# Comparison and Fusion of Various Classification Methods Applied to Aero-engine Fault Diagnosis

Guoxing Lan[1, 2], Nong Cheng[1], Qing Li[1]

1. Department of Automation, Tsinghua University, Beijing, 100084, China

2. AECC China Aero-Engine Control System Institute, Wuxi, 214063, China

E-mail: langx15@mails.tsinghua.edu.cn

**Abstract:** Aero-engine fault diagnosis plays a crucial role in safe operation and cost-effective maintenance. Early detection and isolation of component faults prior to failure of aero-engines is of utmost importance. This paper applied various classification methods, including Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbors (K-NN) and Linear Discriminant Analysis (LDA), to aero-engine component fault detection and isolation. These 4 various methods were tested under different circumstances involving large training samples, small training samples, few features, and noisy data. Comparison of accuracy and time efficiency was made among those methods, and a fusion algorithm of these 4 classification methods was proposed based on their training classification accuracy. The experiment results show that SVM has excellent accuracy performance but poor time efficiency while LDA performs well in both accuracy and time efficiency, and the fusion algorithm has higher accuracy than any single method.

**Key Words:** Aero-engine, Fault Diagnosis, Classification Methods, Comparison and Fusion Algorithm

## 1 INTRODUCTION

Aero-engine fault diagnosis is to detect, isolate and identify faults/malfunctions and failures. Typical aero-engine faults mainly include stability faults, gas path faults, vibration faults, wear faults, flame-out faults, bearing faults, structural fatigue, and control system faults, etc.[1]. Aero-engine gas path faults refer to degradations of gas path components such as fan, compressor, combustor and turbines which can result in the deterioration of performance and reliability of aero-engines over the service life. Common causes of such degradation of gas path components mainly include fan blade damage, compressor fouling, increase of the blade-tip clearance in the turbine, wear and corrosion, and so on [2].

Aero-engine fault diagnosis methods are primarily built on model-based and sensor-data-based analysis. Due to the nonlinearity and complexity of aero-engine model, and the development of data-driven diagnosis and prognostics [3][4] [5], machine learning algorithms have been applied to aero-engine fault diagnosis. Support Vector Machine (SVM) is a common machine learning algorithm and has been a popular classification method for aero-engine fault detection and isolation [6] [7] [8] [9]. However, research on other classification methods used in aero-engine fault diagnosis is still rare.

In this paper, we applied various classification methods, including Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbors (K-NN) and Linear Discriminant Analysis (LDA), to aero-engine component fault detection and isolation, and proposed a fusion algorithm. The classification accuracy and time cost of those methods under different circumstances involving large training samples, small training samples, few features, and noisy data was made, and the results show that SVM has excellent accuracy performance but costs too much time, while LDA performs well in both accuracy and time efficiency. The fusion algorithm has higher classification accuracy than any single method.

## 2 PROBLEM DESCRIPTION

### 2.1 Aero-engine Model

The aero-engine model we used here is a twin shaft turbofan engine consisting of five major components: fan, high-pressure compressor (HPC), combustor, high-pressure turbine (HPT) and low-pressure turbine (LPT). Its structure diagram is shown in figure 1.

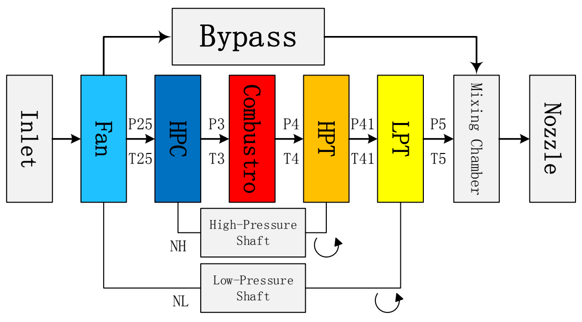

Fig 1. Aero-engine Model Diagram.

This work is supported by the Aviation Science Foundation of China (No. 20100758002).

A Simulink-based test bed was built to simulate the dynamic nonlinear turbofan engine model in component level, given the characteristic curves and aero-thermodynamics equations of each engine component. Systems of nonlinear equations of different components were solved using numerical algorithms to find the correct parameters at given working point.

### 2.2 Fault Types

Aero-engine is a complex system and a number of possible faults could potentially occur in different components. In this paper, we limited our study to gas path faults related to fan, HPC, HPT and LPT. More specifically, we focused on detection and isolation of the following 4 types of component faults [10]:

(1) Fan fault: Fan blade damage caused by foreign object impact (e.g. bird strikes).

(2) HPC fault: High-pressure compressor fouling and blade.

(3) HPT fault: Increase of the blade-tip clearance in the high-pressure turbine, high-pressure wear and corrosion.

(4) LPT fault: Increase of the blade-tip clearance in the low-pressure turbine, low-pressure wear and corrosion.

We made the following assumptions:

(1) Every type of fault is independent of each other.

(2) Only one fault case happens at a time.

(3) Faults occur instantaneously and remain at the same level in a short period of time.

### 2.3 Data Generation

The twin shaft turbofan engine model described in Section 2.1 is implemented in a component-level Simulink model, hence it is convenient to inject different faults described in Section 2.2. Each fault type in Section 2.2 will cause performance deterioration in the related component, which reflected in the reduction of flow and efficiency. The performance deterioration settings are shown in Table 1.

Table 1. Performance Deterioration Simulation

| Fault Type | Component | Performance Parameter | Degradation Factor |
|---|---|---|---|
| Fault 0 | / | Flow | 0 |
| | | Efficiency | 0 |
| Fault 1 | Fan | Flow | 0.05 |
| | | Efficiency | 0.05 |
| Fault 2 | HPC | Flow | 0.05 |
| | | Efficiency | 0.05 |
| Fault 3 | HPT | Flow | 0.02 |
| | | Efficiency | 0.02 |
| Fault 4 | LPT | Flow | 0.02 |
| | | Efficiency | 0.02 |

In Table 1, Fault 0 means the case that the aero-engine works without performance deterioration, and Fault 1 to Fault 4 means the cases that the aero-engine works with performance deterioration in different components respectively.

The 13 sensor measurements of the aero-engine performance parameters are listed in Table 2. These are the standard engine performance measurements used for aero-engine fault diagnosis.

Table 2. Sensor Measurements

| Number | Parameter Meaning | Symbol |
|---|---|---|
| 1 | High-Pressure Shaft Speed | NH |
| 2 | Low-Pressure Shaft Speed | NL |
| 3 | Fan Exit Pressure | P25 |
| 4 | Fan Exit Temperature | T25 |
| 5 | HPC Exit Pressure | P3 |
| 6 | HPC Exit Temperature | T3 |
| 7 | Combustor Exit Pressure | P4 |
| 8 | Combustor Exit Temperature | T4 |
| 9 | HPT Exit Pressure | P41 |
| 10 | HPT Exit Temperature | T41 |
| 11 | LPT Exit Pressure | P5 |
| 12 | LPT Exit Temperature | T5 |
| 13 | Fuel Flow Rate | WF0 |

In different cases of component faults shown in Table 1, we changed the input fuel flow rate WF0 and got 13 columns of output data, respectively corresponding to sensor measurements shown in Table 2.

## 3 CLASSIFICATION METHODS AND FUSION ALGORITHM DESIGN

In this chapter, we would introduce the 4 classification methods (SVM, DT, K-NN and LDA) applied in our study, including their basic theories and specific designs in the experiments. A fusion algorithm of these 4 classification methods was proposed and explained in the last section.

### 3.1 Support Vector Machine

SVM is a typical kernel method in machine learning originated from the statistical learning theory [11]. The mechanism of SVM is to find the optimal separating hyper-plane, defined as the one with the maximal margin of separation between positive and negative samples.

Taking two-class classification problem as an example, given a training set $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}, y_i\in\{-1,+1\}$ and hyper-plane denoted as $w^T x+b=0$, it is required to satisfy the following constraints to ensure all training samples correctly classified.

$$y_i\left(w^T x_i+b\right)\geq 1, i=1,2,\cdots,m \tag{1}$$

The core idea of SVM is to maximize the margin of two support vectors of two different categories:

$$\begin{gathered}\max_{w,b}\frac{2}{\|w\|}\\ s.t.\ y_i\left(w^T x_i+b\right)\geq 1, i=1,2,\cdots,m\end{gathered} \tag{2}$$

In order to solve problem (2) more efficiently, we can convert it into a minimization problem and derive its Lagrange function, then problem (2) is converted into problem (3) [12]:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i \left(w^T x_i + b\right)\right) \\ &s.t. \sum_{i=1}^{m} \alpha_i y_i = 0 \\ &\quad \alpha_i \geq 0, i = 1, 2, \cdots, m \end{aligned} \tag{3}$$

After solving $\alpha_i$ in formula (3), we can obtain the optimal classification function:

$$f(x) = \operatorname{sgn}\left(w^T x + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i x_i^T x + b\right) \tag{4}$$

For the case of linear impartiality, the main idea of kernel method is to map the sample from the original vector space to a higher-dimensional vector space:

$$x \rightarrow \phi(x) \tag{5}$$

Therefore, (4) can be rewritten as:

$$\begin{aligned} f(x) &= \operatorname{sgn}\left(w^T \phi(x) + b\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i \phi(x_i)^T \phi(x) + b\right) \end{aligned} \tag{6}$$

We chose radial basis function (RBF) to be the kernel function $\phi(x)$, of which the expression is

$$\kappa(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right) \tag{7}$$

For the purpose of avoiding over-fitting problem, we tolerate the case of few data points satisfying $y_i\left(w^T x_i + b\right) \geq 1$. Thus the optimization problem in formula (3) can be rewritten as [12]:

$$\begin{aligned} &\min_{w,b,\xi_i} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \\ &s.t.\ \ y_i\left(w^T x_i + b\right) \geq 1 - \xi_i, i = 1, \cdots, m \\ &\quad \xi_i \geq 0, i = 1, \cdots, m \end{aligned} \tag{8}$$

Parameter $\sigma$ and parameter $C$ will affect the performance of SVM. Cross-validation method along with grid search method is used to find optimal $\sigma$ and $C$.

We implemented SVM algorithm with libsvm [13] at MATLAB environment. The SVM model can be obtained with function svmtrain( ) and training data, then the predicted labels and accuracy of testing data can be obtained with function svmpredict ( ) and the model we got earlier. The training data and testing data were normalized between the range of -1 and 1 at the beginning. The Pseudocode of the SVM algorithm can be described as follows:

```
**Input**: training set $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$
        testing set $\tilde{D} = \{(\tilde{x}_1, \tilde{y}_1), (\tilde{x}_2, \tilde{y}_2), \cdots, (\tilde{x}_n, \tilde{y}_n)\}$
**Procedure**: $SVM(\tilde{D}, D)$
$accuracy\_max = 0$; $[C', \sigma'] = [0, 0]$
Divide $D$ into 5 independent subsets $D_1, D_2, D_3, D_4, D_5$ in the same size randomly.
for $\log(C)$=-10 to 15
  for $\log(\sigma)$=-15 to 10
    for i=1 to 5
      $model\_i = svmtrain(D - D_i, [C, \sigma])$
      $accuracy\_i = svmpredict(D_i, model\_i)$
    end for
    $accuracy\_mean = \frac{1}{5}\sum_{i=1}^{5} accuracy\_i$
    if $accuracy\_max >$ $accuracy\_mean$
      $accuracy\_max = accuracy\_mean$
      $[C', \sigma'] = [C, \sigma]$;
    end if
  end for
end for
$model$ = $svmtrain(D, [C', \sigma'])$
$[SVM\_label, SVM\_accuracy] = svmpredict(\tilde{D}, model)$
**Output**: $[SVM\_label, SVM\_accuracy]$
```

### 3.2 Decision Tree

Decision Tree is a classifier that builds the knowledge-based system by the inductive inference from histories in tree form [14]. Generally, a Decision Tree contains a root node, a number of interior nodes and leaf nodes, where each leaf node is a decision, i.e., represents a classification. Each interior node is a test on an attribute. The classification process of a Decision Tree is to classify an instance by starting at the root node and following a top-down path dictated by attribute tests until a leaf node is encountered.

The key design of Decision Tree is the criterion for the partition of attribute test. Generally, the objective of the partition process is to make the "purity" of each branch node higher and higher. We chose CART algorithm to find the best partition attributes. CART algorithm uses Gini Index to choose the best partition attribute [15]:

$$Gini\_index(D, a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v) \tag{9}$$

Where $Gini(D)$ is Gini Value used to measure the purity:

$$Gini(D) = \sum_{k=1}^{U} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{U} {p_k}^2 \tag{10}$$

In our study, we implemented Decision Tree algorithm with function classregtree ( ) and treeval ( ) at Matlab environment. The Pseudocode of the DT algorithm can be described as follows:

```
**Input**: training set $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$;
        testing set $\tilde{D} = \{(\tilde{x}_1, \tilde{y}_1), (\tilde{x}_2, \tilde{y}_2), \cdots, (\tilde{x}_n, \tilde{y}_n)\}$;
```

```
Procedure: $DT(\tilde{D},D)$
  $actual\_label=\{y_1,y_2,\cdots y_n\}$
  $T=classregtree(D)$   %CART algorithm
  $DT\_label=treeval(\tilde{D})$
  $DT\_accuracy=num(DT\_label==actual\_label)/n$
Output: $[DT\_label,DT\_accuracy]$
```

### 3.3 K-Nearest Neighbors

K-Nearest Neighbors is an intuitive classification method without learning process: given a training dataset $D=\{(x_1,y_1),(x_2,y_2),\cdots(x_m,y_m)\}$ ( $x_i$ is feature vector while $y_i$ is label, $i=1,2,\cdots,m$ ), for input instance $\tilde{x}$ , K-NN finds the nearest k instances to $\tilde{x}$ and assign the label of the largest number of instances [16].

The distance measure in K-NN is usually Euclidean distance, or more generally $L_p$ distance. The decision rule of classification can be described as:

$$\begin{gathered} y=\arg\max_{c_j}\sum_{x_i\in m_k(\tilde{x})} I\left(y_i=c_j\right) \\ i=1,2,\cdots,\mathrm{m}; j=1,2,\cdots,k \end{gathered} \tag{11}$$

Where $m_k(\tilde{x})$ is neighborhood of $\tilde{x}$ containing k samples. $I(y_i=c_j)$ is indicator function:

$$I\left(y_i=c_j\right)=\begin{cases}1, \text{ } if\ y_i=c_j \\ 0, \text{ } if\ y_i\neq c_j\end{cases} \tag{12}$$

The selection of k is crucial to the classification result. In this paper, we used cross-validation to find the optimal k. K-NN algorithm was implemented with function knnclassify( ) at Matlab environment. The Pseudocode of the K-NN algorithm can be described as follows:

```
Input: training set $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$
       testing set $\tilde{D}=\{(\tilde{x}_1,\tilde{y}_1),(\tilde{x}_2,\tilde{y}_2),\cdots,(\tilde{x}_n,\tilde{y}_n)\}$
Procedure: $KNN(\tilde{D},D)$
  $k'=1$; $actual\_label=\{y_1,y_2,\cdots y_n\}$
  Divide $D$ into 5 independent subsets $D_1,D_2,D_3,D_4,D_5$ in the same size randomly.
  for k=1 to 10
    for i=1 to 5
      $label\_i=knnclassify(D_i,D-D_i,k)$
      $accuracy\_i=num(label\_i==actual\_label)/n$
    end for
    $accuracy\_mean=\frac{1}{5}\sum_{i=1}^{5}accuracy\_i$
    if $accuracy\_max>$ $accuracy\_mean$
      $accuracy\_max=accuracy\_mean$
      $k'=k$ ;
    end if
  end for
  $KNN\_label=knnclassify(\tilde{D},D,k')$
  $KNN\_accuracy=num(KNN\_label==actual\_label)/n$
Output: $[KNN\_label,KNN\_accuracy]$
```

### 3.4 Linear Discriminant Analysis

The core idea of LDA is projecting data from d dimensions onto a line and determining a threshold which divides the data into two classes [17]. The optimal projection direction should ensure the two classes having the largest possible between-class scatter and the smallest within-class scatter after the projection. Definitions of between-class scatter matrix and within-class scatter matrix are shown in formula (13) and (14):

$$S_b=(m_1-m_2)(m_1-m_2)^T, i=1,2 \tag{13}$$

$$S_i=\sum_{x_j\in\chi_i}(x_j-m_i)(x_j-m_i)^T, i=1,2 \tag{14}$$

Where $i=1,2$ is class label; $x_j$ is feature vector of a sample; $\chi_i$ is dataset of class $i$ in the original space; $m_i$ is the mean vector of samples in class $i$ .

The between-class scatter matrix and within-class scatter matrix of data in the one-dimensional space after projection can be described as formula (15) and (16) respectively:

$$\tilde{S}_b^{\,2}=(\tilde{m}_1-\tilde{m}_2)^2 \tag{15}$$

$$\tilde{S}_i^{\,2}=\sum_{\tilde{x}_j\in\tilde{\chi}_i}(\tilde{x}_j-\tilde{m}_i)^2, i=1,2 \tag{16}$$

Where $\tilde{\chi}_i$ is dataset of class $i$ in the new space after the projection; $\tilde{x}_j$ is the new vector after the projection; $\tilde{m}_i$ is the mean value of samples in class $i$ in the new space after projection.

The goal that the two classes have the largest possible between-class scatter and the smallest within-class scatter after the projection can be described as the following optimization problem:

$$\max\ \frac{\tilde{S}_b^{\,2}}{\tilde{S}_1^{\,2}+\tilde{S}_2^{\,2}} \tag{17}$$

Denote $w$ as The projection direction vector, problem (17) can be converted into the following optimization problem [18]:

$$\begin{gathered}\min -\frac{1}{2}w^TS_bw \\ s.t.\quad w^TS_ww=1\end{gathered} \tag{18}$$

Where $S_w=S_1+S_2$ is called pooled within-class scatter matrix.

After solving problem (20) using Lagrange function method, we can derive the optimal projection direction:

$$w^*=S_w^{-1}(m_1-m_2) \tag{19}$$

The decision rule can be described as:

$$\begin{cases}x\in\omega_1,\ if\ \ g(x)=w^Tx+t_0>0 \\ x\in\omega_2,\ if\ \ g(x)=w^Tx+t_0<0\end{cases} \tag{20}$$

Where $\omega_1$ means class 1 and $\omega_2$ means class 2; $t_0$ is classification threshold, usually set as $t_0 = -\frac{1}{2}(\tilde{m}_1 + \tilde{m}_2)$. Therefore, decision rule (20) can be rewritten as:

$$\begin{cases} x \in \omega_1, \textit{ if } \; g(x) = w^T\left(x - \frac{1}{2}(m_1 + m_2)\right) > \log\frac{P(\omega_2)}{P(\omega_1)} \\ x \in \omega_2, \textit{ if } \; g(x) = w^T\left(x - \frac{1}{2}(m_1 + m_2)\right) < \log\frac{P(\omega_2)}{P(\omega_1)} \end{cases} \tag{21}$$

Where $P(\omega_1)$ is the prior probability of class 1 and $P(\omega_2)$ is the prior probability of class 2. The Pseudocode of the LDA algorithm can be described as follows:

```
Input: training set $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$;
       testing set $\tilde{D} = \{(\tilde{x}_1, \tilde{y}_1), (\tilde{x}_2, \tilde{y}_2), \cdots, (\tilde{x}_n, \tilde{y}_n)\}$;
Procedure: $LDA(\tilde{D}, D)$
$actual\_label = \{y_1, y_2, \cdots y_n\}$
$LDA\_label = predict(Factor, \tilde{D})$
$LDA\_accuracy = num(LDA\_label == actual\_label) / n$
Output: $[LDA\_label, LDA\_accuracy]$
```

### 3.5 Fusion Algorithm

A fusion algorithm of the 4 above-mentioned classification methods was proposed in this section. The main idea of the fusion algorithm is to fuse the predicted labels of each single method by their cross-validation accuracies training set. The label with the largest weight is discriminated to be the final label. Since some classification methods might perform badly in some cases, the method with accuracy lower than a threshold will be abandoned in the fusion algorithm. The Pseudocode of the fusion algorithm can be described as follows:

```
Input: training set $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$
       testing set $\tilde{D} = \{(\tilde{x}_1, \tilde{y}_1), (\tilde{x}_2, \tilde{y}_2), \cdots, (\tilde{x}_n, \tilde{y}_n)\}$
Procedure: $fusion(\tilde{D}, D)$
$weight\_0 = 0, weight\_1 = 0, weight\_2 = 0$
$weight\_3 = 0, weight\_4 = 0$
for j=SVM, DT, K-NN, LDA
    $accuracy\_j = algorithm\_j(D)$
    if $accuracy\_j < accuracy\_threshold$
        $accuracy\_j = 0$
    end if
end for
for i=1 to n
    for j=SVM, DT, K-NN, LDA
        $label\_j = algorithm\_j(\tilde{x}_i)$
        switch ($label\_j$)
            case 0: $weight\_0 = weight\_0 + accuracy\_j$
                break;
            case 1: $weight\_1 = weight\_1 + accuracy\_j$
                break;
            case 2: $weight\_2 = weight\_2 + accuracy\_j$
                break;
            case 3: $weight\_3 = weight\_3 + accuracy\_j$
                break;
            case 4: $weight\_4 = weight\_4 + accuracy\_j$
                break;
            default: break;
    end for
    $fusion\_label\_i = \arg\max_k weight\_k$
end for
Output: $fusion\_label$
```

## 4 EXPERIMENT RESULTS AND COMPARISON

We tested and compared the 4 classification methods and the fusion algorithm under different conditions shown in Table 3:

Table 3: different experiment conditions

| | Training Samples Number | Testing Samples Number | Features Number | Signal to Noise Ratio(dB) |
|---|---|---|---|---|
| Condition 1 | 161075 | 10041 | 13 | / |
| Condition 2 | 10041 | 161075 | 13 | / |
| Condition 3 | 161075 | 10041 | 8 | / |
| Condition 4 | 10041 | 161075 | 8 | / |
| Condition 5 | 161075 | 10041 | 13 | 40 |
| Condition 6 | 10041 | 161075 | 13 | 40 |

Condition 1 is large training samples while condition 2 is small training samples; condition 3 is large training samples with few features while condition 4 is small training samples; condition 5 is large training samples with noise while condition 6 is small testing samples.

Under each condition, the training dataset and testing dataset of each method are the same. The experiment results of classification accuracy and time cost are shown in Table 4 and Table 5：

Table 4: accuracy of various classification methods under different conditions

| | SVM | DT | K-NN | LDA | Fusion |
|---|---|---|---|---|---|
| Condition 1 | 93.18 % | 85.65 % | 92.53 % | 92.37 % | 96.22 % |
| Condition 2 | 90.69 % | 44.94 % | 55.80 % | 97.32 % | 97.47 % |
| Condition 3 | 88.21 % | 85.65 % | 84.38 % | 87.67 % | 90.94 % |
| Condition 4 | 92.53 % | 44.94 % | 43.52 % | 93.53 % | 93.53 % |
| Condition 5 | 67.79 % | 45.40 % | 48.61 % | 63.00 % | 67.73 % |
| Condition 6 | 70.80 % | 30.90 % | 32.80 % | 69.90 % | 70.78 % |

Table 5: time cost of various classification methods under different conditions (sec)

|  | SVM | DT | K-NN | LDA | Fusion |
|---|---|---|---|---|---|
| Condition 1 | 24.49 | 3.70 | 23.84 | 0.85 | 52.89 |
| Condition 2 | 11.08 | 0.58 | 23.04 | 0.33 | 35.01 |
| Condition 3 | 50.05 | 3.50 | 0.78 | 0.46 | 54.80 |
| Condition 4 | 17.44 | 0.54 | 1.50 | 0.27 | 19.76 |
| Condition 5 | 569.14 | 34.07 | 24.54 | 47.62 | 675.38 |
| Condition 6 | 36.24 | 1.91 | 23.62 | 0.33 | 65.2 |

Comparing the results of various methods in Table 4 and Table 5, we can draw the following conclusions of each single classification method:

(1) Under large training samples condition, SVM, K-NN and LDA have high classification accuracy. DT's accuracy is not as high as the former 3 methods but not too bad.

(2) Under small training samples condition, SVM and LDA still perform well in accuracy while DT and LDA perform badly.

(3) The accuracy of all the 4 methods becomes lower when the size of training samples is small.

(4) When the data is with noise, all the 4 methods perform worse in accuracy. SVM and LDA perform better than the other 2 methods.

(5) SVM's time cost is always the most except for the case of small training samples when K-NN's time cost is the most

(6) LDA's time cost is always the least except for the case of large training samples with noise when DT's and K-NN's time cost is less than LDA.

(7) Larger training samples, fewer features and noise in data will increase SVM's time cost.

(8) Larger training samples and noise in data will increase DT's time cost.

(9) Larger features number will increase K-NN's time cost.

(10) Only larger training samples and noise in data will remarkably improve LDA's time cost.

The results of fusion algorithm compared to single method show that the fusion algorithm has the highest accuracy. However, since its algorithm procedure is based on all the 4 single methods, it cost the most time to complete the classification.

## 5 CONCLUSION

Data mining methods based on machine learning are becoming more and more popular and effective in aero-engine diagnosis, which occupies an important position in safe operations and cost-effective maintenance of aircraft engines.

This paper applied various classification methods to aero-engine component diagnosis, including Support Vector Machine, Decision Tree, K-Nearest Neighbors and Linear Discriminant Analysis, and proposed a fusion algorithm of those 4 methods based on their training accuracy. Classification accuracy and time cost were compared under different conditions including large training samples, small training samples, few features, and noisy data. The results show that SVM has relatively high and stable classification accuracy, but poor time efficiency performance. Linear Discriminant Analysis has relatively good performance in both classification accuracy and time efficiency. The fusion has higher accuracy than any other single method in different cases, but since its algorithm procedure is based on all the 4 single methods, it costs the most time to complete the classification process. The fusion algorithm is a better choice for aero-engine fault diagnosis when the time efficiency requirement is not too strict.

## REFERENCES

[1] X. K. Wei, Y. H. Li, State and Trends of Aeroengine Condition Monitoring and Fault Diagnosis, Control Engineering of China, Vol.14, No.1, 84-87+96, 2007.

[2] S. Sarkar, K. Mukherjee, A. Ray, M. Yasar, Fault diagnosis and isolation in aircraft gas turbine engines, Proceedings of 2008 American Control Conference, 2166-2171, 2008.

[3] M. A. Schwabacher, A survey of data-driven prognostics, Proceedings of the AIAA Infotech@ Aerospace Conference, 7002, 2005.

[4] K. Goebel, B. Saha, A. Saxena, A comparison of three data-driven techniques for prognostics, Proceedings of 62nd meeting of the society for machinery failure prevention technology, 119-131, 2008.

[5] A. Widodo, B. S. Yang, Support vector machine in machine condition monitoring and fault diagnosis, Mechanical systems and signal processing, Vol.21, No.6, 2560-2574, 2007.

[6] H. Heng, J. Zhang, C. Xin, Research on aircraft engine fault detection based on support vector machines, Proceedings of 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), 496-499, 2012.

[7] Q. H. Xu, J. Jun, Fault diagnosis for aero-engine applying a new multi-class support vector algorithm, Chinese Journal of Aeronautics, Vol.19, No.1, 175-182, 2006.

[8] X. K. Wei, C. Lu, C. Wang, J. M. Lu, Y. H. Li, Applications of Support Vector Machines to Aeroengine Fault Diagnosis, Journal of Aerospace Power, Vol.19, No.6, 844-848, 2004.

[9] W. L. Zhao, C. G. Hou, Q. H. Wang, Diagnosis of aircraft engine performance deterioration based on support vector machines, Proceedings of 10th International Conference on Reliability, Maintainability and Safety, 44-48, 2014.

[10] W. Z. Yan, C. J. Li, K. F. Goebel, A multiple classifier system for aircraft engine fault diagnosis, Proceedings of the 50th Meeting of the Machinery Failure Prevention Technology (MFPT) Society, 271-279, 2006.

[11] V. N. Vapnik, The nature of statistical learning theory, Springer Science & Business Media, Berlin, Germany, 2013.

[12] L. Li, Selected Applications of Convex Optimization, Tsinghua University Press, Beijing, China, 2015.

[13] C. C. Chang and C. J. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology, Vol.2, No.3, Article 27, 27:1-27:27, 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[14] J. R. Quinlan, Introduction of Decision Trees, Machine learning, Vol.1, No.1, 81-106, 1986.

[15] R. Timofeev, Classification and regression trees (CART) theory and applications, Humboldt University, Berlin, Germany, 2004.

[16] T. M. Cover, P. E. Hart, Nearest Neighbor Pattern Classification, IEEE transactions on information theory, Vol.13, No.1, 21-27, 1967.

[17] A. J. Izenman, Modern multivariate statistical techniques. Regression, classification and manifold learning, Springer, New York, the United States of American, 2008.

[18] M. Welling, Fisher linear Discriminant Analysis, Department of Computer Science, University of Toronto, Vol.3, 1-4, 2005.