Bayesian approaches to failure prediction for disk drives

Greg Hamerly GHAMERLY@CS.UCSD.EDU
Charles Elkan ELKAN@CS.UCSD.EDU
Department of Computer Science and Engineering, University of California, San Diego, La Jolla, California 92093-0114

Abstract

Hard disk drive failures are rare but are often costly. The ability to predict failures is important to consumers, drive manufacturers, and computer system manufacturers alike. In this paper we investigate the abilities of two Bayesian methods to predict disk drive failures based on measurements of drive internal conditions. We first view the problem from an anomaly detection stance. We introduce a mixture model of naive Bayes submodels (i.e. clusters) that is trained using expectation-maximization. The second method is a naive Bayes classifier, a supervised learning approach. Both methods are tested on real-world data concerning 1936 drives. The predictive accuracy of both algorithms is far higher than the accuracy of thresholding methods used in the disk drive industry today.

1. Introduction

In the computer industry, failures of hard disk drives are rare but costly. For a typical model of disk drive, the probability of failure is less than 1% per year. Low failure rates are of course desirable, but they make collecting data about failures and predicting failures difficult. Since the 1980s, most disk drives have had the ability to report information about their internal conditions (Nass, 1995). However this information is not being used by most system manufacturers to assist in failure prediction. The manufacturers that have developed failure prediction methods rely on excessively simple methods, in particular thresholding of single drive attributes. These methods generate many more false alarms than they do correct failure predictions.

In this paper, we describe two learning approaches that use Bayesian methods to predict hard disk drive failures. First we pose the problem from an anomaly detection viewpoint, for which we introduce a mixture of naive Bayes submodels. This model estimates the probability distribution of readings from drives behaving normally, and is trained using expectation-maximization. The second method is a standard supervised naive Bayes classifier. We test our models on a dataset of readings from 1927 disk drives that never fail and 9 drives that do fail.

2. Anomaly detection algorithm

Consider the problem of labeling data with one of two classes, where one class is very rare. This is the situation in disk drive failure detection. For the rare class, there may not be enough data available to allow a supervised learning algorithm to estimate a good probability model for that class. Additionally, data from the rare class may be incomplete due to collection problems. For example, if historical data concerning drive conditions is stored inside a disk drive, this data is available for drives that are functioning correctly, but not for drives that have failed.

An alternative to supervised learning that overcomes these problems is to build a probabilistic model of the majority class and then to use an anomaly detection approach to classify test data. An anomaly detection algorithm attempts to classify data as normal or anomalous based on a probability model of normal behavior. Anomaly detection has been widely studied in computer and network security research (Lee & Stolfo, 1998).

The term data point in this paper refers to a single example presented to a learning algorithm. For example, a data point may be a summary of the behavior of a hard drive for the last hour. An anomaly is a data point to which the majority class model assigns a very low probability of occurring. From a machine learning standpoint, the model parameters may be learned from training data concerning normal behavior (Lane & Brodley, 1997; Towell, 2000). A model parameter is an unknown (but learnable) quantity that is part of the model, e.g. the mean of a Gaussian distribution.

Anomalous/normal classification is similar to positive/negative classification. However, the anomaly detection approach differs because it builds a probability model for only the normal class of data, and not for the anomalous class. An anomaly detection algorithm can learn its parameters in two ways:

- In a fully unsupervised fashion, learning from both normal and anomalous data. This method assumes that the anomalous data is rare enough to not affect the model parameters significantly (Eskin, 2000).
- In a semi-supervised fashion, learning a model from only the normal data and removing data from the anomalous class from training.

In general semi-supervised training is preferable since it should learn the most accurate model for the normal class. Unsupervised training is useful if training data is not labeled at all and the assumption is true that anomalous data is rare in the training set.

Our method of constructing an anomaly detector is as follows. First, choose a probabilistic model $f(\cdot)$ for the data which takes a data point as input and outputs a probability for that data point. Then learn the parameters for f from training data in either a semi-supervised or unsupervised fashion. By choosing a probability threshold t, the model can be used to classify a data point x as typical or anomalous as follows:

$$x \text{ is anomalous iff } f(x) < t.$$

In supervised learning, a data point is labeled with the class that has the highest posterior probability of generating the data point. In contrast, in anomaly detection a data point is labeled "normal" or "anomalous" depending on its probability of occurrence as a normal data point. The threshold value t may be chosen by hand, based on an assumption about the rarity of anomalous data, or learned experimentally if data known to be anomalous is available.

3. Mixtures of naive Bayes submodels

For the anomaly detection task, we use a model that is a mixture of naive Bayes submodels, i.e. clusters. This mixture can be thought of as a clustering model intended to represent a multimodal probability distribution. The model is trained on data using the expectation-maximization (EM) algorithm, so we call this approach the NBEM method. The NBEM algorithm is similar to the AutoClass method of Cheeseman and Stutz (1995). AutoClass is a general framework for mixture model clustering. Like NBEM, AutoClass uses maximum likelihood and EM to learn model parameters. AutoClass also attempts to learn the optimal number of submodels, while NBEM lets this number be a parameter chosen by the user.

3.1 Model overview

Each naive Bayes submodel is itself a nonparametric data model. The word nonparametric does not mean there are no parameters to learn, but that no single simple parameterized probability density (e.g. Gaussian) is assumed. The probability distribution of each naive Bayes submodel is determined from the frequency counts of the training data. Each submodel has an estimated prior probability $P(k)$ where k is the submodel index. The probability estimate that NBEM assigns a data point $\mathbf{x}$ is

$$P(\mathbf{x}) = \sum_{k} P(\mathbf{x}|k)P(k). \tag{1}$$

The term $P(\mathbf{x}|k)$ is the probability that the submodel k generates the data point x and is estimated via training data frequencies.

From equation (1) it is clear that all submodels contribute to the probability of each data point $\mathbf{x}$. Conceptually, data points are assigned to submodels in a "hard" way, because each data point is generated by exactly one submodel. However, different submodels may generate identical data points, so for each data point, it is unknown which submodel actually generated it. NBEM therefore assigns each data point $\mathbf{x}$ to each submodel k in a "soft" way, with proportion $P(k|\mathbf{x})$. This quantity is determined from $P(\mathbf{x}|k)$ and Bayes' rule. Soft assignment allows the model to fit the data better when submodel memberships are unknown because it blurs the manifold that separates submodels.

The naive assumption in naive Bayes is that the attributes of a data point are conditionally independent within each submodel, i.e. that

$$P(x_i, x_j|k) = P(x_i|k)P(x_j|k)$$

where x_i and x_j such that $j \neq i$ are attributes of $\mathbf{x}$. While typically not true, this assumption simplifies probability estimation and often works well in practice (Lewis, 1998). For a d-dimensional data point and a given class k we have

$$P(\mathbf{x}|k) = \prod_{i=1}^{d} P(x_i|k). \tag{2}$$

We use discrete naive Bayes submodels. The values of each attribute are placed in a small finite number of bins. A bin is an internal value used in place of the true value of an attribute. For discrete data, each value gets a separate bin, so the transformation is merely a re-mapping. For continuous data, naive Bayes divides the data range into b bins, and translates each continuous value to its bin number. Our naive Bayes EM uses equal-width binning, where each bin represents an interval of the same size in the input space. The value of b is an input parameter.

Although binning causes some loss of information for continuous data, it allows smooth probability estimates. Binning has two further advantages: it allows the same attribute to take both continuous and discrete values, and it compresses data by replacing continuous values by small integers representing bin numbers.

Naive Bayes EM estimates the probability that an attribute x_i will have a given value v based on training data frequencies:

$$P(x_i = v|k) = \frac{\text{count}(x_i = v \wedge k)}{\text{count}(k)} \quad (3)$$

$$P(k) = \frac{\text{count}(k)}{\sum_k \text{count}(k)} \quad (4)$$

where count(α) is the number of data points in the training dataset which satisfy the predicate α. Since we are using soft assignment, a data point may contribute only a fraction to the count. The formal definition of count is

$$\begin{aligned} \text{count}(x_i = v \wedge k) &= \sum_{\mathbf{x}} P(k|\mathbf{x}) I(x_i = v) \\ \text{count}(k) &= \sum_{\mathbf{x}} P(k|\mathbf{x}) \end{aligned}$$

where $I(\alpha)$ is the indicator function conditioned on the predicate α, and the sum ranges over the entire training set.

Individual naive Bayes models do not define unimodal distributions, unlike the familiar exponential family. Instead, a naive Bayes model may have several regions of high probability. Compared to clustering algorithms based on unimodal distributions such as Gaussians, NBEM may produce surprising results. However, when the true distribution is unknown, multimodality may be preferable. For disk drive modeling, it is not immediately obvious where a unimodal distribution should associate its region of highest probability.

While we give up some flexibility in our model with the naive assumption of class-conditional independence, we gain flexibility through the choice of the number of submodels. For example, one naive Bayes model cannot represent the xor concept, which requires dimensional dependence, but a mixture of multiple submodels can.

3.2 Training with expectation-maximization

While there are multiple submodels in NBEM, it is not known which submodel each data point truly belongs to, and it may not be known how many submodels are appropriate. The submodel membership for each point is missing information, so we use the standard expectation-maximization (EM) algorithm (Dempster et al., 1977) to maximize the likelihood of the data given the model. To avoid underflow in floating point calculations and for mathematical convenience, we maximize the logarithm of the likelihood:

$$\mathcal{L} = \log \prod_{\mathbf{x}} P(\mathbf{x}) = \sum_{\mathbf{x}} \log P(\mathbf{x}). \quad (5)$$

To train the model, we first choose the number of submodels k and the number of continuous bins b. After the model is initialized by assigning points randomly to submodels, EM training determines the submodel probability $P(k)$ and the data probability conditioned on a submodel $P(\mathbf{x}|k)$. These allow us to apply equation (1).

The EM training proceeds in rounds of an expectation step followed by a maximization step. The E-step computes for each data point the probability of each submodel producing that point. This probability is used to soft-assign the data point partially to each submodel and is computed using Bayes' rule as

$$P(k|\mathbf{x}) = \frac{P(\mathbf{x}|k)P(k)}{P(\mathbf{x})}$$

where the right side comes from equations (2) and (1).

Once the E-step has computed estimated memberships, the M-step updates the model parameters $P(x_i|k)$ and $P(k)$ using equations (3) and (4). The M-step maximizes the log-likelihood of the model given the expected membership information. The EM steps repeat until the log-likelihood as defined in equation (5) of the data does not change.

3.3 Implementation issues

Because a naive Bayes model estimates probabilities based on training data that may be noisy and incomplete, we smooth probabilities in two ways. As mentioned above, continuous attribute values are placed in bins so probabilities are estimated for intervals, not individual continuous values.

The second method of probability smoothing is needed to deal with the problem of zero counts. A zero count exists when bin v for attribute i contains no data points in the training set. Then the probability estimate $P(x_i = v|k) = 0$, which causes equation (2) to yield zero regardless of the other dimension probabilities. Zero probabilities prevent generalization and cause instability in EM training. Instability here refers to how inconsistent the model output is when trained multiple times under slightly varying initial conditions, as created for example by cross-validation.

We address the problem of zero counts by adding artificial counts to all bins (Ristad, 1995). This has the effect of increasing low probabilities, and decreasing high probabilities. A popular method of zero count smoothing is Lidstone's law of succession (Lidstone, 1920):

$$P(x|k) = \frac{n_x + \lambda}{n + v\lambda}.$$

Here n_x is the number of data points in submodel k with value x, v is the number of values x may take, n is the total number of data points in submodel k, and λ is the smoothing parameter. Experimental evidence suggests that we should choose λ to be small but non-zero (Kohavi et al., 1997).

Following the results of Kohavi *et al.*, we choose $\lambda = 1/m$ where m is the total number of training data points. Then our probability estimate in equation (3) becomes

$$\hat{P}(x_i|k) = \frac{\text{count}(x_i \wedge k) + 1/m}{\text{count}(k) + b/m}$$

where b is the total number of bins. NBEM models converge to much more consistent results when using this arithmetic smoothing procedure.

4. Supervised naive Bayes learning

We also explore the disk drive dataset using a second method, a naive Bayes classifier. Naive Bayes learning is a well-known supervised learning method that yields a classifier distinguishing between each class of data. For further information on the standard naive Bayes classifier, the reader is referred to (Lewis, 1998). We have also applied boosting to the standard classifier to obtain an ensemble of voting classifiers (Freund, 1995; Elkan, 1997), but for this application boosting produces no appreciable improvement in results.

As with naive Bayes EM, with the naive Bayes classifier we place continuous attribute values into bins. However, for the supervised naive Bayes method we use equal-frequency bins rather than equal-width bins, placing approximately the same number of data points in each bin. Experiments show that results are approximately the same with equal-width and with equal-frequency bins. Results are also not sensitive to the number of bins used, in a range from four to twenty. One major advantage of naive Bayes classifiers is that they are simple enough to be implemented easily inside a disk drive using a field-programmable gate array (FPGA) or other hardware.

5. Quantum SMART dataset

The data we use comes from real-world collection by Quantum, Inc. Modern disk drives incorporate so-called SMART (self-monitoring and reporting technology) firmware which allows a disk drive to report data about its activity, such as the number of hours it has been in operation, the number of seek errors that have occurred and been corrected, etc. More information about SMART can be found in (Nass, 1995).

Table 1 lists the SMART attributes available in the Quantum dataset. SMART attributes are vendor- and drive-specific, but the interface protocol for reading their values is standardized. The standard industry procedure for warning of a potential failure based on SMART attribute values is to use a separate threshold for each attribute. These thresholds are chosen based on engineering knowledge about operating parameters, not based on empirical data analysis. A warning is supposed to be issued if any attribute surpasses its fixed threshold. In practice, SMART data is usually ignored completely because the number of warnings that are false alarms would be unacceptably high.

Table 1. SMART attributes used in the Quantum dataset.

Abbreviation	Description
RET	read error rate
SUT	spinup time
CSS	start-stop count
GDC	grown defects count
SKE	seek errors count
POH	power-on hours
RRT	calibration retries
PCC	power cycles count
RSE	read soft errors count
DMC	CRC errors count
OSS	offline surface scan

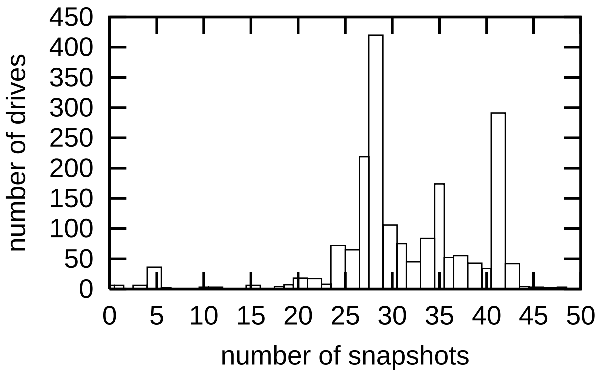

Figure 1. Number of snapshots per drive for drives with less than 50 snapshots. In addition, 21 drives have 50 or more snapshots, and 7 drives have more than 500 snapshots.

For the Quantum dataset, each data point is a time-stamped "snapshot" of SMART attribute values. A snapshot is simply a record of the SMART values for a drive at a particular time. Snapshots are taken at semi-regular intervals, ranging from a half-hour to several days. Figure 1 shows that most drives in the dataset have between 20 and 45 snapshots, but a few drives have several thousand snapshots, with a maximum of 6547 snapshots for one drive. Thousands of snapshots for a single drive may or may not indicate an anomaly in data collection. We view the wide range of snapshots to be an example of the challenges in real-world data collection, and all valid snapshots are used in the experiments described below.

5.1 Data preparation

As with most real-world datasets, the Quantum dataset has inconsistencies and invalid data which require preprocessing. We remove any snapshots that have clearly invalid values. We also remove two duplicate snapshots. There are 11 SMART attributes for the Quantum drives. These include power-on hours, seek errors, and spin-up time. Many attributes are positively correlated with failure, but none are perfectly correlated. Two attributes, read error rate and offline surface scan, are always zero and so are not used.

According to their specification, the SMART attributes are cumulative, meaning that their values are never reset by the drive firmware. Conceptually, it makes more sense to use incremental values than cumulative values, because a drive with a long life may have a large attribute value that is growing slowly, which is not likely to be indicative of a problem. So we transform the dataset by computing the difference between each snapshot and the previous one. Some of the computed differences are negative, contradicting the specification that attribute values are cumulative. These negative differences are set to zero.

There are many zero values in both the cumulative and differenced versions of the dataset. Because of this, zero is given its own bin. Conceptually, a value of one, e.g. one seek error, is very different from a value of zero.

5.2 Predicting drive failure

We predict that a disk drive will fail if *any* of its snapshots are identified as anomalous by the NBEM model or as failures by the naive Bayes classifier. Thus for our tests, only one alarm may sound for one drive. Additional information such as the number of snapshots predicted to fail is not used. We assume that it is impossible to predict disk drive failure more than 48 hours before it happens. Also, we want to maximize utility by not predicting that a drive will fail, and thus causing it to be replaced, until truly necessary. Thus data concerning each drive that fails is assumed to represent two drives: the snapshots within 48 hours of the end of its life represent the failed drive, while all prior snapshots represent a good drive. This approach adds to the data available for good drives.

The Quantum dataset has data concerning 1936 Eclipse drives. Of these, 16 drives were labeled as failures by users, i.e. returned to the factory due to an apparent fault. Six of the failures were not reproducible by drive engineers, so we consider these drives good and include them in the training and test sets as any other good drive. One failed drive has no snapshots in the dataset, and so is unusable. This leaves 9 drives labeled as failures by drive engineers, and 1927 labeled as good, or non-failures. Accounting for failed drives having both good and will-fail snapshots gives 7 additional good drives based on good snapshots of failed drives, because two failed drives only have snapshots for their last 48 hours. In total we have 94022 snapshots representing 1927 + 7 = 1934 "good" drives, and 34 snapshots representing 9 will-fail drives.

Table 2. Labeling of true/false positive/negative drives.

	true failure	true non-failure
predict failure	true positive *tp*	false positive *fp*
predict good	false negative *fn*	true negative *tn*

6. Experimental setup

The goal is to obtain a high true positive rate identifying will-fail drives correctly and a low false positive rate of predicting that a good drive will fail. Classification labels are given in Table 2. The true and false positive rates are defined as

$$\begin{aligned} tpr &= tp/(tp + fn) = tp/9 \\ fpr &= fp/(fp + tn) = fp/1934. \end{aligned}$$

The true positive rate is the number of true positives divided by the total number of drives identified by the model as anomalies, that is $tp/(tp + fn) = tp/9$, since there are 9 true failures. Here fn means false negative, or failed drives not labeled as anomalous. The false positive rate is the number of false positives divided by the total number of non-failed drives, or $fp/(fp + tn) = fp/1934$, since there are 1934 non-failures in this dataset. Here tn means true negative, or non-failed drives labeled as such.

Because the EM algorithm is based on random initialization, we perform 10-fold cross-validation to obtain average prediction performance. We always hold out the will-fail drives during training, thus making our approach a semi-supervised method. We hold out 10% of the good drives for testing for each cross-validation iteration. We always place all the will-fail drives in the test set, due to their small number. Note that because cross-validation is performed at the drive level, the number of snapshots used for training and testing varies. The typical fold uses 84620 ± 3698 snapshots for training, and 9436 ± 3698 snapshots for testing. For each of the 10 trained models, we perform a parameter sweep with the threshold t to obtain a receiver operating characteristic (ROC) curve (Swets et al., 2000), which plots the true positive rate against the false positive rate.

For the standard naive Bayes classifier, we plot a similar ROC curve. A naive Bayes classifier gives an estimate of the probability p that a snapshot belongs to the class of failed drives. By choosing a threshold t, a class label is predicted depending whether p is above or below t. Figure 3 shows the result of training and testing the standard

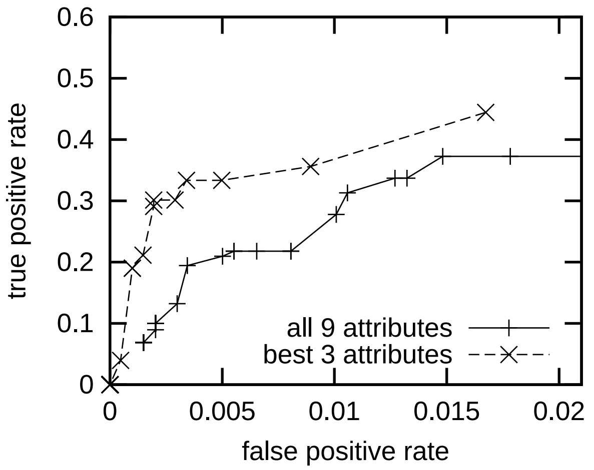

Figure 2. NBEM performance on disk drive failure prediction. Each point plotted is the true positive and false positive rate averaged over 10-fold cross-validation for a selected threshold value. Threshold values are not shown.

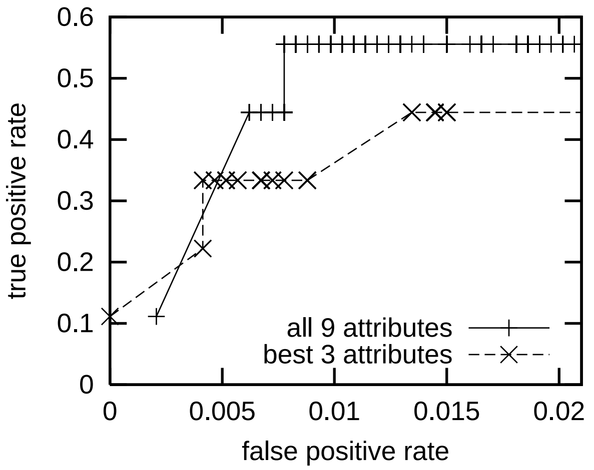

Figure 3. Standard naive Bayes performance on disk drive failure prediction. Each point plotted is the the true and false positive rate for a selected threshold value for the naive Bayes classifier. Threshold values are not shown.

naive Bayes classifier on the entire Quantum dataset, without cross-validation. The results may therefore be overoptimistic because of overfitting. However, naive Bayes classifiers typically do not overfit large datasets much, and informal experiments with data subsets show that overfitting is negligible here.

As mentioned above, two of the 11 SMART attributes for the Quantum dataset always have the value zero and are not used. Other research (Hughes et al., 2000) has shown that three attributes are most predictive: grown defect count (GDC), read soft errors (RSE), and seek errors (SKE). These were found by searching all combinations of attributes and ranking them by classification performance using other classification methods. We train and test NBEM models separately using all nine attributes and using only the three most predictive. For all NBEM experiments, we use two submodels and four equal-width continuous bins, so there are five bins for each attribute in all: one for zero, and four for the quartiles of the non-zero attribute values. We performed other tests (not shown here) that showed that these choices for the number of submodels and bins give the best performance.

7. Results

The results for naive Bayes EM and standard naive Bayes at predicting hard drive failures are presented in Figures 2 and 3 respectively. The receiver operating characteristic (ROC) curves show predictive accuracy for different threshold values. The threshold values are not shown in the figures. Optimal performance would be at the top left of each figure, i.e. with high true positive rate and low false positive rate. Note the very different scales on the two axes.

Because we use cross-validation for NBEM, each plotted point in Figure 2 corresponds to a single threshold value. The coordinates of each point plotted are averages taken over all folds of 10-fold cross-validation. For each point, the independent variable is the threshold t, and the dependent variables are the average false and true positive rates. Figure 3 shows the ROC curve for the single naive Bayes classifier.

Figures 2 and 3 show that we can adjust the threshold to trade-off true and false positive rates. This decision can be made based on the expected cost of each type of error. For example, at a threshold of 0.01, we can achieve a true positive rate of 0.33 (33% of will-fail drives) at a false positive rate of 0.0067 (0.67% of good drives are predicted likely to fail soon). With reference to this dataset, this means detecting 3/9 will-fail drives and giving 13/1934 incorrect warnings.

Both NBEM and standard naive Bayes are able to detect failures in the dataset while avoiding predicting too many good drives as failures. For example, using the three best attributes at a snapshot threshold of $t = 0.005$, NBEM can identify 0.33 = 3/9 failures while having a false positive rate of 0.0036 = 7/1934. Similarly, using all 9 predictive attributes at a class threshold of 0.001, standard naive Bayes can identify 0.56 = 5 / 9 of failures while having a false positive rate of 0.0082 = 16/1934. These threshold values were chosen by inspecting the ROC curves.

Error bars for NBEM are not shown in Figure 2, because they are too small to be clearly visible. They are given in Table 3. The errors are small enough to make statistically

Table 3. Error bars for NBEM true and false positive rates. Each number is the average of the relevant standard deviation for all thresholds.

	true positive rate	false positive rate
All attributes	0.0085	0.0012
Best 3 attributes	0.0083	0.0006

significant the difference between NBEM trained with all attributes versus with only the three best. Using the three best attributes gives a better true positive rate for every false positive rate over 0.001.

We considered measuring performance based on snapshot prediction accuracy, but we believe that measuring drive prediction performance is a more useful metric. In the same experiments, we found that the snapshot true positive rates were significantly higher, with lower false positive rates when compared to the drive rates. For example, NBEM achieves 0.52 ± 0.0029 true positive rate at 0.0009 ± 0.0001 false positive rate on snapshots.

The supervised naive Bayes classifier performs better using all SMART attributes than using only three. In general, naive Bayes classifiers are good at combining the contributions of multiple predictors that in isolation have low predictive power. Naive Bayes classifiers are also robust in the face of irrelevant attributes. These general characteristics are confirmed for this dataset. On the other hand, most clustering methods work much better in low dimensions than in high dimensions. The reason why the NBEM method performs worse with all predictors may be that its ability to find a good clustering is degraded.

Experiments for which results are not shown in this paper show that arithmetic smoothing via Lidstone's law of succession is important for reducing the variability of the NBEM model. Without smoothing for zero-counts, EM training converges to different models given different initializations. Adding zero-count smoothing gives more consistent results, and makes EM converge more quickly.

8. Conclusion

The failure prediction methods presented here perform better than the current industry standard methods, and they perform well enough to be useful in practice. The thresholding methods that are standard today in the disk drive industry have been estimated to yield a drive-level true positive rate of 0.04 to 0.11 at a false positive rate of 0.002 (Hughes et al., 2000). At the same false positive rate, NBEM achieves a true positive rate of 0.30, about three times better.

For another point of view on our results, suppose that the Quantum dataset is representative of a RAID (redundant array of inexpensive disks) containing 128 drives in a stressful environment, where a snapshot of each drive in the RAID is taken once per hour. Suppose that for anomaly detection we use NBEM on the three best attributes. If we sound an alarm for the most anomalous 0.05% of snapshots, then we will have an alarm about once every 16 hours on average. This percentage of snapshots corresponds at the drive level to a true positive rate of 0.30 and a false positive rate of 0.002, i.e. a ratio of $0.30 \times 9 = 2.7$ true failures predicted to $0.002 \times 1934 = 3.9$ false alarms. Therefore when an alarm is sounded about once every 16 hours, with probability $2.7/(2.7 + 3.9) = 0.41$ the alarm will predict a true failure, while with probability 0.59 the alarm will be false. This drive-level false alarm rate is tolerable because hot-swapping a drive in a well-designed RAID is an easy operation.

Acknowledgments

Thanks to Gordon Hughes and Joseph Murray, from the Center for Magnetic Recording Research SMART project, for providing the data for this project as well as their time and advice. Chris Reynolds from Quantum Corporation provided help and the original data.

References

Cheeseman, P., & Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. *Advances in Knowledge Discovery and Data Mining* (pp. 158–180). AAAI Press.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 185–197.

Elkan, C. (1997). *Boosting and naive Bayesian learning* (Technical Report CS97557). University of California, San Diego.

Eskin, E. (2000). Anomaly detection over noisy data using learned probability distributions. *Proceedings of the International Conference on Machine Learning* (pp. 255–262).

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation*, *121*, 256–285.

Hughes, G., Murray, J., Kreutz-Delgado, K., Elkan, C., & Tran, W. (2000). Improved disk-drive failure warnings. To appear in IEEE Transactions on Reliability, see `http://cmrr.ucsd.edu/smart/`.

Kohavi, R., Becker, B., & Sommerfield, D. (1997). Improving simple Bayes. *Proceedings of the European Conference on Machine Learning* (pp. 78–87).

Lane, T., & Brodley, C. E. (1997). An application of machine learning to anomaly detection. *National Information Systems Security Conference*, *1*, 366–380.

Lee, W., & Stolfo, S. (1998). Data mining approaches for intrusion detection. *Proceedings of the 7th USENIX Security Symposium* (pp. 79–83).

Lewis, D. (1998). Naive Bayes at forty: The independence assumption in information retrieval. *Conference proceedings of European Conference on Machine Learning* (pp. 4–15).

Lidstone, G. (1920). Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, *8*, 182–192.

Mitchell, T. M. (1996). *Machine learning*. New York: McGraw Hill.

Nass, R. (1995). SMART failure-prediction method now being endorsed for SCSI disk drives. *Electronic Design*, *43*, 40.

Ristad, E. S. (1995). *A natural law of succession* (Technical Report CS-TR-495-95). Princeton University.

Swets, J. A., Dawes, R. M., & Monahan, J. (2000). Better decisions through science. *Scientific American*, *283*, 82–87.

Towell, G. (2000). Local expert autoassociators for anomaly detection. *Proceedings of the International Conference on Machine Learning*.