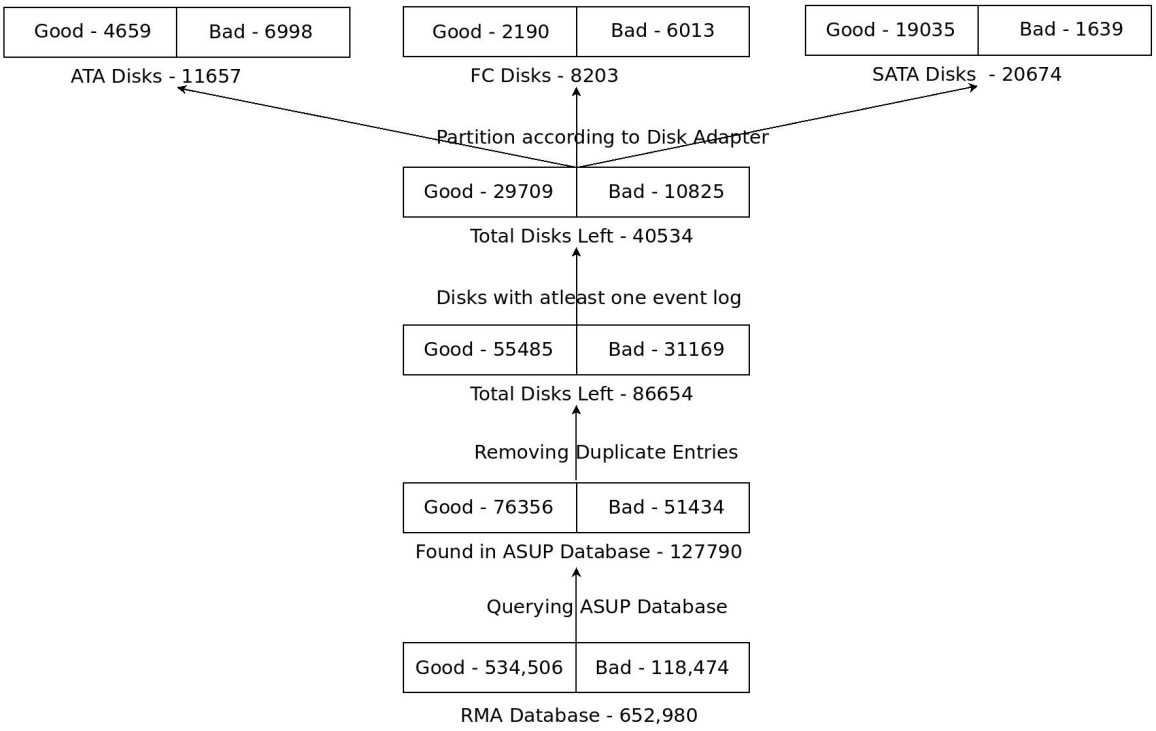

Figure 1: Steps involved in collection of data

the purpose of comparison, we also applied MLRules algorithm to the dataset used by Murray et al. [14].

4.3 Dataset

For the purpose of creating a dataset for use with MLRules algorithm, we calculated the cumulative disk events for every disk that occurred in its lifetime. We created a matrix of dimension N x M, where N is the number of disks in the dataset and M is the number disk events under consideration. The matrix thus formed was such that each row contained a particular disk's data with disk events forming the columns. We found this matrix to be very sparse, which severely affected the results. We explored the effect of adding disk details to the dataset. For this, we simply added more columns to the existing matrix of disk events. We also partitioned the dataset based on the interface, i.e. FC and SATA, to see whether accuracy could be increased and whether the generated rules showed any marked difference.

All the previous work regarding hard disk failure prediction have been on relatively very small datasets consisting of a few hundred failed disks. The data collected was not totally field data but consisted of data collected during testing in uniform controlled environment using the S.M.A.R.T. features only. For our data we relied only on the field data that was collected when the disks were in actual use. The attributes were also very different consisting of read and write errors, checksum errors, RAID-level errors and also the disk model, manufacturer, disk size etc. It did not consist of attributes like temperature, flyheight etc.

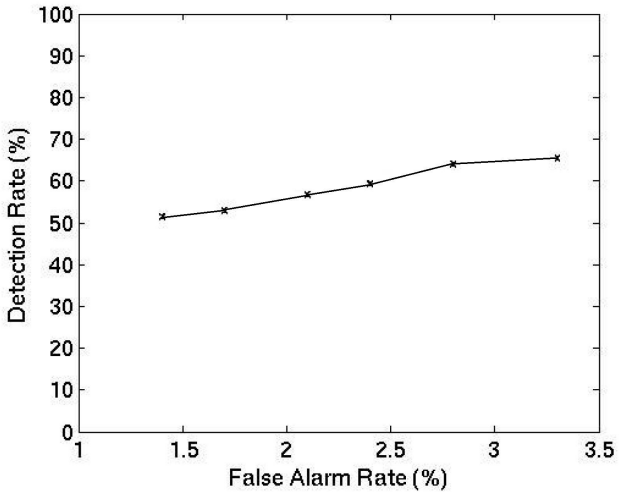

Figure 2: Graph of Detection rate vs FAR using MLRules algorithm. The points represent number of rules generated (15, 20, 30, 40, 50, 100 from left to right)

Murray et al. [14] used SVM with radial basis kernel applying Multiple-Instance learning technique on their dataset. We cannot apply their technique to our dataset since 1) Our dataset is really big in comparison to theirs (about order of 2 times the magnitude). 2) our dataset is not a time-series to apply MI technique. Even if we make our data a time series, the resulting dataset would be large enough to make SVMs inappropriate for this kind of problem domain which prefers minimal computational and memory requirements.

We also applied MLRules algorithm to the dataset used by Murray et al. [14]. Since it is time-series data, we cannot directly apply MLRules algorithm to it. Instead, we calculated the cumulative sum of attributes that were given for a particular time interval such as read and write errors. Other attributes such as temperature, fly height etc. were removed since neither we can take their sum nor select one particular value with confidence.

4.4 Results

The dataset was divided into 10 folds, where the folds are the disjoint sets generated by partitioning the data. For every iteration, one of the folds was chosen as the test set one by one and the rest formed the training set. This whole procedure was repeated 10 times and the results averaged. The results are shown in Figure 2. *Detection rate* is defined as the number of failed disks that were classified correctly as failing. *False Alarm Rate(FAR)* is defined as the number of good disks that were classified as failing. Using the MLRules algorithm, we could predict the failure of around 66% of the total number of disks (28877) with only 3% false alarm rate(FAR) by generating 100 rules. We can tradeoff better FAR for worse detection rate. This tradeoff between detection rate and FAR can be realized by changing the number of rules generated. For example, the detection rate falls to 60% at 2.5% FAR using 40 rules and 51% at 1.4% FAR using just 15 rules.

The detection rate varies with the number of rules generated as shown in Figure 3. Initially, the detection rate increases with increase in number of rules. This is due to the fact that the additional

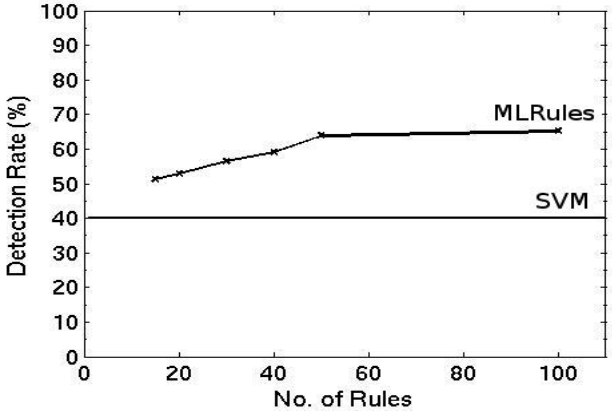

Figure 3: Variation of detection rate with number of rules

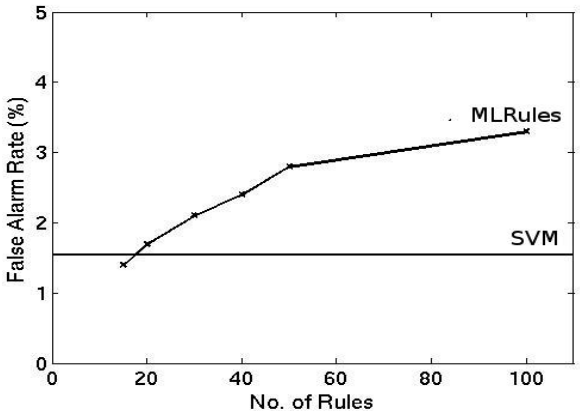

Figure 4: Variation of false alarm rate with number of rules

rules generated provide better identification of failed disks. However, the false alarm rate steadily increases with number of rules as shown in Figure 4. We conjecture that that with an increase in the number of rules, there is a greater stress on correct identification of failed disks, which may lead to an increasing amount of good disks being classified as failed. The accuracy of classification of all disks, both failed and good, being correctly classified as shown in Figure 5. Here, we find that the performance steadily increases and then begins to degrade due to overfitting. It also indicates that a small number of rules are sufficient to get the best results for such a system.

To benchmark the MLRules algorithm against other classifiers, we implemented an SVM with a Gaussian kernel. We used cross validation to find the appropriate parameters. We achieved detection rates of range 20-40% only at 1.5-2.5% FAR. Although it can achieve a good FAR, the detection rate was not satisfactory. We believe that to obtain accuracy comparable to our system, we would have to design kernels that are more suited for this purpose than the Gaussian kernel. This will be taken up in future work. However, even if the accuracies are comparable, SVMs still suffer from the lack of intuitive interpretability that the rule learning framework offers.

Also, on the disk dataset analyzed by Murray et al. [14], which consists of a total of 369 disks of

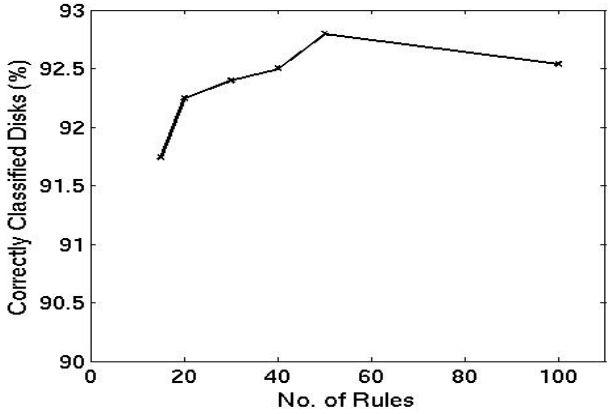

Figure 5: Variation of percentage of correctly classified disks with number of rules

which 178 are good and 191 are bad disks, we could get about 96% detection rate at 3% false alarm rate. In contrast, the paper's best results using SVMs could only get around 60% detection at 3% FAR. However, we coud not manage to bring the FAR below 3%, which may be due to removal of incompatible attributes. The dataset under consideration was time series data and some attributes could not be included for applying MLRules on it.

4.4.1 Partitioning Based on Interface

We partitioned the dataset according to their interface, i.e., Fibre channel (FC) and SATA. There were 6013 good disks and 2190 bad disks with an FC interface and 19035 good disks and 1639 bad disks with a SATA interface. The results are shown in Table 1. For SATA disks, we could predict the failure of around 70% of them with only 1% FAR. For FC, we could predict the failure of around 73% of disks but only at 8% FAR. The reason for this could be that although the set of disk events was able to capture the conditions of failing hard disks, it was more suitable for prediction for disks using SATA than FC.

4.5 Improving Detection Rates

By varying the number of rules in the ensemble, we could achieve extremely low false alarm rates with good detection rates. This type of model can be used in a scenario where identification of good disks is much more important. However, in the process, we would classify some failed disks also as good. It could be deployed to reduce the number of disks being RMAed. In some cases, we need to have very high detection rates and compromise on false alarm rates. If such a classifier could be

Table 1: Detection rate and FAR after partitioning data based on adapter

Adapter	Detection Rate	FAR
SATA	70%	1%
FC	73%	8%

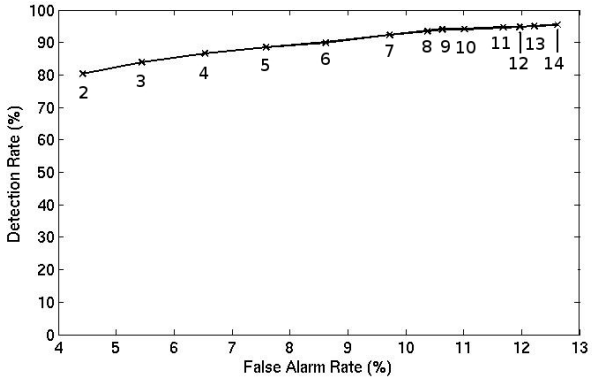

Figure 6: Graph of Detection rate vs FAR using MLRules algorithm after duplicating the data from failed disks. The annotation points on the line represent the number of times the data is duplicated (2 to 14 from left to right). The ratio of good disk to failed disks was 7:1 in the original dataset.

generated, then it could be deployed just before the Maintenance Center (MC) to reduce the number of disks passing through it. We could not further improve the detection rates just by varying the number of rules as shown in Figure 3. The detection rate tends to stabilize after a certain point. However, the false alarm rates keeps on increasing with increase in number of rules as shown in Figure 4.

We found that the MLRules algorithm behaved in a peculiar manner. It was found to be sensitive to imbalanced datasets, i.e., datasets in which the number of examples of one class are very less in comparison to the other class. The set of rules generated were found to, generally, favour the class with majority representation in the training set. This led to better identification of good disks and thus, low false alarm rates, since our dataset contained mostly good disks. Ideally, the best performance is seen if each class is equally represented in the training set. To increase the detection rates, we tried to duplicate the data from failed disks ("oversample") to make it comparable in number with the good disks. This technique is common when dealing with imbalanced datasets [5, 4, 18, 16]. After running our experiments on this new dataset we found that the detection rates were considerably increased with little increase in false alarm rates. The results are shown in Figure 6. The number of good disks was 7 times the number of failed disks in the original dataset. As we can see, the detection rates significantly improved and reached as high as 95%. The detection rate seem to stablize as the dataset becomes balanced in terms of good and failed disks although false alarm rate keeps on increasing.

4.5.1 Partitioning Based on Disk Model

We attempted to create individual rule sets for each of the 51 disk models in our dataset. Although only a few disk models had enough good and bad samples to be significant for prediction, we found that the detection rates and false alarm rates were so poor that it made little sense generating rules for individual models. This was due to the fact that the individual disk model datasets were highly skewed and would result in all disks being classified as either good or failed. We used the same

Rule:

INTERFACE is SATA
MEDIA_ERR_TYPE1 $\leq$ 2.5
IO_REASSIGN_SUCCESS $\leq$ 77.0
MEDIA_ERR_TYPE2 $\leq$ 0.5
DISK_SIZE $\geq$ 285.0
IO_RECOVERED_ERROR $\leq$ 14.5
MEDIA_ERR_TYPE3 $\leq$ 133.0
REWRITE_DATA_FAILED $\leq$ 6.5

=> vote for class **GOOD**
with weight 0.08922562491818498

Figure 7: A sample rule generated by MLRules

technique of duplicating data as described before to make the number of good and failed disks comparable. The results are shown in Table 2. This result is significant because we can generate disk model specific classifiers that may be better suited for the task of identifying good and failed disks.

Table 2: Detection rate and FAR after partitioning data based on disk model

Disk Model	Detection Rate	FAR
X276_FAL9E288F10	98.4%	38%
X274_FAL9E146F10	90.7%	43%
X261_MTOMC250P05	96%	21%

4.6 Interpretable Rules

A sample rule generated by MLRules algorithm is shown in Figure 7. A collection of such rules form the rule ensemble. During classification, for each data instance, all the rules are progressively checked for satisfiability. If a particular rule is found to be satisfied, its weight is added to the class label it predicts. In the end, the class label with largest value is assigned to the given instance.

Rule-based techniques have a great advantage. The generated rules are easy to understand. For example, the rule shown in Figure 7 can be interpreted as *"If the given disk uses SATA interface, its size is greater than 285 GB, and the disk error and event counts are below stated thresholds then the disk can be said to be working fine with some probability"*. Such an interpretation is not possible in case of other machine learning techniques. The interpretation can also help in pruning insignificant rules from the classification model. Intuitively, a rule that is voting for class *good* should not contain *greater than* conditions for disk events, i.e. a disk is *good* if some event count exceeds a threshold. Such rules can be pruned from the final model.

Using disk events instead of just S.M.A.R.T. attributes make the rules meaningful and increase their information content. Rules can provide an insight about which disk events can be helpful in predicting failures. This could help in getting to the real cause of failures in hard disks which can help in improving the reliability of storage systems. We found that I/O recovered errors, events related to rewriting data, checksum errors, media errors, and transport errors were present in significant number of rules. This implies that such events have more predictive capabilities than others. Table 3 shows the list of events which had a sizable number of occurrences in rules.

5. Conclusions and Future work

We have shown that the disk events captured in AutoSupport have good value in predicting impending disk failure. Our results also indicate that by using disk events instead of just S.M.A.R.T. attributes, we have significantly increased the prediction accuracy while keeping the false alarm rates to a minimum.

We have also shown that our rule-based classifier outperforms the existing techniques for predicting impending hard disk failures. Generating rules is computationally inexpensive and less time-consuming. The generated rules are meaningful and intuitive. These characteristics make rule-based learning techniques much more suitable for hard disk failure prediction.

However, this work can be improved in many ways. First, we have only used the cumulative error counts in hard disks for prediction of impending failures. Intuitively, hard disk failure seems to be of time-evolving nature. We plan to incorporate the time dimension also, which could improve the prediction accuracy and lower false alarms. Second, we have to study how to completely automate the process of which attributes to consider for generating the rules. The problem here is that the database is very large and sparse, and pruning the attribute set is a difficult task.

We are building a prototype, incorporating this system into the ONTAP workflow just before Maintenance Center is triggered. Our goal is to reduce the number of good disks entering Maintenance Center. We are also trying to use the MLRules method to predict and suggest which of the Maintenance Center tests to run, given the disk events.

6. Appendix: MLRules Algorithm

MLRules stands for *maximum likelihood rules*. The main idea of the MLRules algorithm consists of inducing rules by greedily minimizing the negative log-likelihood to estimate the conditional class probability distribution $P(y|x)$ by which we can measure the prediction confidence. The algorithm is shown in Algorithm 2. It considers the estimation of probabilities using the maximum likelihood estimation (MLE) method which minimizes the empirical risk by taking the negative log-likelihood as the loss function.

$$l = \sum_{i=1}^{n} -logP(y_i|x_i) \tag{1}$$

The K class probabilities $P(1|x), ..., P(K|x)$ is represented by a vector f$(x) = (f_1(x), ..., f_K(x))$ using logistic transform:

$$P(y|x) = \frac{e^{f_y(x)}}{\sum_{k=1}^{K} e^{f_k(x)}} \tag{2}$$

Table 3: Number of occurrences of disk events in 100 rules

Event	Number of occurrences
Rewrite Data	71
I/O Recovered Error	63
Checksum Error	54
Transport Error	53
Media Error	49

The negative log-likelihood of Equation 1 becomes:

$$l(\mathrm{f}) = \sum_{i=1}^{n} log \left(\sum_{k=1}^{K} e^{f_k(x_i)} \right) - f_{y_i}(x_i) \tag{3}$$

The loss function is then:

$$L(y_i, \mathrm{f}(x_i)) = log \left(\sum_{k=1}^{K} e^{f_k(x_i)} \right) - f_{y_i}(x_i) \tag{4}$$

6.1 Construction of Rules

Let X_j be the set of all possible values of attribute j. *Condition* is conjunction of elementary expressions of the form $x_j \in S_j$ where:

- x_j is the value of example x on attribute j and
- S_j is a subset of X_j

Elementary expressions are of the form $x_j \geq s_j$ or $x_j \leq s_j$. In case of nominal attributes, the elementary expressions can be constructed as $x_j = s_j$ or $x_j \neq s_j$. Let $\Phi(x) = 1$ if x satisfies the *condition* otherwise $\Phi(x) = 0$. The rule output is a vector $\alpha \in R^K$. The decision rule can be written as

$$r(x) = \alpha\Phi(x) \tag{5}$$

The classification function is a linear combination of M rules

$$\mathrm{f}(x) = \sum_{m=1}^{M} r_m(x) \tag{6}$$

Algorithm 2 MLRules

---

input: Set of n training examples
M - number of decision rules
output: rule ensemble R

for m = 1 to M do
Find the decision rule minimizing the loss function
Calculate the rule weight
Add new rule to the existing ensemble

Construction of optimal rules is a hard optimization problem. The algorithm uses a forward stage-wise strategy [10]. The rules are added one by one, greedily minimizing the loss function. Still the problem remains computationally hard. To overcome this, it restricts to the rules voting for only one class: $\alpha = \alpha v$, where $\alpha \in R_+$ and $v = (0, ..., 0, 1, 0, ..., 0)$.

References

[1] Lakshmi N. Bairavasundaram, Garth R. Goodson, Shankar Pasupathy, and Jiri Schindler. An analysis of latent sector errors in disk drives. In *Proceedings of the 2007 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 2007.

[2] Lakshmi N. Bairavasundaram, Garth R. Goodson, Bianca Schroeder, Andrea C. Arpaci-dusseau, and Remzi H. Arpaci-dusseau. An analysis of data corruption in the storage stack. In *Proceedings of the 6th USENIX Symposium on File and Storage Technologies (FAST 08)*, 2008.

[3] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, volume 2:121–167, 1998.

[4] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16: 321–357, 2002.

[5] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets*, 6(1):1–6, 2004.

[6] William W. Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 335–342, 1999.

[7] Krzysztof Dembczynski, Wojciech Kotlowski, and Roman Slowinski. Maximum likelihood rule ensembles. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[8] Jerome H. Friedman and Bogdan E. Popescu. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2:916, 2008.

[9] Greg Hamerly and Charles Elkan. Bayesian approaches to failure prediction for disk drives. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001. ISBN 0387952845.

[11] Gordon F. Hughes, Joseph F. Murray, Kenneth Kreutz-Delgado, and Charles Elkan. Improved disk-drive failure warnings. *IEEE Transactions on Reliability*, 2002.

[12] Weihang Jiang, Chongfeng Hu, Yuanyuan Zhou, and Arkady Kanevsky. Are disks the dominant contributor for storage failures? a comprehensive study of failure characteristics. In *Proceedings of the 6th USENIX Symposium on File and Storage Technologies (FAST 08)*, 2008.

[13] Joseph F. Murray, Gordon F. Hughes, and Kenneth Kreutz-Delgado. Hard drive failure prediction usig non-parametric statistical methods. In *Proceedings of ICANN/ICONIP*, 2003.

[14] Joseph F. Murray, Gordon F. Hughes, and Kenneth Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning research*, volume 6, 2005.

[15] Eduardo Pinheiro, Wolf Dietrich Weber, and Luiz André Barroso. Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Symposium on File and Storage Technologies (FAST 07)*, 2007.

[16] Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12), 2007.

[17] Sholom M. Weiss and Nitin Indurkhya. Lightweight rule induction. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[18] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Trans. on Knowl. and Data Eng.*, 18(1), 2006.