Vietnam Journal of Science and Technology ***55*** *(5) (2017) 557-571*

DOI: 10.15625/2525-2518/55/5/8582

A DATA-DRIVEN FRAMEWORK FOR REMAINING USEFUL LIFE ESTIMATION

Nguyen Dinh Hoa

Posts and Telecommunications Institute of Technology, 122 Hoang Quoc Viet St., Cau Giay Dist., Ha Noi, Viet Nam

Email: *hoand@ptit.edu.vn*

Received: 2 August 2016; Accepted for publication: 5 April 2017

ABSTRACT

Remaining useful life (RUL) estimation is one of the most common tasks in the field of prognostics and structural health management. The aim of this research is to estimate the remaining useful life of an unspecified complex system using some data-driven approaches. The approaches are suitable for problems in which a data library of complete runs of a system is available. Given a non-complete run of the system, the RUL can be predicted using these approaches. Three main RUL prediction algorithms, which cover centralized data processing, decentralize data processing, and in-between, are introduced and evaluated using the data of PHM'08 Challenge Problem. The methods involve the use of some other data processing techniques including wavelets denoise and similarity search. Experiment results show that all of the approaches are effective in performing RUL prediction.

Keywords. Remaining useful life, prognosis, structural health management, wavelets denoise, similarity search, principal component analysis.

1. INTRODUCTION

Remaining useful life (RUL) estimation is one of the most common studies in the field of prognostics and structural health management. The aim of RUL prognosis is to estimate the RUL of a system, given a short historic measurement, using either a prediction model or a data-driven technique. It helps provide an acknowledgment about the working time of the system so that appropriate maintenance or replacement actions may be scheduled prior to the failure of the system. The prediction accuracy plays an important role in reducing unnecessary maintenance, such as early replacement of components, or production downtime due to unexpected machine failures [1]. RUL of one system or components can be estimated either directly by using a multivariate matching process or indirectly by damage estimation followed by extrapolation of damage progression [2]. Definitions of both damage and a system failure criterion are main obstacles that prevent the latter approach from being used widely in all problems despite its ability to align with engineering matters. Directed approaches for RUL prognosis relate to data-driven techniques, which are based on the availability of past observations to estimate the RUL of a system at the current run. However, a dynamic system can only be observed indirectly by using data in time series or features extracted from available measurement processes such as temperature, vibrations, pressure, etc. This leads to the fact that the reliability of the predictions depends on two main factors: the accuracy of the gathered data (how closely the data describe the working states of a system) and the appropriateness of the prediction methods applied to the given data in a given condition. The availability of run-to-failure data plays an important role in data-driven prognosis. However, there are not many public data repositories that provide available run-to-failure data to enable a comparative analysis of various prognosis algorithms. This explains why there are not many data-driven techniques developed for RUL estimation so far. In circumstances where run-to-failure data are provided, data-driven techniques are good solutions for RUL estimation. That leads to a need to develop a reliable data-driven framework for RUL estimation.

As one branch of general prognosis methods, RUL prediction algorithms can be roughly classified into two categories: model based prediction and data-driven based prediction. In data-driven prediction methods, past data are exploited to build a library of complete runs or a library of system itineraries. Then, prediction techniques are applied to estimate the RUL of the system. Model based estimation methods tend to rely on a degradation model of the system, based on which RUL is estimated by using predefined failure criteria. Generally, the difference between these two types of RUL prognosis is not very clear. Most of the prediction algorithms introduced in literature are the combination of the two methods.

Thanks to a wide range of applications, model-based RUL prediction algorithm attract a lot of attention. Xue *et al.* [3] introduce an instance-based model to evaluate the RUL of aircraft engines. In this method, a four-step algorithm mainly based on similarity functions is applied. They focus on using instances of the past runs that are similar to the current operation to create an ensemble of local models. These four steps retrieve similar instances from the database, evaluating similar measures between the test sample and the instances using the truncated generalized Bell function, creating local models using the most similar instances, and aggregating outputs of local models using a locally weighted model. Yan *et al.* [4] develop a prognostic method to detect system degradation for RUL estimation. Logistic regression with or without enough historical data using maximum likelihood techniques is used to classify the machine running conditions from normal to failure. The performance of the machine is estimated at each cycle of life, and then an ARMA model is used to predict the tendency of future performance based on previous assessment results. Consequently, the number of remaining life cycles to failure is derived. Shao and Nezu [5] propose a progression-based prediction of RUL using different methods in different bearing running stages. This algorithm includes online modeling of the running state using neural networks and logic rules. It allows a multi-step prediction, which helps the approach adapt to changes in environment. Liao *et al.* [6] develop two models for RUL estimation: the proportional hazards model and the logistic regression model. These models evaluate the multiple degradation features of sensor data compared to specific reliability indices of the system in order to predict its RUL. Kiddy [7] uses the Monte Carlo simulation to create conservative predictions based on known component reliability to estimate the RUL of the component. Saha *et al.* [8] develop an RUL prediction model using a relevance vector machine based on a Bayesian part of a kernel- based regression technique, or a support vector machine. The model is then used in a particle filter framework. The RUL is estimated in the form of a probability density function under the process of statistical estimation of the noise and operational conditions.

Pattipati *et al.* [9] predict the RUL of the battery using a moving average Support Vector Machine Regression for different thresholds on capacity fade and power fade.

In addition to model based prediction, many data-driven techniques for RUL estimation have been developed and applied to solve a lot of real problems. Cheng and Pecht [10] combine the Multivariate State Estimation Technique (MSET) with life cycle damage prediction to produce a method for RUL estimation. In this method, MSET algorithms monitor the relationship between actual data and the expected data, which is based on the historic data covering complete runs of the system. Based on this relationship, the cri- terion of the failure of the system is set, based on which RUL can be estimated. Byington *et al.* [11] present a data-driven method using a neuron network to predict the RUL. In this method, the health state of the system is modeled using specific data features, which are used within a classification environment to determine the true health state of the monitored system. A prognostic function is used to store the classification and fused information during the whole operating life of the system to estimate the remaining useful life under some specified bounds. This method mainly relies on fault detection and failure prediction beyond the point of the fault. Kavanagh *et al.* [12] develop a two-stage RUL estimation algorithm. First, they use the envelop analysis for feature extraction, then the mutual information for feature subset selection. Second, they develop a degradation model and locate the current position of the machine in that model. The nearest neighbor classification algorithm is used to determine the RUL of the machine. Lao *et al.* [13] design a two-layer neuron network (NN) to predict the bearing's RUL based on time-frequency features. The signal's frequency spectrum and feature extraction are first analyzed by DFT and PSD. The first layer of the network is used to identify the bearing as one of three states: normal, unbalance failure, and other failures. The second network layer is designed for RUL prediction using a continuous unipolar nonlinear function. Liu *et al.* [14] improve the efficiency of NN based prediction method by using data clustering. In their method, the data are classified into small clusters, having similar data. The clustering algorithm being used is fuzzy C-means. Data in each cluster then are used to train corresponding NN for prediction. The input data is first judged to be in one of defined cluster, then desired information (system state, RUL, ...) is predicted by corresponding NN.

In summary, RUL prediction based on knowledge about damage progressions and failure criteria is more commonly used in research fields. However, due to difficulties in determining the definitions of the damage of the system as well as the stopping criterion, this prediction method is not very useful in some of the problems that require data analysis on available run-to-failure data. In many dynamic engineering systems, the faults that shorten the life time of the system are not always the same in all cases. This leads to variations in determining the damage propagation as well as the stopping condition of the system. In this case, modifications applied on some data-driven prognosis techniques can be done to solve specific problems. In this research, three data-driven RUL prediction approaches are investigated based on similarity search algorithms covering both centralized and decentralized data processing. Performances of all three proposed methods are evaluated using a PHM Challenge data. The experiments show that those algorithms, with sufficient modifications, are useful for RUL estimation in real problems.

The remainder of the paper is organized as follows. The problem of RUL estimation is formulated in Section 2. Section 3 addresses the general framework for RUL estimation, that includes three main approaches, such as centralized processing, decentralized processing, and hierarchical processing. All three methods are evaluated using the dataset from the Prognostics and Health Management 2008 Challenge. Experimental results and discussion are presented in Section 4. Conclusions are provided in Section 5.

2. PROBLEM FORMULATION

The system dynamics under the consideration are usually very complex. In most cases, there is no closed-form physical models available for direct prognosis. For the problem of RUL estimation, data-driven techniques mainly rely on the availability of run-to-failure data. Assume that the data consists of measurements collected from N sensors distributed widely in the system. The system here can be either a stand-alone engineering machine or a large-scale multi-model system. The sensors, which are denoted by S_n, $n = 1...N$, measure the outputs of the system to build up response surfaces and operation margins. These outputs in time series describe the characteristics of the system, e.g., they can be the measurements of the temperature, vibrations, pressure, etc. The number and types of sensors required as well as the locations of these sensors are chosen such that the system health state can be estimated accurately based on the data collected. Run-to-failure data of the system consists of time series recorded from a starting point, which is assumed to be in a normal working condition of the system, until the system goes down due to the faults occurring sometime during the work and developed in magnitude causing the system failure. Using the discrete time representation, the data collected by sensor S_n can be expressed as $\left\{S_n^{-(T-1)}, \dots, S_n^0\right\}$. The superscript "0" represents the end of lifetime of the system, and the system lifetime is T. Assume that in the training data set, there are a total of M complete runs, each of which has a life span of T_j. In short, the whole training data can be represented by $\left\{S_n^{t,j}\right\}$, $n = 1, \dots, N$, $t = -(T_j - 1), \dots, 0$, and $j = 1, \dots, M$. Based on the information presented in the training data set composed of current system measurements from N sensors for a duration T_c, $\left\{S_n^{-(T_c+T_R-2)}, S_n^{-(T_c+T_R-3)}, \dots, S_n^{-(T_R-1)}\right\}$, $n = 1, \dots, N$, our goal is estimate the RUL T_R accurately.

With the advances of sensing technology, a large number of sensors can be deployed on modern systems. Traditional centralized processing which cooperates all these sensors can be very computationally complex and resource consuming. Experiences tell us that not all the sensors provide relevant information regarding the system health state. Furthermore, some sensors may provide redundant information. On the contrary, some sensors are sensitive to the system degradation, and their data vary accordingly to the occurrence of the faults. Prognosis algorithms should be able to recognize and extract the positive contributions of the measurements from these sensors and facilitate decentralized processing.

3. GENERAL FRAMEWORK

Based on the availability of the data from the historic complete runs as training data, given the current incomplete runs as testing data, in general, it is possible to directly diagnose the similarities between test and training datasets. Similarity searching is an effective data processing technique to solve this kind of problem. Data-driven techniques based on similarity search can be centralized data processing, fully decentralized data processing, or in-between, so called hierarchical data processing. They incur different computational complexity and communication cost.

In the centralized structure, all data from N sensors are processed at a central unit. K features describing the system health state can be extracted from N sensors. This stage is so-called feature extraction. The number of dimensions of the data is, then, transformed from N to K. Similarity searching can then be applied individually on K independent features. The fusion stage fuses the RUL estimates from these K features to provide the final decision regarding systems remaining useful life.

In the fully decentralized structure, the sensor selection is conducted at the first stage based on the training data to remove all the sensors that useless for the RUL prediction. Only K most relevant sensors are kept for RUL estimation. The similarity search algorithm is applied each local sensor, providing K estimates of the RUL of the system. The RUL fusion stage combines these answers to produce a final RUL estimate.

Hierarchical data processing is a structure between the two extreme cases described above. N sensors are first classified into K groups. Each sensor group contains sensors having similar characteristics or highly correlated to each other. Important features are extracted from each sensor group, based on which system RUL can be estimated. These estimates from different sensor groups are then fused to further improve the estimation accuracy.

In the following sections, we first briefly describe the similarity search approach, then the details of three RUL estimation methods are provided.

3.1. Similarity search

Similarity search has attracted a lot of research attention recently. A brief description of the technique of "similarity search" in time series data was done by Goldin *et al.* [15]. This method has been developed and successfully applied to many applications such as economics [16, 17], video and image processing [18] and many other data-driven techniques [19. 20]. There are various ways to conduct similarity search depending on how the query sequence is compared with the database sequences [19]. The most common way is to compare the entire time series, known as full comparison, by using appropriate distance functions. In this method, all time series matching with the full length of the query are retrieved. Another approach is subsequence matching, in which any time series in the database that match a subsequence of the query is selected. The length of the subsequence is defined depending on required applications, and the matching sequences need not to be in a common time frame. A different method from subsequence matching is interval-focused similarity search, in which the time slots as well as the predefined time intervals relevant for matching are fixed [19].

The similarity between two sequences can be measured by an appropriate distance function, such as Euclidean distance, Dynamic Time Warping, Pearson's correlation coefficient, angular separation, etc. The Euclidean distance is adopted in this research due to its simplicity. Given K features/sensors extracted/selected from the training data, the similarity search algorithm can be conducted as follows. For feature f_k , we have past complete run j of the system contained in the training data set $\left\{f_{train\_k}^{-(T_j-1)}, \dots, f_{train\_k}^{0}\right\}$ of length T, and current generalized non-complete run i in the test data set $\left\{f_{test\_k}^{-(T_{C_i}+T_{R_i}-1)}, \dots, f_{test\_k}^{-(T_{R_i}-1)}\right\}$ of length T_{C_i}. To estimate the RUL T_{R_i} , we need to search for a portion of the past run that is the most similar to the current run. The actual data set of the test unit i is $\left\{f_{test\_k}^{-(T_{C_i}-1)}, \dots, f_{test\_k}^{0}\right\}$. The last time sample of test unit i is moved forward from $t = -(T_j - 1)$ towards $t = 0$ and the Euclidean distance between the overlap parts is calculated. For a fair comparison, the similarity between these two portions is measured by the average distance per time sample,

$$D_k^{t,ij} = \frac{1}{Q}\sqrt{\sum_{x=1}^{Q}\left(f_{test\_k}^{-(x-1)} - f_{train\_k}^{-(x-t-1)}\right)^2}, t = -(T_{\boldsymbol{j}} - 1), \ldots, 0$$

where t is the time index in cycles, indicating the position of the portion of training unit j to be compared; $Q = t + T_j$, and $Q \leq T_{C_i}$ is the number of data samples of the overlap portions; T_{C_i} and T_j are the lengths of test unit i and training unit j, respectively. Time index t corresponding to the minimum value of $D_k^{t,ij}$ indicates that the portion of training data from $-(T_j - 1)$ to t is the most similar to test data. Then, the number of remaining life cycles from t to the end of training unit denotes the best estimate of RUL based on training unit j,

$$\widehat{T_{R\_k}^{ij}} = -\arg\min_t D_k^{t,ij}$$

with M training units in the data set, this procedure essentially provides M samples of RUL estimates based on the k^{th} feature. Proper fusion and estimation schemes need to be adopted.

3.2. Centralized processing

In centralized processing, all measurements are processed at a central unit. For data with a high dimension, where graphical representation is not available, patterns in data can be hard to find. Principal component analysis (PCA) [21] is a powerful tool for analyzing such data. PCA helps analyze the structure by highlighting the patterns in data. The most important advantage of PCA is that once these patterns are identified in the data, the data can be compressed to lower dimensions without much loss of information.

The principal components (PCs) are defined by an orthogonal linear transformation of data vector X. The PCA uses covariance matrix to determine K most significant patterns of training data. In fact, the limitation of this method is the sensitivity of the PCs to the measurement units used for each element of X. To overcome this disadvantage, researchers tend to use PCA based on correlation matrix [21]. However, that is not the case for the being proposed method because the main purpose of the algorithm is to find the pattern of the training data and the test data, then we find the similarity between those data. As long as the PCA is the same for both training data and test data, the results are still valid. It is shown that there are many ways to decide how many principal components should be kept in order to account for most of the variance of X. In this research, the value of K is chosen based on the desired percentage of total variation that the selected PCs contribute, let’s say more than 80%. The variance of each PC is presented by its eigenvalue of the covariance matrix. Then, the required number of PCs is determined by the number of K largest eigenvalues that chosen percentage is exeeded:

$$\sum_{k=1}^{K}\lambda_k \Big/ \sum_{n=1}^{N}\lambda_n \geq 0.80.$$

PCA is usually applied to multi-dimensional data with independent and identically distributed samples to identify a subspace with reduced dimensionality containing most information/variation of the original data. For the current RUL estimation problem, each sensor can be regarded as one dimension and we try to extract the most important subspace of dimension K from the original dimension N. However, the samples we have for each dimension/sensor are in the form of a time series. That means the samples at different time points may not be independent or identically distributed. In fact, since the system degrades with time, these time samples are not of the same distribution. The system degradation usually causes changes in mean and/or variance of the distributions of time series. In general, it may change the distribution function itself. In other words, variations in the time samples indicate the trend of system health state. PCA is a simple linear operation to locate the subspace with the most of variations of data. Moreover, the main objective of PCA is descriptive, not inferential, then non-independence does not seriously affect this purpose. Here, we apply PCA to time series to locate those dimensions with large variations. Note that we have to distinguish variations related to system health state and variations that do not.

As stated above, system degradation may cause changes in different characteristics of sensor measurements. Certain data preprocessing, e.g., proper data normalization, denoising, etc., is required before applying PCA.

RUL estimation based on PCA

Given the training dataset, which includes all the run-to-failure data in the past, the system health state in each run can be derived by concatenating all sensor measurements of each training data using PCA. K eigenvectors denoting K most significant patterns of the training data is selected to transform the data in both training and test data. By using the same eigenvectors to transform the data in both training and test units, it is ensured that all of the data is projected on the same transformed subspace. Thus, the similarity search results are reliable. After PCA transformation, there are K features in each testing data. Each feature k produces a set of M answers $\widehat{T^{ij}_{R\_k}}$ for the RUL of test data i, M is the number of training data in the library, $j \in M$. Totally, there are $K \times M$ answers for the RUL of each test data i. M answers $\widehat{T^{ij}_{R\_k}}$ then are fused by the weighted sum based on the distances D^{ij}_{k}, that determine $\widehat{T^{ij}_{R\_k}}$, as below.

$$RUL_{i\_k} = \frac{\sum_{j=1}^{M} \frac{\min(D^{ij}_{k})}{D^{ij}_{k}} \widehat{T^{ij}_{R\_k}}}{\sum_{j=1}^{M} \frac{\min(D^{ij}_{k})}{D^{ij}_{k}}}$$

The similarity between two data sequences is inversely proportional to the distance between them. Then, the training data containing the most similar portion to the test data i is believed to produce the closest answer to the actual RUL of the test data. This is denoted by the distance $\min(D^{ij}_{k})$. As a result, the reliability of the answer for the RUL of test data i given by training data j is inversely proportional to the distance $\min(D^{ij}_{k})$. In the RUL fusion stage, the final RUL$_i$ of the test unit i is the average of K answers RUL_{i_k}. The final scores based on the correct answer is used to evaluate the precision of the proposed method.

3.3. Hierarchical processing

In this approach, N sensors are divided into K clusters containing correlated sensors. These sensor clusters are maximally independent to each other. Each sensor cluster produces its own answer for RUL estimation. The final RUL answer is derived by the linear combination of results provided by those clusters. The PCA algorithm is the basis of the whole process.

Consider a linear transformation from an N dimension random vector $X \in R^N$ with zero mean and covariance matrix C_X to a lower K_1 dimension random vector $Y \in R^{K_1}$, $K_1 < N$. $\boldsymbol{Y}_{K_1 \times L} = \boldsymbol{A}^T_{N \times K_1} \boldsymbol{X}_{N \times L}$, where $\boldsymbol{A}$ is an $N \times K_1$ transformation matrix whose columns are K_1 orthogonal eigenvectors corresponding to the first K_1 largest eigenvalues of the covariance matrix C_X. As discussed above, K_1 is defined by how much information should be kept from the original data after the transformation. Let N vectors V_1, V_2, ..., $V_N \in R^{K_1}$ be the rows of the transformation matrix $\boldsymbol{A}$. Each vector V_n, so called weighted vector, represents the projection of the n^{th} feature of the vector X to the lower dimensional space R^{K_1}. Two independent features of X have maximally separated weighted vectors, while two correlated features have identical weighted vectors. To divide N features of the data X into K separated clusters containing correlated features, we could apply K-means algorithm [22] using the structure of the rows V_n and classify these vectors into K separated groups G_k. RUL estimation algorithm based on feature clustering in a hierarchical approach, can be described as follows.

Step 1: Compute the approximated covariance matrix C_X of the N-dimensional vector X.

Step 2: Compute K_1 eigenvectors of C_X to form matrix $\boldsymbol{A}$.

Step 3: Form a set of N row vectors V_1, V_2, ..., V_N from the matrix $\boldsymbol{A}$. Remove all vectors V_n having the norm $||V_n|| \leq \varepsilon$, (ε is very small). Update the new vector set $\{V_1, V_2, ..., V_{N'}\} \in R^{K_1}$. Divide the new vector set into K vector clusters G_k using K-means algorithm.

Step 4: For each sensor cluster G_k, apply the PCA-based centralized RUL estimation to find the local RUL for the test data. Assuming that all these sensor clusters are independent to each other, K local RUL_{i_k} estimates of K sensor clusters can be linearly combined, i.e. averaged, to provide the final RUL_i of test data i.

3.4. Decentralized processing

In this approach, each individual sensor measurement is used to provide the local RUL estimation of the test data. These answers are then fused to give final RUL prediction. Given a set of distributed sensors, many of them can be useless. The usefulness of one sensor is presented by its contribution to the whole system information. There are two main types of useless sensors. The first one includes those sensors whose data are constant during the life time of the system or contain only noises. The second one is those sensors whose measurements are highly correlated with the others. Useless sensor data must be removed before any further diagnostics. Mutual information [23] can be used to distinguish these useless sensors.

3.4.1. Sensor selection based on mutual information

The mutual information between two random variables measures the dependent information between them. The higher the mutual information, the higher the correlation between two variables, and vice versa. Mutual information between two random variables X and Y can be calculated as [23] $I(X;Y) = H(Y) - H(Y|X) = H(X) - H(X|Y)$, where $H(Y)$ is the entropy of random variable Y. $H(Y) = -\int p_Y(y) \log\big(p_Y(y)\big)\, dy$, where $p_Y(y)$ is the probability density function of Y, which can be estimated using histograms, kernels, B- splines, or Nearest Neighbors [24]. The conditional entropy $H(Y|X)$ denotes the entropy of variable Y when the values of X are given.

$$H(Y|X) = -\int p_X(x) \int p_{Y|X}(y|x) \log\Big(p_{Y|X}(y|x)\Big)\, dxdy$$

The joint entropy between two random variables X and Y is defined in the chain rule [23] as $H(X,Y) = H(X) + H(Y|X)$. Then, the mutual information can be calculated as $I(X;Y) = H(X) + H(Y) - H(X,Y)$. We can see that $I(X;X) = H(X)$. This means mutual information of a random variable with itself is its entropy, or it can be referred as self-information [23]. The entropy of one variable presents the information it carries about the system. This means the feature having too small self-information is useless.

The conditional mutual information of random variables X and Y given random variable Z is defined as $I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$. There are some other chain rules for entropy and mutual information as $H(X_1, X_2, \dots, X_n) = \sum_{i=1}^{n} H(X_i|X_{i-1}, X_{i-2}, \dots, X_1)$ and $I(X_1, X_2, \dots, X_n; Y) = \sum_{i=1}^{n} I(X_i; Y|X_{i-1}, X_{i-2}, \dots, X_1)$

The feature having "high" mutual information with the remaining feature set is considered to be dependent on the other features and should be removed. Based on these consideration, a sequential backward sensor/feature selection algorithm can be described in five steps as follows.

Step 1: Start with a full set of N given sensors/features $F = \{f_1, f_2, \dots, f_N\}$.

Step 2: Calculate the self-information, $H(f_n)$, of all features. Remove the features f_n having $H(f_n)$ smaller than a predefined small value ε. F then has $\tilde{N}$ features.

Step 3: Calculate the mutual information between each feature f_n and the set of remaining features, $I_n = I\left(f_n; \{f_{m|m\neq n}\}\right)$.

Step 4: Calculate the mean of all I_n, $M(I) = \frac{1}{\tilde{N}}\sum_{n=1}^{\tilde{N}} I_n$. Calculate the gap between the biggest I_n and the smallest I_n, $G(I) = \max I_n - \min I_n$. If $M(I) < G(I)$, remove the feature having the highest I_n. Update the feature set, return to Step 3.

Step 5: The algorithm stops when $M(I) \geq G(I)$, meaning that all the remaining features in the set are closely dependent if $M(I)$ is large (or independent if $M(I)$ is small) on each other. The final feature set has K features.

3.4.2. RUL estimation

An RUL estimation algorithm based on similarity searching is applied on each of K features/sensors. Depending on how K features depend on each other, some fusion techniques can be applied to provide the final RUL estimation. Assuming that these features are independent, the simplest way to find the final RUL is to take the average value of all K RUL estimates.

4. EXPERIMENTS

4.1. Dataset

In this research, the dataset from the Challenge Problem of the International Conference on Prognostics and Health Management 2008 [25] is used to evaluate the proposed RUL estimation algorithms. The dataset consists of multiple multivariate time series, which are divided into training and testing subsets. There are 218 series provided in the training set, also 218 series in the testing set. All of these series are different in length. Each time series is from a different running instance of the same complex engineered system (referred to as a "unit"), e.g., the data might be from a fleet of ships of the same type. Each unit starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e., it is not considered a fault condition. There are three operational settings that have a substantial effect on unit performance. These settings are also included in the data. The data is contaminated with sensor noise. The unit is operating normally at the start of each time series, and develops a fault at some point during the series. In the training set, the fault grows in magnitude until system failure. In the test set, the time series ends some time prior to system failure. The objective is to predict the number of remaining operational cycles before failure in the test set, i.e., the number of operational cycles after the last cycle that the unit continues to operate. The data are provided with 26 columns of numbers, separated by spaces. Each row is a snapshot of data taken during a single operational cycle, each column is a different variable. The original explanation to the Problem and the dataset can be referred and downloaded in [25].

4.2. Evaluation criteria

The score used for evaluating the prediction algorithms is defined as the exponential penalty to the prediction error, and the score of whole algorithm is the sum of all scores B_i from all RUL estimations of N test runs:

$$d_i = \widehat{RUL_i} - RUL_i$$

$$B_i = \begin{cases} e^{-d_i/13} - 1, d_i \le 0 \\ e^{d_i/10} - 1, d_i > 0 \end{cases}, i = 1, \dots, N$$

$$B = \sum_{i=1}^{N} B_i$$

The key aspect of a prognosis algorithm is to avoid failures. It is preferable to have an early prediction than a late one. The scoring formula is asymmetric around the true time of failure such that late predictions are more heavily penalized than early ones. In both cases, the penalty grows exponentially with increasing error. A perfect prediction scores zero.

4.3. Data preprocessing

The dataset is first going through a mean extraction process, then is denoised. There have been many signal densoising approaches in the literature. Lerga *et al.* [26] introduce a denoising method based on a modification of the intersection of confident interval rule, which is complemented by the relative intersection of confidence intervals length. Matz *et al.* [27] use dual filtering algorithms for denoising ultrasonic A-scan signals. Their methods are based on applications of efficient denoising algorithms such as Wiener filter and discrete wavelet transform. Signal denoising based on wavelet transform [2] has attracted a lot of attention in recent. Wavelet transform is preferable to other transforms like Fourier transform or cosine transform because of its ability to locate the signal simultaneously in time and frequency domains. Input signal is first decomposed using the orthogonal wavelet basis. The wavelet coefficients are then suppressed using hard or soft thresholding rule [2]. Finally, the signal is transformed back to the original domain to provide a “noiseless” version of the data.

Hard threshold rule: $T(y) = \begin{cases} y \ \ if \ |y| \ge \lambda \\ 0 \ \ if \ |y| < \lambda \end{cases}$

Soft threshold rule: $T(y) = \begin{cases} y - sgn(y)\lambda \ \ if \ |y| \ge \lambda \\ 0 \qquad if \ |y| < \lambda \end{cases}$

where, y is the input data to the thresholding process, $T(\cdot)$ is the output data, λ is the threshold level. In this research, we implement wavelet denoise with soft thresholding using *ddencmp* and *wdencmp* functions in the Wavelet Toolbox of MATLAB to estimate fitting curves of noisy signals.

4.4. Procedures and results

4.4.1. Centralized processing with PCA

218 training series are used as input to PCA algorithm. Each training series has its own PCs corresponding to the eigenvectors and eigenvalues of its approximated covariance matrix. Table 4.1 presents an example of eigenvalues in descending order of the approximated covariance matrix of training series 1. The eigenvalues are not the same in different training series due to the different variation of the data resulting from different degradations from one run to the others. However, there is only one dominant eigenvalue from the list of 21 eigenvalues in all training series. This means there is only one PC is retained. As a result, there is only one feature needs to be extracted from the original data.

Table 4.1. Eigenvalues sorted in descending order of training unit 1.

1st eigenvalue	66.713
2nd eigenvalue	0.51103
3rd eigenvalue	0.35184
4th eigenvalue	0.092441
…	…
21st eigenvalue	0

The answer for the RUL of each test unit is calculated from 218 answers of all training units. Since there is only one PC selected, we do not need to combine the results from many different PCs. The result of PCA method is provided in Table 4.2

4.4.2. Hierarchical processing with Feature classification

Since there is only one eigenvector selected for data transformation, the matrix A has only one column, and the vectors V_1, V_2, ..., V_N are scalars. By dropping out all sensors n whose $V_n \leq \varepsilon,\ \varepsilon = 10^{-3}$, seven sensors are eliminated. The final sensor set is {2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, 21}. K-means algorithm, K=7, is applied to find out correlated sensor groups. Seven sensor groups are {2, 11, 15}, {3, 4}, {7, 12}, {8, 13}, {9, 14}, {17}, {20, 21}. PCA algorithm is then applied to all seven sensor groups separatedly to find seven RUL estimates for each test series. The final RUL prediction of each test series is the average of all seven answers. The resulting score of this approach is provided in Table 4.2.

For this specific problem, the hierarchical approach can help improve the result significantly from the centralized appoach. The reason can be explained as follows. The investigated system is a dynamic system. The fault causing the system to fail down is not the same in every run. As a result, the system degradation process is different in all training series. High complex faults happening during one lifetime of the system may cause more complicated changes in sensor measurements than simpler ones. By using only one PC in all training series, we cannot fully investigate the complication in patterns of the data. This leads to the fact that the centralized RUL estimation algorithm cannot produce the exact results for all test units. In the hierarchical processing with feature classification, the feature set is divided into smaller groups. It helps provide many multi-dimensional data sets. As a result, detailed patterns of the original data can be further investigated, and the combination of these results after clustering can provide a more accurate prediction.

The advantages of centralized processing over hierarchical processing are the smaller computational time and the less complexity of the algorithm. However, for this specific challenge problem, hierarchical processing performs better than centralized processing. In many other real problems, where the training data need more than one PC to depict the system health state, e.g, the number of PCs needed to investigate in PCA algorithm might be equal to the number of selected feature groups, the estimation results provided by the two processing methods are believed to be the same. In this case, centralized processing is preferable due to its simplicity.

Table 4.2. Comparison among different methods.

Methods	Score
Centralized processing with PCA	53428
Hierarchical processing with feature classification	12757
Decentralized processing with feature selection	16727

4.4.3. Decentralized processing with Feature selection

In the given problem, each sensor measurement is treated as one feature. The feature selection algorithm is applied to find the optimum feature subset F. In this research, mutual information calculation packet written in Matlab codes by Peng *et al.* [28] is used. There are totally five features to be selected, $F = \{2, 7, 8, 11, 17\}$. These features are denoised using wavelets with soft threshold. The RUL estimation algorithm using similarity search is used to find the RUL_k from each selected sensor S_k. The final RUL is the average of all the results from five selected features. The score of feature selection method is provided in Table 4.2.

The score of decentralized method is a bit higher than the hierarchical method. This is due to the elimination of many unuseful features. Indeed, the number of the final selected features used to estimate the RUL can effect the final score of the algorithm. A larger number of features tends to improve the precision of the prediction results. However, enlarging the final feature subset leads to increasing the computational time of the whole algorithm. Moreover, dependent features added to the final set do not produce much improvement.

5. CONCLUSIONS

All methods for RUL prediction proposed in the research have been evaluated using the data provided by PHM2008 competition. The actual remaining life time of the test runs are provided as a reference to calculate scores for performance evaluation of these methods. However, it is sensitive to use the scores to evaluate how a predicting method works because in any situation one approach can produce some very good predictions beside some bad results. To be specific, small number of large errors can dominate in the final score. In this Challenge Problem, some testing series have very short history, which potentially cause large prediction errors. All of RUL prediction methods introduced in this Research do not use any RUL adjustment algorithm, and prediction results could be improved significantly if some distribution of the predicted RUL is exploited. For example, using the distribution of the actuall RUL can help cut off some too large values of predicted RUL to a normalized smaller one in order to reduce the risk. Moreover, late RUL estimates are judged more heavily than early ones.

All three mentioned RUL estimation methods depend heavily on similarity search algorithm. The outputs of similarity search can be used intuitively. Specifically, the similarity searching algorithm helps find out the most matched data from the library for the current query based on a distance function. These distances depend heavily on the length of the portions used to compare between two sequences. Besides, these proposed methods do not fully exploit the domination of training series which contain the most matched portion with the test data among all training data. This is the key for tuning techniques that can be used to improve the prediction results, such as fixing the maximum length of the comparison portions, choosing only fixed number of training series most similar to the test data to estimate the RUL, etc. Another matter that should be discussed is the size of the library of past runs. The larger the library we have, the more samples of past runs can be used to estimate the RUL of the current test data, as a result, the more accurate the prediction could be. The author believes that these prediction methods with sufficient modifications can be very useful in solving many real problems.

REFERENCES

1. Shepperd M., Kadoda G. - Comparing software prediction techniques using simulation, IEEE transactions on software engineering **27** (11) (2001) 1014–1022.
2. Durand S., Froment J. - Artifact free signal denoising with wavelets, International Conference on Acoustics, Speech and Signal Processing, Salt Lake City, UT, 2001, pp. 3685-3688.
3. Xue F., Goebel K., Bonissone P., Yan W. - An instance-based method for remaining useful life estimation for aircraft engines, Journal of Failure Analysis and Prevention **8** (2) (2008) 199–206.
4. Yan J., Ko M., Lee J. - A prognostic algorithm for machine performance assessment and its application, Journal of Production Planning and Control **15** (8) (2004) 796–801.
5. Shao Y., Nezu K. - Prognosis of remaining bearing life using neural networks, Proceedings of the Institute of Mechanical Engineer, Part I, Journal of Systems and Control Engineering **214** (3) (2000) 217–230.
6. Liao H., Zhao W., Guo H. - Predicting remaining useful life of an individual unit using proportional hazards model and logistic regression model, Annual Reliability and Maintainability Symposium, Newport Beach, CA, 2006, pp. 127–132.
7. Kiddy J. - Remaining useful life prediction based on known usage data, Proceedings of SPIE 5046, Nondestructive Evaluation and Health Monitoring of Aerospace Materials and Composites II, **11** (2003) 11–18.
8. Saha B., Goebel K., Poll S., Christophersen J. - A bayesian framework for remaining useful life estimation, AAAI Fall Symposium on Artificial Intelligence for Prognostics (working notes), Arlington, VA, 2007, pp. 1-6.

9. Pattipati B., Pattipati K., Christopherson J., Namburu S., Prokhorov D., and Qiao L. - Automotive battery management systems, IEEE AUTOTESTCON, Salt Lake City, UT, 2008, pp. 581–586.
10. Cheng S., Pecht M. - Multivariate state estimation technique for remaining useful life prediction of electronic products, AAAI Fall Symposium on Artificial Intelligence for Prognostics, Arlington, VA, 2007, pp. 26–32.
11. Byington C., Watson M., Edwards D. - Data-driven neural network methodology to remaining life predictions for aircraft actuator components, Proceedings of the IEEE Aerospace Conference, New York, NY, 2004, pp. 3581–3589.
12. Kavanagh D., Scanlon P., Boland F. - Envelope analysis and data-driven approaches to acoustic feature extraction for predicting the remaining useful life of rotating machinery, IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, NV, 2008, pp. 1621–1624.
13. Lao H., Zein-Sabatto S. - Analysis of vibration signal’s time-frequency patterns for prediction of bearing’s remaining useful life, Proceedings of the 33rd Southeastern Symposium on System Theory, Athens, OH, 2001, pp. 25–29.
14. Liu F., Du P., Weng F., Qu J. - Use clustering to improve neural network in financial time series prediction, Third International Conference on Natural Computation (ICNC 2007), Haikou, 2007, pp. 89-93.
15. Goldin D., Kanellakis P. - On similarity queries for time-series data: constraint specification and implementation, International Conference on Principles and Practice of Constraint Programming (CP 1995), 1995, pp. 137-153.
16. Rafiei D. - On similarity-based queries for time series data, Proceedings of the 15th IEEE Int'l. Conference on Data Engineering, Sydney, Australia, 1999, pp. 410–417.
17. Chan K., Fu W. - Efficient time series matching by wavelets, In Proceedings of the 15th IEEE Int'l. Conference on Data Engineering, Sydney, Australia, 1999, pp. 126-133.
18. Lee S., Chun S., Kim D., Lee J., Chung C. - Similarity search for multidimensional data sequences, Proceedings of 16th International Conference on Data Engineering, San Diego, CA, 2000, pp. 599-608.
19. Abfalg J., Kriegel H., Kroger P., Kunath P., Pryakhin A., Renz M. - Interval-focused similarity search in time series databases, Proc. 12th Int. Conf. on Database Systems for Advanced Applications, Bangkok, Thailand, 2007, pp. 1-12.
20. Liu J., Djurdjanovic D., Ni J., Casoetto N., Lee J. - Similarity based method for manufacturing process performance prediction and diagnosis, Computers in Industry **58** (6) (2007) 558–566.
21. Jolliffe I. - Principal Component Analysis, Springer-Verlag New York, Inc., 2002.
22. Kanungo T., Mount D., Netanyahu N., Piatko C., Silverman R., Wu A. - An efficient k-means clustering algorithm: Analysis and implementation, IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (7) (2002) 881–892.
23. Cover T., Thomas J. - Elements of Information Theory, John Wiley and Sons, Inc., 1991.
24. Krier C., Francois D., Wertz V., Verleysen M. - Feature scoring by mutual information for classification of mass spectra, Procs. of the 7th International FLINS Conference - Applied Artificial Intelligence, Genova, Italy, 2006, pp. 557–564.

25. PHM-2008 Prognostics Data Challenge Dataset.
    http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/ - Access date: 4/4/2017.
26. Lerga J., Vrankic M., Sucic V. - A signal denoising method based on the improved ici rule, IEEE Signal Processing Letter **15** (2008) 601–604.
27. Matz V., Kreidl M., Smid R., Starman S. - Ultrasonic signal de-noising using dual filtering algorithm, 17th World Conference on Nondestructive Testing, Shanghai, China, 2008, pp. 25-28.
28. Peng H., Long F., Ding C. - Feature selection based on mutual information: Criteria of max-dependency, max-relevance, min-redundancy, IEEE transactions on pattern analysis and machine intelligence **27** (8) (2005) 226–1238.