Disk Failure Prediction in Data Centers via Online Learning

Jiang Xiao

SCTS/CGCL

Huazhong University of Science and Technology

Wuhan, China [jiangxiao@hust.edu.cn](mailto:jiangxiao@hust.edu.cn)

Yusheng Yi

SCTS/CGCL

Huazhong University of Science and Technology

Wuhan, China [hust@hust.edu.cn](mailto:hust@hust.edu.cn)

Zhuang Xiong

SCTS/CGCL

Huazhong University of Science and Technology

Wuhan, China [xzhuang@hust.edu.cn](mailto:xzhuang@hust.edu.cn)

Hai Jin

SCTS/CGCL

Huazhong University of Science and Technology

Wuhan, China [hjin@hust.edu.cn](mailto:hjin@hust.edu.cn)

Song Wu∗

SCTS/CGCL

Huazhong University of Science and Technology

Wuhan, China [wusong@hust.edu.cn](mailto:wusong@hust.edu.cn)

Kan Hu

SCTS/CGCL

Huazhong University of Science and Technology

Wuhan, China [hukan@hust.edu.cn](mailto:hukan@hust.edu.cn)

# ABSTRACT

磁盘故障已成为数据中心存储系统快速扩展的主要问题。基于SMART（自我监控，分析和报告技术）属性，许多研究人员使用机器学习技术来推导磁盘故障预测模型。尽管取得了重大进展，但大多数作品都依赖于脱机培训，因此受到“模型老化”问题的困扰，阻碍了它们适应即将来临的数据的不断更新。因此，我们有动力发现根本原因-“模型老化”的动态SMART分布，旨在解决性能下降的问题，以便在实践中进行全面研究。

在本文中，我们介绍了一种使用在线随机森林（ORF）的新型磁盘故障预测模型。我们基于ORF的模型可以随着数据的连续到达而自动演化，因此可以高度适应SMART随时间变化的变化。此外，就出色的预测性能而言，它与离线竞争对手相比具有有利的优势。在真实数据集上的实验表明，我们的ORF模型可以快速收敛到离线随机森林中，并实现93-99％的稳定故障检测率，且误报率较低。此外，我们证明了我们的方法能够为数据中心的长期使用保持稳定的预测性能

# KEYWORDS

Failure prediction, online learning, hard disk drive, SMART, storage system reliability

∗The corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [p](mailto:permissions@acm.org)[ermissions@acm.org.](mailto:ermissions@acm.org)

*ICPP 2018, August 13–16, 2018, Eugene, OR, USA*

© 2018 Association for Computing Machinery. ACM ISBN 978-1-4503-6510-9/18/08. . . $15.00

<https://doi.org/10.1145/3225058.3225106>

**ACM Reference Format:**

Jiang Xiao, Zhuang Xiong, Song Wu, Yusheng Yi, Hai Jin, and Kan Hu. 2018. Disk Failure Prediction in Data Centers via Online Learning. In *ICPP 2018: 47th International Conference on Parallel Processing, August 13–16, 2018, Eugene, OR, USA.* ACM, New York, NY, USA, [10](#_bookmark1) pages. [https://doi.org/10.](https://doi.org/10.1145/3225058.3225106) [1145/3225058.3225106](https://doi.org/10.1145/3225058.3225106)

# INTRODUCTION

数据中心存储系统的快速扩展使得最初偶然的组件故障成为常态[1]。 磁盘是最常出现故障的组件之一。 基于Microsoft数据的最新研究报告表明，磁盘故障占数据中心所有故障组件的76-95％[2]。 磁盘故障可能导致严重的数据丢失和随后的服务停机，从而给业务造成巨大损失[3]。 然而，磁盘容量的不断增长将增加扇区错误和数据损坏的可能性[4]。 Google的工程师[5]认为，数据中心的磁盘应容忍更高的错误率，应对这些错误的责任应重新分配给更高的存储层，以实现高性能和低成本。 因此，磁盘故障成为确保数据中心高度可靠服务的主要关注点。

关于提高存储系统可靠性的研究越来越多，主要有两个方面。 一种是无功容错技术，它通过设计擦除代码和数据冗余机制来重建磁盘故障后的数据[6]。 但是，它可能会影响磁盘的读/写操作并降低整体系统性能。 相反，主动的容错技术利用磁盘的过去行为，在实际发生故障之前执行预测。 自我监测，分析和报告技术（SMART）就是这样一种方法，它检测并报告各种驱动器可靠性指标，例如，重新分配的扇区数，负载周期，寻道错误率，读取错误[7]。 通过采用统计和机器学习技术，不断提出了各种基于SMART属性的主动磁盘故障预测模型，并在近年来变得越来越流行[8-17].

根据一定时期内的历史数据，可以通过上述主动方法进行离线培训和评估。请注意，这些模型在其应用的早期阶段可以表现出色。但是，随着时间的流逝，预测效果将大大降低。我们将此与当前离线模型相关的严重问题称为“模型老化”。为了更深入地了解模型老化的根本原因，我们进行了初步实验以模拟实际的长期使用。得出[14]的一致结果是，顺序收集的数据将逐渐改变累积SMART属性的基本分布，使现有模型随着时间的流逝而失去有效性，并且不足以预测未来的SMART数据。更具体地说，累积属性（例如“重新分配的扇区数”，“开机时间”）记录了磁盘整个生命周期中的累积发生次数，通常可以作为磁盘故障的有力指标。

现有的解决方案通过在离线模式下更新磁盘故障预测模型来解决模型老化问题。 Li等。 [14]提出了两种离线学习策略，即替换和累积。前者仅使用最后一个周期内收集的样本定期（即，每周一次）更新模型，而后者则使用从开始到最后一个周期收集的所有样本，然后应用更新的模型预测当前周期内的故障。尽管两种策略都能很好地保持预测准确性，但是它们都不能绕过定期重新训练模型的过程。此外，离线学习缺乏适应到达数据的动态模式的能力。这激励我们找出一种替代机制，该机制支持在SMART属性的变化分布下自动进行模型更新。在线学习[18]表现出巨大的性能提升潜力，数据的顺序到达和优于离线学习的优势，包括实时预测和较低的内存需求

不幸的是，将在线学习方法用于磁盘故障预测并不是一件容易的事，但具有挑战性。第一个挑战是如何在在线操作中标记SMART样品。与离线学习不同，SMART样本可以很容易地用磁盘状态的全部知识标记为阳性（即有缺陷）或阴性（即健康），而在线学习则很难标记出传入的样本。这是因为离线模式提供了静态训练数据，可用于提前通知数据集中的磁盘是否有故障。相比之下，在线模式下的预测模型将随着串行收集的训练数据的进行而不断发展，而磁盘的实际状态仍然不确定。特别是某些操作磁盘可能在极短的时间内出现故障，而相应的样本只有在发生磁盘故障之前才能被标记为正。为此，我们引入了一种自动在线标签方法，该方法可以为系统中的每个磁盘存储在最近一段时间（例如一周）内收集的样本。除非确实发生磁盘故障或新的样本到来，否则这些存储的样本将保持未标记状态。当磁盘发生故障时，属于故障磁盘的未标记样本将被标记为正并被删除。所有过时的样本将被标记为阴性，并由新收集的样本替换。标记样品后，将使用它们来更新模型。

另一个关键挑战在于，故障磁盘与普通磁盘之间的比率非常不平衡，因为磁盘故障相对少见，并且故障磁盘仅占所有磁盘的一小部分。根据现实世界的数据集[19]，阴性样本的数量可能是阳性样本的数百至数千倍。在离线模式下，不平衡的情况可以通过将整个训练集的负类下采样到接近正类的大小来解决[20]。但是，在在线模式下，随着时间的推移会收集训练样本，因此无法进行采样。我们从在线装袋法[21]中得出想法，并提出了一个可行的解决方案，其中正样本和负样本的顺序到达分别由两个不同速率参数的泊松分布建模。结果，相对很少选择顺序到达的负样本来更新模型。我们的实验证明了该方法在解决样品不平衡问题方面的有效性。

在本文中，我们设计了一种基于在线随机森林（ORF）[22]的新颖的在线学习模型，用于磁盘故障预测。 ORF算法固有地比以前的工作具有多个优势，例如，ORF的整体训练和测试过程可以很容易地并行化，因为森林中的每棵树都是独立于其他树构建和测试的[22]，并且模型具有很高的可解释性，因此它们可以用于揭示磁盘故障的真正原因并帮助提高存储系统的可靠性[14，23]。更重要的是，ORF使用逐渐收集的SMART样本即时生成随机树。为了对知识进行时间加权，它还利用“包外错误”（OOBE）丢弃了过时的树。

[26]并提出了新的树，以适应当前的培训数据分布作为替代。因此，我们基于ORF的预测模型具有对SMART属性分布变化的自动适应性，以及对标签噪声的良好鲁棒性，使其适合于实际的长期使用。

此外，我们的在线学习方法比以前的研究具有更高的竞争性故障预测性能，并且可以与普通离线随机森林（RF）相媲美（甚至更好）。对从34,535个磁盘收集的数据集进行的实验（在三年内进行监控）[19]表明，我们基于ORF的模型在六个月内迅速收敛到离线RF的性能，并实现了93-99的故障检测率（FDR）前六个月后具有合理的低误报警率（FAR）的％。与使用更新策略的离线RF模型相比[14]，我们证明了基于ORF的算法可以维持相当低的FAR，同时可以与这些定期更新的RF模型实现可比的FDR，但是，在初始部署后无需模型重新训练。

总之，我们在本文中提出的主要贡献如下：

据我们所知，我们率先在磁盘故障预测中使用在线学习方法来绕过离线副本的模型老化问题。

我们指出了将在线学习应用于磁盘故障预测的挑战，并提出了解决方案。

我们还提出了一种基于ORF的磁盘故障预测的特定在线学习方法。

我们模拟了基于ORF的预测模型的长期使用，并证明了我们的方法在实际数据中心中的有效性和适应性。

在下面的第2节中，我们调查了基于SMART的磁盘故障预测的相关工作。 第三部分介绍了用于磁盘故障预测的在线学习方法。 我们评估模型并在第4节中介绍实验结果，然后在第5节中给出结论。

# RELATED WORK

Different from the traditional reactive fault tolerance technique which troubleshoots a system upon occurrence of disk failures, proactive fault tolerance technique anticipates potential disk fail- ures and provides preventive measurement to prevent failure from occurring. It can also significantly improve the reliability and avail- ability of storage system, however, without affecting the read/write performance of drives. Therefore, many researchers have focused especially on SMART-based proactive disk failure prediction.

SMART is a self-monitoring system supported by most disk man- ufacturers in their products [[7](#_bookmark8)]. The system detects and reports vari- ous indicators correlated with impending disk failure. The anomaly detection method used by SMART is simple threshold-based algo- rithm, which triggers a system warning when any SMART attribute exceeds its predefined threshold. These thresholds are set conser- vatively by manufacturers to avoid false alarms at the expense of prediction accuracy. Due to the simpleness of threshold algo- rithm and the conservative settings of thresholds, this technology achieves poor FDRs of 3-10% [[10].](#_bookmark11)

To enhance the SMART-based failure prediction performance, machine learning and statistical techniques are proposed to build prediction models. Hamerly and Elkan [[8](#_bookmark9)] employed a supervised naive Bayes classifier to predict disk failures and achieved a pre- diction accuracy of 33% with 0.67% FAR on a small dataset of 1,936 drives (i.e., only 9 of them do fail). A mixture model of naive Bayes submodels trained using *expectation-maximization* was also inves- tigated and achieved a similar performance.

Hughes *et al.* [[9](#_bookmark10)] applied a non-parametric statistical method, the *Wilcoxon rank-sum test*, to train prediction models, as they found that most of the critical SMART attributes are non-parametrically distributed. Based on a dataset containing 2-3 months of reliability design test data from 3,744 drives, the highest FDR they achieved was 60% at 0.5% FAR. In their later work [[10](#_bookmark11)], several methods including the rank-sum test, *support vector machines* (SVM), *unsu- pervised clustering*, and *reverse arrangements test* were compared on a dataset containing only 369 drives. When using a certain set of four SMART attributes, the results showed that rank-sum test could outperform SVM with 28.1 percent detection rate. However, SVM provided the best FDR of 50.6% with no measured false alarms when all of 25 attributes were used. Wang at al. [[11](#_bookmark12)] then improved the SVM-based prediction model by attaching the change rates of SMART attributes as explanatory variables and reached a FDR of 80% at 0.3% FAR.

Despite the good prediction performance of SVM, its compu- tational efficiency and memory use are too expensive for online monitoring [[12](#_bookmark13)]. Wang *et al.* [[12](#_bookmark13)] used *Mahalanobis distance* (MD) to aggregate the input variables into one index and detect disk anom- aly by setting an appropriate threshold. They found that using only the critical SMART attributes selected by FMMEA (*Failure Modes, Mechanisms and Effects Analysis*) could lead to better performance

compared with using all attributes. In their later work [[13](#_bookmark14)], they ex- panded the work and proposed a sliding-window-based generalized likelihood radio test to track the anomaly progression in disks. The method delivered a 68% FDR with zero FAR on the same dataset used by [[10].](#_bookmark11)

*Back Propagation Artificial Neural Networks* (BP ANN) [[11](#_bookmark12)] and *Classification and Regression Trees* (CART) [[14](#_bookmark15)] were explored by Zhu *et al.* to further improve the FDR. The results on a real-world dataset containing 8 weeks of SMART data from 23,395 drives showed that both models could achieve great FDRs which were about 95% with a reasonable low FAR. Recently, this research group focused on gauging the different health statuses of disk drives. They formulated the disk failure prediction as a multi-level classification problem and predicted the health degree rather than binary status of disks. *Recurrent Neural Networks* (RNN) [[15](#_bookmark16)] and *Gradient Boosted Regression Trees* (GBRTs) [[16](#_bookmark17)] were used to build the residual life prediction models and both models achieved reasonable accurate health status assessment. On a large real-world dataset, the RNN- based method delivered about 40-60% ACC (*accuracy of residual life level assessment*) on failed samples [[17].](#_bookmark18)

Recently, Mahdisoltani *et al.* [[25](#_bookmark26)] took a different approach, pre- dicting sector errors rather than disk failures, to improve the relia- bility of storage system. They explored several techniques including CART, SVM, NN, *Logistic Regression* (LR), and RF, and found that the simplest and easiest to train machine learning models, RF, achieved the highest prediction accuracy. The training of sector error predic- tors was found to be robust for small training sets or training data coming from a different drive model. In addition, they proposed a number of possible use cases for the predictors and shown that the mean time to detecting errors (and hence the window of vulnera- bility to data loss) can be greatly reduced by adjusting scrub rates based on error predictions.

The aforementioned methods are all designed to train prediction models in batch or offline mode. Though possess great prediction performance and have demonstrated their superiority in a number of circumstances, all these models cannot get rid of the model aging problem, since the underlying distribution of SMART attributes changes over time. Besides that, they all require the access to full training data and wish the dataset sufficient enough to contain ade- quate information of building high-performance models. However in practice, the full training data may not be given initially but gathered gradually. The process of collecting informative dataset can take a very long time (maybe years), especially for the small- scale data centers, as disk failure is an accidental event while the SMART information of failed disk is what matters. This implies the deployment of models that support online learning is more enlightened, as the online models are capable of integrating future data automatically.

In this paper, we attempt to adopt online learning method for disk failure prediction and present an ORF-based method for practical usage. ORF can evolve with sequential arrival of data on-the-fly and forget old information by controlled discarding outdated trees. As a result, our online learning model can dynamically adapt to new patterns of data and operate without further concern of model aging.

# THE PROPOSED METHOD

Our goal is to predict whether a disk will fail within a given time interval, based on the SMART data that the disk reported. This problem of predicting future errors can be formulated as a binary classification problem. In the following discussion, we constrain such period into seven days before a faulty event, for the sake of simplicity.

In this section, we start with a brief introduction to ORFs al- gorithm and then present our ORF-based online learning method in detail. The difficulties of adopting ORFs to build disk failure prediction models are discussed concomitantly.

# Online Random Forests

*Online Random Forests* (ORFs) [[22](#_bookmark23)] are commonly used for classi- fication and regression, as a combination of strengths of *Random Forests* (RFs) [[24](#_bookmark25)] and *online machine learning* [[18](#_bookmark19)]. Given a train- ing set consisted of two classes, the ensemble method operates by constructing numerous decision trees in the training phase and outputting the class that is the mode of two classes of the individual trees.

ORFs operate in online mode benefiting from online bagging

[[21](#_bookmark22)] and online tree growth techniques. To achieve the same effect as offline bagging (that is, each training sample is randomly selected by each tree in the forest), ORFs use the online bagging method proposed by Oza *et al.* [[21](#_bookmark22)], which applies a *Poisson distribution* function to model the sequential arrival of data. Oza *et al.* proved convergence of this method to offline bagging.

During the growth of a randomized tree in ORF, each decision node creates a set of random tests and chooses the best according to some impurity measurement. For hard drive failure prediction, each random test is in the form of *SMART\_i* > *θ* , where *SMART\_i* represents the SMART attribute of ID number *i*, and *θ* is a threshold which decides the left and right partitions of SMART samples in that node. To assess the quality of each test *s* in the set of *N* random tests S = {*s*1, . . . , *sN* }, we use the *Gini Impurity* as the impurity measurement. The Gini Impurity of node D is calculated as

*G*(D) = *p*0(1 − *p*0) + *p*1(1 − *p*1) (1)

where *p*0, *p*1 are restricted by *p*0 +*p*1 = 1, denoting the label density of the two classes in node , respectively. Note that *G* 0, 0.5 and a larger value implies that the dataset is more impure. For a random test *s* , we assume that node is split into left child node *ls* and right child node *rs* according to the test, and the information gain of test *s* is measured as

D D

∈ S D

D (D) ∈ [ ]

∆*G*(D, *s*) = *G*(D) − |D*ls* | *G*(D*ls* ) − |D*rs* | *G*(D*rs* ) (2)

samples a node should see before it splits, ensuring the node has rubust statistics. *MinGain* (symbolized as *β*) limits the minimum gain a split should achieve, ensuring the split is worthwhile for the predition purpose. Therefore, the essential condition for the node D to split is |D| ≥ *α* and ∃*s* ∈ S : ∆*G*(D, *s*) ≥ *β*.

# Online Learning for Disk Failure Prediction

Previous studies leverage offline machine learning techniques to design disk failure prediction model, however, their proper func- tioning relies on the assumption that the entire training data is accessible and sufficient enough to contain the information of build- ing high-performance models. The practical case is that the training data is gradually gathered instead of being given in advance. This implies that the training data collected within the initial period may be insufficient and become the bottleneck of performance. There- fore, the deployment of models that support online learning is more enlightened. More importantly, online learning supports automatic evolution with the sequential arrival of data and is highly adaptive to the dynamic distribution of SMART data, and thus get rid of the model aging problem.

Our online learning technique innovates the prediction model with ORF algorithm, which inherits all the merits of RF and pos- sesses beneficial properties for predicting disk failure. First, the overall training and testing procedures of ORF can be easy paral- lelized, as each tree in a forest is built and tested independently from others, which makes the time efficiency of ORF much higher than that of gradient boosting methods, such as GBDT (*Gradient Boosted Decision Trees*). Second, ORF models are highly interpretable so they can be used to reveal the real cause of disk failures and help improving the reliability of storage systems [[23](#_bookmark24)]. Additionally, ORFs are also more robust against label noise compared to boosting and other ensemble methods [[22].](#_bookmark23)

We build prediction models using SMART attributes as explana- tory variables together with the response variable representing whether the disk will fail within the next seven days. A major challenge of fitting high quality models is the highly unbalanced distribution of SMART training data, since healthy drives account for the vast majority. Note that in the original ORF algorithm, all the training samples are modeled by a same Poisson distribution function, that is to say, both positive and negative classes are treated equally. As a result, the prediction models may exhibit poor per- formance, as the classification algorithms are typically optimized to maximize the overall accuracy [[20](#_bookmark21)] and thus seriously biased towards the negative class. To address this issue, we put forward a strategy developed from online bagging [[21](#_bookmark22)] and introduce two

( ) F { }

|D|

|D|

hyper-parameters: *λp* and *λn* . We denote the *tth* tree in the forest

where represents the number of samples contained in node . Aiming at reducing the impurity of , we prefer the Gini Impurity of child nodes to be as small as possible. Hence, when a node meets split conditions, the algorithm firstly calculates the information gain of each test created for the node and then chooses the one with highest gain as the splitting function of the node.

D

|D| D

Since each decision node gathers statistics of the sequentially col- lected training samples on-the-fly, two hyper-parameters: *MinGain*

as *ft x* and the entire forest is denoted as = *f*1, ..., *fT* , where *T* is the number of trees in the forest. When a training sample *x*, *y* arrives, each tree *ft* will be updated on the sample for *k* times. The *k* for positive and negative sample is generated from *Poisson λp* and *Poisson λn* , respectively. This strategy can be formulated as

.*Poisson*(*λp* ) y=1

( ) ( )

⟨˙ ⟩ ∈ F

and *MinParentSize* [[22](#_bookmark23)], are used to control the node splitting.

*MinParentSize* (symbolized as *α* ) limits the minimum number of

*k*(⟨*x*˙, *y*⟩) =

*Poisson*(*λn*

, (3)

) y=0

**Algorithm 1** ORF for disk failure prediction

**Input:** Sequential training sample: *x*, *y* **Input:** Number of trees in forest: *T* **Input:** MinParentSize: *α* ; MinGain: *β*

⟨ ˙ ⟩

**Input:** Sample balance parameters: *λp* , *λn*

**Output:** ORF model for disk failure prediction

1: // For each tree *ft*

∈ F

2: **for** *t* = 1 *T* **do**

→

3: **if** *y* == 1 **then**

4: *k Poisson λp*

← ( )

5: **else**

6: *k Poisson λn*

← ( )

7: **end if**

8: **if** *k* > 0 **then**

9: // Update the tree *ft* for *k* times

10: **for** *i* = 1 *k* **do**

→

11: *j* = FindLeaf(*x*)

D ˙

12: UpdateNode( *j* , *x*, *y* )

D ⟨˙ ⟩

13: **if** *j α* and *s* : ∆*G j* , *s β* **then**

|D | ≥ ∃ ∈ S (D ) ≥

14: // Find the best split from

S

15: *sj* = argmax*s* ∆*G j* , *s*

∈S (D )

16: CreateLeftChild( *jls* )

D

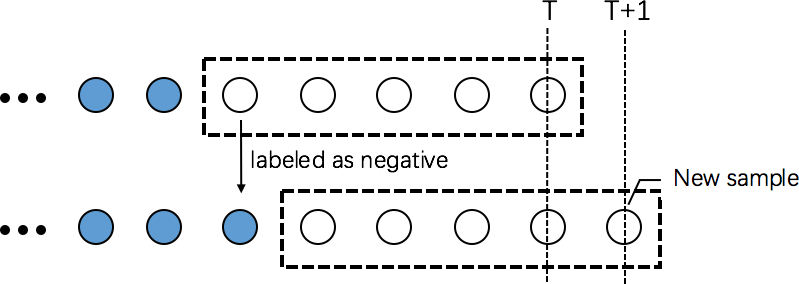
17: CreateRightChild( *jr s* )

D

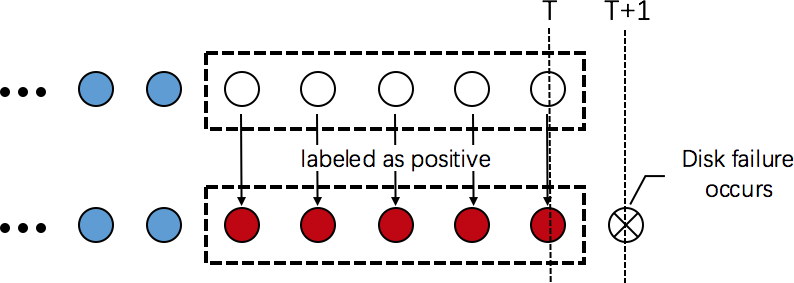
18: **end if**

19: **end for**

20: **else**



1. If new sample arrives at T+1, the last sample in T’s queue will be labeled as negative.



1. If disk fails at T+1, all samples in T’s queue will be labeled as positive.

## Figure 1: Automatic online label method. White dots stand for unlabeled samples, blue and red dots stand for negative and positive samples respectively.

and is usually set to a constant number greater than 1. The impact

21: // Update the OOBE of tree *ft*

22: *OOBEt* ← UpdateOOBE(⟨*x*˙, *y*⟩)

of *λ*, *λp* ,and *λn* on the prediction performance of offline RF and

23: // Estimate whether *ft* is decayed

24: **if** *OOBEt* > *θOOBE* and *AGEt* > *θAGE* **then**

25: // Replace the outdated tree with new tree

26: *ft* = NewTree()

27: **end if**

28: **end if**

29: **end for**

where *k x*, *y* denotes the update frequency *k* for sample *x*, *y* . The hyper-parameters *λp* and *λn* are usually set to constant num- bers. In our case, we set *λp* equal to 1 and *λn* to be a decimal much less than 1 (such as 0.01), thus the *k* for negative sample has a much smaller chance of taking non-zero positive integer value than that for positive sample. As a result, sequentially arrival negative samples are relatively rarely selected by a tree for update.

(⟨ ˙ ⟩) ⟨˙ ⟩

For compared offline models, we introduce a hyper-parameter: *NeдSampleRadio* (symbolized as *λ*), to balance the training data. Given a training set *D*, only its subset *Dp* + *Dnc* is specified as the actual input of the RF model, where *Dp* represents the set of positive samples and *Dnc* represents a subset of negative samples, which is randomly selected from *Dn* . The hyper-parameter *λ* is used to modify the probability distributions of the positive and negative samples, formulized as

ORF models is discussed in detail later in section 4.4.

Additionally, our ORF-based prediction model provides a mecha- nism of unlearning old information, which can be very useful since the underlying distribution of SMART data changes over time. For each tree *ft* , the sample *x*, *y* can be used to update the tree only if this *k* takes a non-zero positive integer. When the *k* takes a value of zero from *Poisson λ* , the sample *x*, *y* then will be used to update the *OOBE* of tree *ft* (denoted as *OOBEt* ). Based on the estimate of *OOBEt* and the age of tree *ft* (denoted as *AGEt* ), ORF algorithm discards the outdated trees with significant out-of-bag- error and generates new trees as replacements. As the effect of a single tree in the forest is relatively small, it should be unharmful for the ensemble to discard one tree. However, continuously re- placing decayed trees with new fitting trees ensures our prediction models great adaptivity to distribution changes throughout time. The improved ORF algorithm for disk failure prediction is shown in Algorithm 1.

∈ F ⟨˙ ⟩

( ) ⟨ ˙ ⟩

As described in Section 1, sample labeling can be another chal- lenge when prediction models operate in online mode. Unlike offline models learning from static training data, online ones evolve with the sequentially gathered training samples on-the-fly. However, currently collected samples can not be easily labeled because the actual status of the disks are still uncertain. That is to say, some of the disks that are still in operation may be very risky and will fail soon, but their samples can only be labeled as positive after the occurrence of actual failures. We temporarily store the samples collected within the most recent period for each disk and keep

*λ* = |*Dnc* |

|*Dp* |

(4)

them unlabeled unless there is enough confidence to label them. The proposed automatic *online label* method is shown in Figure 1.

**Algorithm 2** ORF-based online learning

**Input:** Gathered SMART sample: *x* (can be NULL if *y* = 1)

˙

**Input:** Disk identifier: *i*; Current disk status: *y*

**Output:** Prediction for disk: *y*′

1: // Model update phase

2: **if** *y* == 1 **then**

3: // Disk *Di* failed

4: **while** *Qi* is not empty **do**

attributes are manufacturer-specific as their encoding and meaning are distinct across disk models. Prior works [[14](#_bookmark15), [20](#_bookmark21)] pointed out that separate training is in demand for different disk models, by which the entire dataset is divided. Since some disk models have very small population and few available SMART attributes, we select two models with the highest data volume to conduct our experiments. Table I presents the statistics of chosen disk models. Compared with the dataset used by Wang *et al.* [[14](#_bookmark15)], our datasets cover a much

5: *x*˙′ ← dequeue(*Qi* )

longer duration of sample acquisition and contain more failed drives,

6: updateORF( *x* ′, 1 ) // call Algorithm 1

⟨ ˙ ⟩

7: **end while**

8: deleteDisk(*Di* )

9: **else**

10: // Disk *Di* is operating

11: **if** isFull(*Qi* ) **then**

|  |  |  |
| --- | --- | --- |
|  | ***STA*** | ***STB*** |
| **DiskModel** | ST4000DM000 | ST3000DM001 |
| **Capacity(TB)** | 4 | 3 |
| **#GoodDisks** | 34,535 | 2,898 |
| **#FailedDisks** | 1,996 | 1,357 |
| **Duration** | 39 months | 20 months |

which allow us for better simulation and assessment.

## Table 1: Overview of dataset

12:

*x*˙′ ← dequeue(*Qi* )

13: updateORF( *x* ′, 0 ) // call Algorithm 1

⟨ ˙ ⟩

14: **end if**

15: enqueue(*Qi* , *x*)

˙

16: // Prediction phase

˙

17: *y*′

←′

18: **if** *y*

predictORF(*x*)

== 1 **then**

# 4.2 Feature Selection

19: // *Di* is risky

20: // Immediate data migration is recommended

21: Trigger an alarm

22: **end if**

23: **end if**

We denote disk drive with identifier *i* in the system as *Di* . A fixed- length queue, denoted as *Qi* , is initialized for *Di* to store its last reported SMART samples. These samples stored in *Qi* will remain unlabeled until *Di* actually fails or new sample arrives. After *Di* has failed, all the samples in the queue *Qi* will be labeled as positive and then used to update ORF model. While *Di* is still in operation, the oldest samples in *Qi* will be labeled as negative and replaced continuously with new ones. Meanwhile, ORF model is applied to forecast the health status of *Di* with currently gathered SMART sample, and a positive prediction indicates a high risk of *Di* . Our detailed ORF-based online learning method is shown in Algorithm 2.

# EXPERIMENTAL RESULTS

This section aims to evaluate the prediction performance of the ORF model in practice. It is necessary to simulate the long-term use of such online learning mechanism and compare it with the manual update strategies for offline models.

# A Look at the Field Data

To validate our online learning method, we use the public dataset provided by Backblaze [[19](#_bookmark20)] which took a daily snapshot of each operational disk drive in their data centers since 2013. The snapshot includes all SMART values reported by a drive along with some basic information, *e.g.,* time stamp, the drive’s serial number, and model number. There are over 100, 000 disks recorded in the dataset, covering more than 30 different disk models. Note that the SMART

Before in-depth analysis, a prerequisite called ‘feature selection’ is executed to remove redundant and irrelevant features. This pre- processing can not only reduce the time of model training and prediction, but enhance the prediction performance [[10].](#_bookmark11)

For both datasets *STA* and *STB*, each disk drive reports 24 SMART attributes. Note that each of these SMART attributes contains two values: a 6-byte raw value (denoted as *Raw*) and a 1-byte normalized value (denoted as *Norm*) [[7](#_bookmark8)]. The *Norm* of a SMART attribute is usually calculated from the *Raw* by a vendor-specific formula. Since some *Norm*s may lose accuracy while their corresponding *Raw*s are more sensitive to health status of disks [[14](#_bookmark15)]. We treat both *Norm* and *Raw* of each SMART attribute as candidates, such that there are 48 features in total to explore.

We first operate Wilcoxon rank sum tests on each feature to indicate if it can differentiate between the positive and negative samples, such that 20 features are filtered out by this test since they fail to make a distinction. We then continue to study how the remaining 28 features can contribute disk failure detection separately, by comparing the FDRs of RF models built on different combinations of these features. As a result, nine of them are found to be redundant and thus abandoned. Table II lists the selected 19 features, including 9 *Norm*s and 10 *Raw*s.

The range of values spanned by different features varies widely. To avoid bias towards features with larger values, we apply feature scaling for data normalization, referring to the following formula:

*x* ′ = *x* − *xmin* , (5)

*xmax* − *xmin*

where *x* is the original value of a feature, *xmax* and *xmin* are the maximum value and the minimum value of this feature for the set of data with the same disk model, respectively.

# Metrics

We use *failure detection rate* (FDR) and *false alarm rate* (FAR) for evaluation. A failed disk is correctly detected only when at lease

## Table 2: Selected SMART Features

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID#** | **Attribute Name** | **Norm**  √ | **Raw** | **Rank**∗ |
| 1 | Read Error Rate | √ | √ | 13 |
| 5 | Reallocated Sectors Count | √ |  | 3 |
| 7 | Seek Error Rate |  | √ | 7 |
| 9 | Power-On Hours |  | √ | 5 |
| 12 | Power Cycle Count |  | √ | 11 |
| 183 | Runtime Bad Block | √ | √ | 8 |
| 184 | End-to-End Error | √ | √ | 4 |
| 187 | Reported Uncorrectable Errors | √ |  | 1 |
| 189 | High Fly Writes | √ | √ | 10 |
| 193 | Load Cycle Count | √ | √ | 6 |
| 197 | Current Pending Sector Count | √ | √ | 2 |
| 198 | Uncorrectable Sector Count |  | √ | 9 |
| 199 | UltraDMA CRC Error Count |  |  | 12 |

∗The Rank represents the sort of SMART attribute’s contribution.

one of the samples collected within the last week before failure is predicted positive. FDR is defined as the fraction of failed disks that are correctly predicted to be failed:

#true positives #true positives + #false negatives

FDR =

A good disk is mis-classified if any of the samples collected outside the latest week is predicted to be positive. FAR means the fraction of good disks that are mis-classified as failed:

## Table 3: Impact of *λ* on Offline RF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *λ* | ***STA*** | | ***STB*** | |
| **FDR(%)** | **FAR(%)** | **FDR(%)** | **FAR(%)** |
| 1 | 98.22 ± 0.25 | 11.88 ± 2.62 | 92.26 ± 0.27 | 6.46 ± 0.54 |
| 2 | 99.02 ± 0.31 | 2.33 ± 0.95 | 88.77 ± 0.38 | 1.35 ± 0.18 |
| **3** | **98.16**±**0.74** | **0.76**±**0.17** | **85.45**±**0.54** | **0.73**±**0.06** |
| 4 | 94.58 ± 0.64 | 0.05 ± 0.04 | 82.61 ± 0.65 | 0.67 ± 0.04 |
| 5 | 92.00 ± 0.14 | 0.00 | 80.00 ± 0.55 | 0.66 ± 0.07 |
| *Max* | 35.14 ± 0.18 | 0.00 | 29.45 ± 0.76 | 0.00 |

*Hyper-parameter Impact.* A major impedance for prediction model is the highly unbalanced distribution of training data, since good disks account for the vast majority and only a small fraction of samples reported by failed disks are labeled positive. When trained on intrinsic imbalanced datasets, the outcome of prediction models may be no longer adequate since typical classifiers are optimized to maximize the overall accuracy [[20](#_bookmark21)]. To overcome this limitation, we introduce hyper-parameters *λp* , *λn* for the ORF model and *λ* for the offline RF model to strike a balance of the training data. These hyper-parameters have great impact on the prediction performance. To identify the optimal parameters, we train the models by varying the settings and then apply the final models on the test set.

To be concrete, the number of tests (*N* ) is set to 5,000 and the number of trees (*T* ) in the forest is set to 30. We run experiments with more trees, but no significant improvement is observed. Other

FAR

#false positives

= #true negatives + #false positives

important parameters for ORF are set as: *MinParentSize α* = 200,

*MinGain β* = 0.1. These parameter settings remain unchanged

Note that there is usually a trade-off between FDR and FAR, and the FARs should be kept the same when we make comparison with other prediction models.

# Evaluation Results

*Experimental Setup.* To evaluate our ORF model, disks in each dataset are randomly divided into training set and test set, where the training set contains 70% of all the good and failed disks and the remaining 30% are in the test set.

As mentioned before, our goal is to predict whether a disk will fail within the next seven days. To better simulate the real-world scenario, for each failed disk in the training set, only samples col- lected within the last week before failure are labeled as positive (*y* = 1). The rest samples of this disk are labeled as negative (*y* = 0), as the disk did not break down within a week after these samples were collected. For each good disk in the training set, however, the samples collected in the latest week can not be labeled, as there is no guarantee that the disk will not fail in the next few days, and the samples collected outside the latest week can be labeled as negative samples. It should be noted that there may be some samples of the failed disks showing obvious fault characteristics but labeled as negative. However, these samples only account for a very small part of all negative samples and have a rare chance of being selected to train or update prediction models. Moreover, both ORF model is highly robust against label noise, thus the impact of these samples on prediction performance is negligible.

in the following discussion unless otherwise specified. We repeat each experiment five times and calculate the average and standard deviation of the FDR and FAR.

As described in section 2, the *λ* is used to downsample the neg- ative class to an amount that is close to the size of the positive class. The impact of *λ* on the prediction performance of offline RF model is presented in Table III. We can observe that, if no action is taken to balance the training data (*λ* = *Max*), the RF models will be seriously biased towards the good disks and lead to poor FDRs. With the decrease of *λ*, the FDR of the RF model gradually increases along with the FAR. To ensure an acceptable FAR, we set the *λ* = 3 for the RF model as well for other offline models.

*λp* and *λn* are used to adjust the ratio of sequentially arrival positive and negative samples being selected to update the ORF model. A larger value of *λn* means that the negative samples have a greater chance of being selected. We set *λp* = 1 and the impact of *λn* on the ORF model is presented in Table IV. As expected, adjusting the *λn* provides a trade-off between FDR and FAR. When the *λn* is set to 0.02, our ORF models achieve competitive prediction performance with a 98% FDR for *STA* and a 85% FDR for *STB* at acceptable FARs. In the following experiments, we set *λp* = 1, *λn* =

0.02 for ORF models.

*Algorithms Compared.* In addition to RF algorithm, we also com- pare our ORF algorithm with two common classification algorithms: SVM and *decision trees* (DT). For SVM, we use the LIBSVM [[28](#_bookmark28)] library. Parameters for SVM are set as follows: *svm*\_*type* = C-SVC,

## Table 4: Impact of *λn* on ORF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *λn* | ***STA*** | | ***STB*** | |
| **FDR(%)** | **FAR(%)** | **FDR(%)** | **FAR(%)** |
| 0.01 | 98.50 ± 0.19 | 24.88 ± 3.33 | 90.91 ± 0.23 | 5.25 ± 0.58 |
| **0.02** | **98.08** ± **0.37** | **0.66** ± **0.35** | **85.64** ± **0.37** | **0.85** ± **0.14** |
| 0.03 | 95.86 ± 0.75 | 0.10 ± 0.11 | 74.37 ± 1.75 | 0.58 ± 0.04 |
| 0.05 | 84.44 ± 0.65 | 0.01 ± 0.01 | 59.58 ± 0.50 | 0.30 ± 0.05 |
| 0.10 | 65.67 ± 3.11 | 0.00 | 47.81 ± 0.93 | 0.12 ± 0.03 |
| 1.00 | 23.58 ± 0.00 | 0.00 | 28.23 ± 1.55 | 0.09 ± 0.03 |

100

90

80

70

FDR(%)

60

50

40

ORF vs Offline models

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | | |
|  |  |  |  | | |
|  |  |  |  | | |
|  |  |  |  | | |
|  |  |  |  | | |
|  | ORF | |
|  |  |  |  | Offline RF DT | |
|  |  |  |  |  | SVM |

*kernel* \_*type* = *radial basis function* (RBF). The *gamma* in RBF ker- nel and *cost* are adjusted to trade off between FDR and FAR. We perform a grid search to find the parameter combination that pro- duces the highest FDR with a FAR less than 1%. For DT, we use

5 10 15 20

Number Of Months

## Figure 2: Failure detection rates of ORF and offline models on dataset *STA*. All points on each curve ensure FARs around 1.0%.

ORF vs Offline models

the *f itctree* [[29](#_bookmark29)] function provided by Matlab to build the classi- fication tree model. Some critical parameters are set as follows: *SplitCriterion* = gdi (Gini’s diversity index), *Max NumSplits* = 100. Different *Weiдhts* for positive and negative classes can be used to adjust prediction performance. Other parameters are set to default values.

*Evaluation of ORF Model.* We simulate the sequential arrival of training data according to the timestamp of labeled samples, and our ORF model evolves with the data over time. During the evolution procedure, we evaluate the prediction performance of the ORF model on the test set monthly. To ensure a fair comparison, each month we build offline models with all the training data collected

90

85

ORF

Offline RF DT

SVM

80

75

FDR(%)

70

65

60

55

5 10 15 20

Number Of Months

so far and investigate their performance on the same test set.

Figure 2 and Figure 3 depict the FDRs of ORF and the three offline models on dataset *STA* and *STB*, respectively. All the points on both figures are measured under the constraint that the FAR is around 1.0%. At the beginning, all models exhibit poor prediction accuracy and the FAR of them even cannot be adjusted below 2% due to the fact of lacking valid samples. Thus, we do not plot all

the figures starting from the first month. In addition, the curves after the 21*st* month are omitted since all models tend to stabilize their performance after the 15*th* month. As we expected, the offline RF model shows better prediction performance than SVM and DT models. Furthermore, the experiments on dataset *STA* show that our ORF model can converge rapidly (within six months) to the performance of the offline RF and achieves stable FDRs of 93-99% after the first six months. Note that in the 6*th* month, there are only 299 positive samples (45 failed disks) in the training set, which account for only 4% of all positive samples. However, the ORF model still achieves desirable prediction performance. This observation can demonstrate the applicability of our ORF model in small-scale data centers, where the valid data is very limited. The results in

Figure 3 also indicate that our ORF model can perform comparable performance against offline RF and outperforms the other two offline algorithms. In summary, our ORF model demonstrates its superiority over the offline counterparts for accurately predicting disk failure even with highly unbalanced datasets.

## Figure 3: Failure detection rates of ORF and offline models on dataset *STB*. All points on each curve ensure FARs around 1.0%.

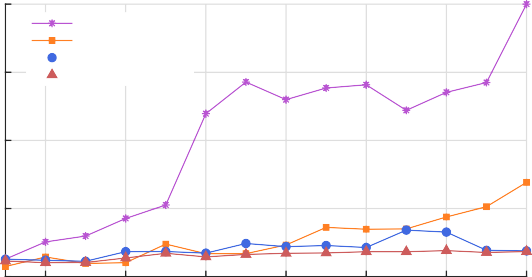
**4.5 Simulating Practical Long-term Use**

Having understood the performance improvement of ORF over the offline counterparts, we further simulate practical long-term use in data centers. Since the underlying distribution of SMART attributes changes over time, the long-term use of offline trained model without update may cause seriously decline in performance. ORF-based online learning method is claimed to evolve with the sequential arrival of data and possess adaptivity to new patterns of training data. To assess the effectiveness of the automatic evolution mechanism, we compare the ORF-based method with two update strategies for offline models: the *accumulation* update strategy and the 1-month *replacing* strategy [[14](#_bookmark15)]. Recall that the accumulation strategy updates the model periodically, *e.g.*, once a month, using all the data collected from the beginning, while the 1-month replacing strategy only uses the data collected within the current month.

To evaluate the performance of these update methods, we firstly label the samples the same way we did in Section 4.4, and then we divide all the labeled samples in a dataset according to their timestamp and each subset contains the samples collected in the same month. For convenience, we denote the *ith* month’s subset of

ORF vs Offline update for RF

8



No updating

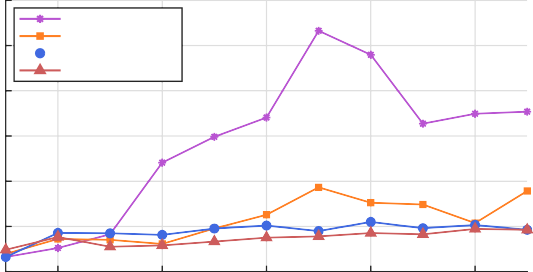
1-month replacing Accumulation

ORF

6

ORF vs Offline update for RF

12



No updating

1-month replacing

Accumulation ORF

10

8

4 6

FAR(%)

FAR(%)

4

2

2

0

8 10 12 14 16 18 20

Number Of Months

0

6 8 10 12 14

Number Of Months

## Figure 4: FARs of ORF and monthly updated RFs on dataset

***STA*.**

*STA* as *STAi* , where *i* 1, . . . , 39 . Without any update strategy, we train the offline model using samples collected within the first few months (*i.e.*, first six months for *STA* and first four months for *STB*) and then apply it to test the samples collected in the following months. The accumulation update strategy uses all the samples in *STA*1, . . . , *STAi* 1 to train the offline model, while the 1-month replacing strategy uses only the samples in *STAi* 1, and then the updated models are applied to test *STAi* , where *i* = 7, . . . , 39. As for our ORF-based model, we directly apply the model trained in *i* 1 *th* month to test *STAi* and no offline retraining is needed. Note that the way we evaluate the models in the *ith* month is different (mainly on the test set) from what we did in the previous subsection. We previously focused on comparing the ability of different models when they were used for disk failure prediction, and here we want to evaluate the actual performance of the models in practical use.

−

∈ { }

{ }−

( − )

We test the ORF model and different offline update strategies for

RF model on both dataset *STA* and *STB*. Figure 4 and Figure 5 show the FARs of the two models when they are in long-term use. Figure 6 and Figure 7 shows the FDRs of the models. As can be seen, if the RF model is used without updates, its FAR will gradually increase along with the descent of FDR. This illustrates that offline models may gradually lose their effectiveness for disk failure prediction as time goes on. Note that a FAR greater than 5% is unacceptable because it implies too many false alarms and results in heavy processing cost. We can conclude that it is essential for offline trained prediction models to be updated periodically, which is also confirmed by work [[14].](#_bookmark15)

The accumulation update strategy works, since retrained models can well fit the current training data and achieve good prediction performance in the near months. Figure 6 shows the FDR of these monthly updated RF models varies obviously from 93%-100%. This is because there are significant differences between the number

of failed disks and the number of unpredictable failures1 in each

month. By contrast, the 1-month replacing strategy performs a little worse, especially in stability. One possible reason is that the newly updated models are trained with only the samples collected in one month and thus have a less robust estimate of the statistics.

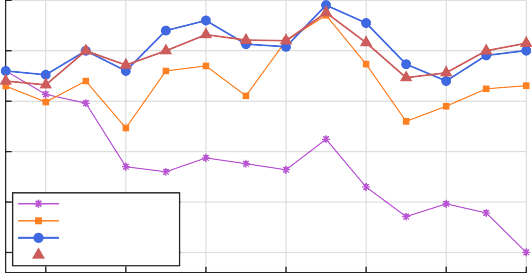
1 The SMART attributes of these disks do not show obvious fault characteristic. These disk failures are mainly caused by mechanical or electronic component problems.

## Figure 5: FARs of ORF and monthly updated RFs on dataset

***STB*.**

ORF vs Offline update for RF

100



No updating

1-month replacing

Accumulation

ORF

95

90

FDR(%)

85

80

75

8 10 12 14 16 18 20

Number Of Months

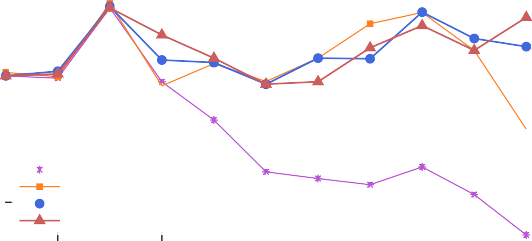
## Figure 6: FDRs of ORF and monthly updated RFs on dataset

***STA*.**

ORF vs Offline update for RF

95

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  |
|  |  |  | |  |  |  |
|  |  |  | |  |  |  |
|  |  |  | |  |  |  |
|  |  |  | |  |  |  |
| No updating | | |  |  |  |  |
| 1-month replacing | | |
| Accumulation | | |
|  |  |  |  |
| ORF | | |

90

85

80

FDR(%)

75

70

65

60

6 8 10 12 14

Number Of Months

## Figure 7: FDRs of ORF and monthly updated RFs on dataset

***STB*.**

Compared with these update strategies for RF model, it is shown that our ORF-based algorithm can maintain reasonably lower FARs while achieve comparable FDRs with these periodically updated RF models. Moreover, no model retraining is required after the initial deployment.

# CONCLUSION

In this paper, we present a novel way for proactive fault tolerance using the online learning method. There exist two major challenges for training disk failure prediction models in online mode: 1) how to label the sequentially gathered samples on-the-fly? 2) how to over- come the highly imbalance distribution of healthy and failed disks? For the former, we introduce an automatic *online label* method which temporarily keep the samples collected in the latest days to be unlabeled until a disk failure occurs or new samples arrive. For the latter, two Poisson distributions are proposed to model the sequential arrival of positive and negative samples, and the negative samples have a smaller chance of being selected by the model for update. Consequently, negative samples for training can be comparable with the size of positive samples.

Additionally, an ORF-based prediction model is built to support our method. Compared to the offline models proposed by previ- ous works, our ORF model has lower memory requirements, good robustness against label noise, and better prediction performance. The experiments on real-world datasets show that our ORF models can achieve FDRs of 93-99% with reasonable low FARs. In addition, our ORF model possesses special superiority for practical use, as it can perform automatic update with sequential arrival of data in real-time and thus is highly adaptive to the dynamic distribution of training data. As a result, our online learning method can get rid of the model aging problem and no offline update is needed. To evaluate the effectiveness of our method, we simulate the practical long-term use of ORF models and compare it with update strate- gies for offline RF models. The results show that our ORF-based algorithm can maintain reasonably lower FARs while achieve com- parable FDRs with these periodically updated RF models. Thus, we demonstrate the ability of our online learning method on main- taining stable prediction performance for the practical long-term use.

Finally, although our method is built and evaluated on two disk models from Seagate, it can be easily applied to other disk models and manufacturers as long as SMART is supported. Moreover, our method should work for a wide range of detection applications where the training data becomes available sequentially or it is necessary for the algorithm to automatically adapt to new patterns of data.

# ACKNOWLEDGMENTS

This work is supported by National Key Research and Development Program under Grant 2018YFB1003600, National Science Founda- tion of China under Grant No.61702203, Science and Technology Planning Project of Guangdong Province in China under Grant 2016B030305002, and Pre-research Project of Beifang under Grant FFZ-1601.

# REFERENCES

1. B. Schroeder and G. A. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proceedings of 5th USENIX Conference* *on File and Storage Technologies (FAST’07)*, pp. 1–16, Feb. 2007.
2. I. Manousakis, S. Sankar, G. McKnight, Thu D. Nguyen, and R. Bianchini. Environ- mental Conditions and Disk Reliability in Free-Cooled Datacenters. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST’16)*, pp. 53– 65, Feb. 2016.
3. Data center downtime costs. [*http://www.emerson.com/en-us/News/Pages*](http://www.emerson.com/en-us/News/Pages) */Net-* *Power-Study-Data-Center.aspx.*
4. L. N. Bairavasundaram, G. R. Goodson, B. Schroeder, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. An analysis of data corruption in the storage stack. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies* *(FAST’08)*, Feb. 2008.
5. E. Brewer, L. Ying, L. Greenfield, R. Cypher, and Theodore. Disks for Data Centers. Keynote talk for *FAST’16*, Technical report, Google, Feb. 2016.
6. P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High- performance, reliable secondary storage. In *ACM Computing Surveys (CSUR)*, vol 26, no. 2, pp. 145–185, June, 1994.
7. S.M.A.R.T. *https://en.wikipedia.org/wiki/S.M.A.R.T.*
8. G. Hamerly and C. Elkan. Bayesian approaches to failure prediction for disk drives. In *Proceedings of the 18th International Conference on Machine Learning* *(ICML’01)*, pp. 202–209, Jun. 2001.
9. G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan. Improved disk-drive failure warnings. In *IEEE Transactions on Reliability*, vol. 51, no. 3, pp. 350–357, Sept. 2002.
10. J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple instance application. In *Journal of* *Machine Learning Research*, vol. 6, pp. 783–816, May. 2005.
11. B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma. Proactive drive failure prediction for large scale storage systems. In Proceedings of *29th IEEE Conference on Massive* *Storage Systems and Technologies (MSST)*, pp. 1–5, May. 2013.
12. Y. Wang, Q. Miao, E. W. Ma, K.-L. Tsui, and M. G. Pecht. Online anomaly detection for hard disk drives based on mahalanobis distance. In *IEEE Transactions on* *Reliability*, vol. 62, no. 1, pp. 136–145, Mar. 2013.
13. Y. Wang, W.M. Ma, W. S. Chow, and K.-L. Tsui. A Two-Step Parametric Method for Failure Prediction in Hard Disk Drives. In *IEEE Transactions on Industrial* *Informatics*, vol. 10, no. 1, pp. 419–430, Feb. 2014.
14. B. Zhu, G. Wang, X. Ji, J. Li, and Y. Jia. Hard Drive Failure Prediction Using Classification and Regression Trees. In Proceedings of *44th IEEE Conference on* *Dependable Systems and Networks (DSN)*, pp. 383–394, Dec. 2014.
15. C. Xu, G. Wang, X. Liu, D. Guo, and T. Liu. Health Status Assessment and Failure Prediction for Hard Drives with Recurrent Neural Networks. In *IEEE Transactions* *on Computers*, vol. 65, no. 11, pp. 3502-3508, Nov. 2016.
16. J. Li, R. J. Stones, G. Wang, X. Liu, Z. Li, and M. Xu. Hard drive failure prediction using Decision Trees. In *Reliability Engineering and System Safety*, vol. 164, pp. 55–65, Mar. 2017.
17. J. Li, R. J. Stones, G. Wang, Z. Li, X. Liu, and X. Kang. Being accurate is not enough: New metrics for disk failure prediction. In *Proceedings Symposium on* *Reliable Distributed Systems (SRDS)*, pp. 71–80, Dec. 2016.
18. O. Fontenla-Romero, B. Guijarro-Berdinas, D. Martinez-Rego, B. Perez-Sanchez, and D. Peteiro-Barral. Online machine learning. In *Efficiency and Scalability* *Methods for Computational Intellect*, pp. 27–54, 2013.
19. The backblaze hard drive data and stats. [*https://w*](http://www.backblaze.com/b2/hard-drive-)*ww*[*.backblaze.com/b2/har*](http://www.backblaze.com/b2/hard-drive-)*d-driv*[*e-*](http://www.backblaze.com/b2/hard-drive-) *test-data.html*.
20. M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann. Predicting Disk Replace- ment towards Reliable Data Centers. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 39–48, Aug. 2016.
21. N. C. Oza and S. J. Russell. Online bagging and boosting. In *Artificial Intelligence* *and Statistics*, pp. 105–112, 2001.
22. A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line Random Forests. In Proceedings of *12th International Conference on Computer Vision* *Workshops (ICCV)*, pp. 1393–1400, 2009.
23. V. Agrawal, C. Bhattacharyya, T. Niranjan, and S. Susarla. Discovering rules from disk events for predicting hard drive failures. In Proceedings of *International* *Conference on Machine Learning and Applications*, pp. 782–786, Dec. 2009.
24. L. Breiman. Random forests. *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.
25. F. Mahdisoltani, I. Stefanovici, and B. Schroeder. Proactive error prediction to improve storage system reliability. In *Proceedings of the 2017 USENIX Annual* *Technical Conference (USENIX ATC’17)*, pp. 391–402, Jul. 2017.
26. L. Breiman. Out-of-bag Estimation. *Citeseer*, 1996.
27. E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *Proceedings of 5th USENIX Conference on File and Storage Technolo-* *gies (FAST’07)*, pp. 17–29, Feb. 2007.
28. C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. In *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, Apr. 2011.
29. Fit binary classification tree for multiclass classification.

*https://cn.mathworks.com/help/stats/fitctree.html*.