2018 IEEE International Congress on Big Data

Large Scale Predictive Analytics for Hard Disk Remaining Useful Life Estimation

Preethi Anantharaman, Mu Qiao, Divyesh Jadav
IBM Almaden Research Center
San Jose, California, CA
Email: {Preethi.Anantharaman1, mqiao, divyesh}@us.ibm.com

***Abstract*—Hard disk failure prediction plays an important role in reducing data center downtime and improving service reliability. In contrast to existing work of modeling the prediction problem as classification tasks, we aim to directly predict the remaining useful life (RUL) of hard disk drives. We experiment with two different types of machine learning methods: random forest and long short-term memory (LSTM) recurrent neural networks. The developed machine learning models are applied to predict RUL for a large number of hard disk drives. Preliminary experimental results indicate that random forest method using only the current snapshot of SMART attributes is comparable to or outperforms LSTM, which models historical temporal patterns of SMART sequences using a more sophisticated architecture.**

***Keywords*-predictive analytics, deep learning, remaining useful life, hard disk drive**

I. Introduction

Hard disk drive (HDD) has been a primary type of storage in data centers. With the advance of latest HDD technology, Western Digital predicts that 70% of all data will be stored in HDDs and 90% of all data center data will be in HDDs by 2020 [1]. Hard disk failure is an important contributor to data center downtime, leading to significant business cost. Therefore, it is critical to identify disks that have short remaining useful life (RUL) so that IT service admins can take proactive actions.

SMART (Self-Monitoring, Analysis and Reporting Technology) monitoring has been widely used by HDD manufacturers to determine when disk failures are likely to happen. SMART collects disk sensor data, such as temperature and sector error rates, and deploys disk with embedded proprietary predictive models. These models are often simple threshold-based, with very low false alarm rates and weak predictive power. Advanced prediction models are later developed to determine whether a disk will fail or not in the future based on SMART attributes. This task is usually modeled as a binary classification problem [2] [3].

In this paper, we develop machine learning models to predict the RUL for a large number of hard disks. While existing works on hard disk failure prediction are focused on classifying whether a disk will fail or not, we aim to predict the exact value of RUL. This predictive analytics provide IT service admins deeper insights into disk conditions and enable better disk replacement planning and management. We apply two types of machine learning models to predict the RUL: (1) random forest using the current snapshot of SMART readings, and (2) long short-term memory (LSTM) neural networks, a type of recurrent neural networks (RNN), modeling the historical temporal pattern of SMART attributes. Most recently, [4] also applies LSTM to predict RUL of hard disk drives, where RUL is first binned and the prediction is modeled as a multi-classification task. Different from their work, we directly predict the value of RUL via a regression approach. In addition, we are deploying the machine learning models on a large number of hard disks from different disk manufactures.

II. Machine Learning Models

We introduce two types of machine learning models used in the RUL prediction: random forest, a traditional machine learning method, and LSTM, a deep neural network, which is widely used to model sequential data.

A. Random Forest

Random forest (RF), as an ensemble method, combines the predictions of several decision tree based estimators. Each tree in the ensemble is built from sub-samples drawn with replacement from the training data. When splitting a node during the construction of the tree, the split is chosen as the best split among a random subset of the features. As a result of this randomness, the variance of the forest is reduced due to averaging, hence yielding better predictive accuracy and mitigating over-fitting. Random forest has been applied to predict hard disk failure in a binary classification task [2]. In our work, we apply RF as a regressor on the RUL based on current SMART readings.

B. Recurrent Neural Networks

Recurrent neural networks (RNN), which accept data with temporal or sequential dependencies, have been applied in various image processing, computer vision, and natural language understanding tasks. For example, RNN is used to tag word sequences or predict the next word given the input of a sequence of words that appear before the target prediction [5]. We apply RNN to model the temporal pattern of SMART attributes and predict RUL given the input of SMART sequences.

However, the training of RNNs usually faces the vanishing or exploding gradient problems. The magnitude of

978-1-5386-7232-7/18/$31.00 ©2018 IEEE
DOI 10.1109/BigDataCongress.2018.00044

the gradient can increase or decrease exponentially within the beginning or end of the sequence of the network, when the output prediction is conditioned on long distance features [6]. Long short-term memory recurrent neural networks have been proposed to utilize memory based gates to help mitigate these issues. An LSTM memory cell is illustrated in Figure 1.

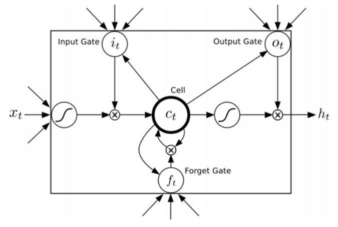


Figure 1: LSTM cell structure [7]

The memory cell acts as an integrator over time. Suppose the input data at time t is x_t and the hidden state at the previous time step is h_{t-1}, then the memory cell at time t has the value:

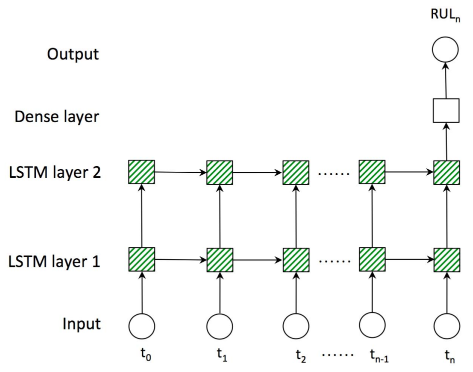
$$c_t = \alpha \otimes c_{t-1} + \beta \otimes g(x_t, h_{t-1})$$

where $\otimes$ is denoted as an element wise multiplication between two vectors. c_t is a linearly weighted combination between c_{t-1} and $g(x_t, h_{t-1})$, and is computed additively. Therefore, if the gradients of the error during the back propagation through time (BPTT) cannot pass through the function g at time step t, it has an opportunity to be propagated backwards further through the previous cell time step, c_{t-1}. To prevent c from exploding, g is often associated with a sigmoid or hyperbolic function (tanh) function. The memory cell allows another path for the gradient of the error to flow, which effectively solves the vanishing or exploding gradient problem occurring in RNN architectures. Specifically, the LSTM cell in Figure 2 is implemented as:

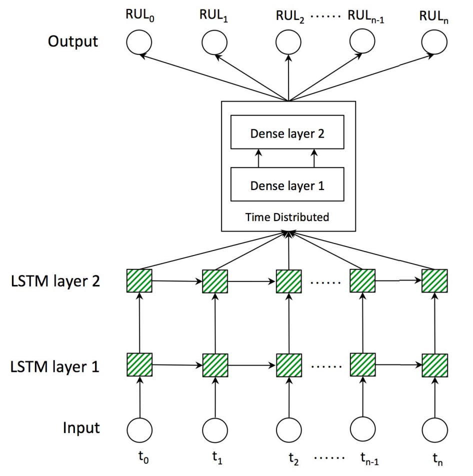
$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes g(x_t, h_{t-1}) \\
g(x_t, h_{t-1}) &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \\
h_t &= o_t \otimes \tanh(c_t) \ ,
\end{aligned}$$

where σ is the logistic function, i, f, o, and c are the input gate, forget gate, output gate, and cell vectors, respectively, and W is the weight matrices.

We design two different LSTM models. Figure 2(a) shows the LSTM many-to-one (LSTM-M2O) model, where two stacked LSTM layers, followed by a dense layer, are used to predict the RUL at the time step t_n. Figure 2(b) shows the LSTM many-to-many (LSTM-M2M) model, where an RUL is predicted at each time step. LSTM-M2M also has two stacked LSTM layers, where a time distributed two stacked dense layers are applied to the hidden state of each cell in the second LSTM layer.



(a) LSTM many-to-one model



(b) LSTM many-to-many model

Figure 2: LSTM model architectures

III. Experiment

We apply random forest and LSTM to predict the RUL of a large number of hard disk drives based on the SMART attributes. We start with dataset preparation, introduce our experiment setup, and provide our preliminary results below.

A. Data Preparation

1) Data Set: The data set used for our analysis is the publicly available Backblaze hard drive data [8]. It contains hard drive information along with daily SMART attributes collected for over 91,305 hard drives in a datacenter. We extract failed disk information from Seagate for a period of over 16 months (from October 2016 to December 2017). For each disk model, we collect SMART attributes for over 200 disk drives that had reported failure over this period.

The SMART attributes from every unique disk drive form a multivariate time series with daily time steps. We focus our experiment on seagate disk model ST4000DM000, as the number of disk failure reported was high compared to other models.

2) *Data Cleanup and Normalization:* The dataset for each hard disk contains information about the serial number, model, hard disk capacity, failure flag (0 is "OK" and 1 is "failure") and 90 SMART attributes. As these attributes are manufacturer specific, we observe that many of them have null values. We transform all the null values to zeros. The range of values collected varies by model and serial number. We group disks having the same model numbers, and scaled the SMART attributes at each time step to a value between 0 and 1.

3) *RUL Generation:* As we are dealing with time series of SMART attributes for individual disk, each disk drive is assumed to be operating normally at the start of each time series. It exhibits a change in performance over the time steps and eventually fails at the last step in the time series. We generate the RUL for each time step, which provides the remaining working days a drive would last before it fails. In addition, we also evaluate our models using a piece-wise RUL target generation function, which limits the maximum RUL to a constant value and begins to degrade linearly after a certain point in time. Specifically, the piece-wise function is implemented as:

$$RUL = \begin{cases} pw, & t \leq tmr - pw \ . \\ -t + tmr, & t > tmr - pw \ . \end{cases}$$

where tmr is the true maximum RUL, pw is the piece-wise maximum RUL, and t is the time step. The reason that we use a piece-wise linear RUL instead of a linear degrading one is that when the disk drive is still in "good" condition, estimating RUL may not be of significance. Therefore, we can set the RUL during such phase to a constant piece-wise maximum value and have it linearly degrading after certain time point. The piece-wise RUL generation is also used in [9].

4) *Data Transformation:* The multivariate time series data for a disk drive is modified into three dimensional array of sequences, time steps and features, where each time step consists of multiple SMART attributes. In addition, we have implemented a sliding window mechanism to augment the total number of time series in the training data.

B. Building Machine Learning Models

We develop two different LSTM models, as shown in Figure 2. LSTM-M2O has the many-to-one architecture, where multivariate time series disk data are grouped into sequences of predefined length. These sequences are mapped to a single output value, which corresponds to the RUL at the last time step in each sequence. LSTM-M2O has two stacked LSTM layers, the cell of which has 64 units (aka neurons), followed by a single dense layer. The second model, LSTM-M2M, is time distributed, where the SMART attributes feature vector at each time step in the sequence is mapped to a corresponding RUL value. Hence, each sequence has multiple output values. LSTM-M2M also has two stacked LSTM layers, which have the same number of units (i.e., 64) in each cell as LSTM-M2O. The unit number of each time distributed dense layer is set to 8. We also apply dropout to avoid overfitting.

The LSTM models are trained using the three-dimensional array of sequences, time steps, and features, which are obtained from the data transformation step in III-A4. To train random forest, we leverage the same input data as LSTM. As random forest takes feature vectors as input, we only use the SMART attributes at the last time step of each time series and its corresponding RUL for training. LSTM models the historical temporal patterns in time series while random forest uses the current snapshot of SMART attributes. We implement random forest using the Python scikit-learn package and LSTM using Keras with Tensorflow as the backend. All The experiments are conducted on a NVIDIA Tesla P40 GPU. We use the RMSprop optimizer to estimate the parameters. For random forest, we use the default setting and empirically set the number of trees to 100. The training of random forest is significantly faster than LSTM.

C. Experiment Results

We train the machine learning models on the SMART attributes from one set of hard disks and test the models on a different set of disks belonging to the same disk model. The total number of hard disk drives in both training and test data are around 100. All the 90 SMART attributes are used as the input features. We compare the performance of random forest and two LSTM models (i.e. LSTM-M2O and LSTM-M2M) in terms of their mean absolute error (MAE) on the test data.

Table I: Mean absolute error (MAE) of random forest (RF) and LSTM

Models	Setup	LSTM	RF
M2O	no sliding window	27.62	22.66
M2O	sliding window (size=10)	24.81	24.08
M2M	no sliding window	23.26	19.55
M2M	sliding window (size=10)	29.04	21.10
M2M	piece-wise RUL - no sliding window	8.15	6.40
M2M	piece-wise RUL- sliding window (size =10)	9.31	7.86

The length of each sequence, or the number of steps in the time series, is set to 50. To augment the total number of sequences, we apply a sliding window mechanism. A smaller sliding window length will result in more overlapping between sequences as well as more sequences. We compare the performance of these models with and without sliding window augmentation. In addition, we also compare the model performance difference with and without piece-wise RUL target generation. The results of models under different setups are summarized in Table I. Without sliding window augmentation, we obtain 666 training sequences and 268 test sequences. A sliding window of size 10 results in 2,628 training sequences and 1,094 test sequences. In order to have sequences of the same length and avoid dropping any significant time steps, zeros padding is performed at the end of the hard disk life. We randomly split the test data into 75% as the final test samples and 25% as the validation samples for LSTM hyper-parameter tuning. For LSTM models, the number of batch size is 10, the number of epochs is 300, and the learning rate of the RMSprop optimizer is 0.001.

From Table I, we can see that LSTM-M2O has an MAE score of 27.62 with no sliding window and 24.81 with a sliding window of size 10. For random forest, the MAE scores are 22.65 and 24.08 with and without sliding windows, respectively. LSTM-M2O and random forest models have similar performance with a sliding window of size 10. For LSTM-M2M, it achieves an MAE score of 23.26 with no sliding window and an increased score of 29.04 when the sliding window is of size 10. As a comparison, the random forest model shows better performance with a score of 19.55 without sliding window and 21.55 with a sliding window. Note that for M2M, the corresponding random forest model also predicts an RUL at each time step. The MAE is calculated over all the steps.

In M2M, the RUL value of hard disks has degraded linearly at each time step. We also generate the piece-wise RUL using the approach introduced in Section III-A3, where the RUL is set to a piece-wise maximum constant value and degrades linearly after a certain time point. Specifically, we set the piece-wise maximum RUL to be 100. We observe a significant improvement in the MAE scores for both LSTM and random forest. With piece-wise RUL, the MAE of LSTM-M2M is reduced from 23.2 to 8.15 without sliding window and from 29.04 to 9.31 with the sliding window. Random forest achieves the lowest MAE score of 6.4 without sliding window and a score of 7.86 with a sliding window. Comparing with the LSTM models, random forest shows comparable or better performance for all the setups. While the sliding window mechanism can augment the size of training data, it does not improve the prediction performance of both LSTM and random forest, except for LSTM-M2O.

IV. Conclusion and Future Work

We apply machine learning models to predict the remaining useful life (RUL) of hard disk drives. Different from existing work of hard disk failure prediction in a classification setup, we directly predict RUL via regression approaches. Preliminary experimental results indicate that random forest using the current snapshot of SMART readings has comparable or better performance than LSTM, which models the sequential pattern of SMART attributes. We are deploying the machine learning models for RUL predictions on a large number of hard disks and investigating if the performance difference also happens in other disk models. In the current work, we do not observe a superior performance of LSTM. One reason may be that the current time sequences of SMART attributes used in the training data do not carry strong predictive power with respect to RUL. Our time series data spans over a period of 16 months, a relatively long period. We are investigating if the time sequence near disk failure may have more significant patterns in terms of RUL prediction. In addition, we use daily SMART attributes, which may not be stable due to the recovery mechanisms embedded in the disk. It may be desirable to use a smoothing mechanism to aggregate the SMART values over a specific time window. More careful feature selection may also help improve the performance of LSTM. The application of deep learning on multivariate time series modeling is an active research area, which requires more innovative neural network architecture design.

References

[1] C. Tom, "MAMR Hard Disk Drives Enable Future Data Centers," [Online] https://www.forbes.com/sites/tomcoughlin/2017/10/11/mamr-hard-disk-drives-enable-future-data-centers/.

[2] M. Botezatu, I. Giurgiu, J. Bogojeska, and D.Wiesmann, "Predicting Disk Replacement towards Reliable Data Centers," in *Proceedings of the ACM SIGKDD Conference,* pp. 39-48, 2016.

[3] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado, "Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application," *Journal of Machine Learning Research*, vol. 6, pp. 783–816, 2005.

[4] F. D. Lima, G. M. Amaral, L. G. Leite, J. P. Gomes and J. Machado, "Predicting Failures in Hard Drives with LSTM Networks," in *Proceedings of the 6th Brazilian Conference on Intelligent Systems,* pp. 222-227, 2017.

[5] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179-211, 1990.

[6] A. Graves and J. Schmidhuber, "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures," in *Proceedings of the IEEE IJCNN Conference*, pp. 2047-2052, 2005.

[7] "Long short term memory," [Online] http://blog.otoro.net/2015/05/14/long-short-term-memory/.

[8] Backblaze, "Hard Drive Data and Stats," [Online] https://www.backblaze.com/hard-drive-test-data.html.

[9] S. Zheng, K. Ristovski, A. Farahat and C. Gupta, "Long Short-Term Memory Network for Remaining Useful Life estimation," in *Proceedings of the IEEE ICPHM Conference,* pp. 88-95, 2017.