Unsupervised Classification of Vibro-acoustic Signals based on Machine Learning

Professorship for Audio Information Processing
Technische Universität München, Germany
Univ.-Prof. Dr.-Ing. Bernhard U. Seeber

Research Internship's Thesis

Author: Zhen Zhou
Advisor: Norbert Kolotzek; Prof. Dr.-Ing. Bernhard U. Seeber

Started on: 19.10.2020
Handed in on: 09.02.2021

Abstract

In this work we analyze 44 spectral and temporal features of the vibro-acoustic signals from Dec. 2019 to Apr. 2020. Through unsupervised learning methods, we can extract information about the machine state or abnormal behavior from the measurements. This method firstly preprocesses the data, which includes feature selection and PCA(Principal components analysis) dimensionality reduction. Then through self-supervised learning AutoEncoder it can achieve data compression. Finally, the clustering algorithms (DBSCAN, K-Means) can be used to classify the data and analyse abnormal behavior.
On the basis of our methods we show that healthy and unhealthy states can be clearly predicted. At the same time the critical points can also be found.

Key words: PCA, AutoEncoder, Clustering Algorithms

Contents

1

Introduction

This work is based on vibro-acoustic signals. There are 1755 recordings during the time period from Dec. 2019 to Apr. 2020, in other words over 4 months. 1 recording lasts 8 minutes every 90 Minutes. Each file is split into 319 blocks. 44 spectral & temporal features are extracted in each block. Therefore, Total of 559845 data will be analyzed. The goal is to classify the data and hope to obtain effective information from them, such as the health of the machine, abnormal behavior. This determines the use of unsupervised learning in this work. At the same time, the data can also be called a time series, because they occur in chronological order. If they are disrupted, the data will become meaningless.

So this work becomes an unsupervised classification problem based on time series, which reminds me of RNN-Recurrent neural network, because RNN is a neural network based on CNN that specializes in processing sequences. It can predict the next data based on the correlation of the previous continuous data. There are two types algorithms in RNN,classification and regression. However, here RNN cannot be used to classify data directly, because the original data does not contain any labels, the neural network cannot learn from it. If we want to perform RNN, only regression can be used rather than classification. My initial thoughts were: I perform LSTM(RNN) to predict on the data and compare the output data with the original data. If the deviation is too large, then there is an abnormal point here [HL]. After I actually implemented this algorithm, I realized its limitations. This algorithm can only simply distinguish the abnormal points, but cannot obtain other effective information which led to the first failure. When I clarified my thinking again, I realized that clustering algorithm should be used. Then it comes to the final method firstly preprocesses the data, which includes feature selection and PCA(Principal components analysis) dimensionality reduction. Then through self-supervised learning AutoEncoder it can achieve data compression. Finally, the clustering algorithms (DBSCAN, K-Means) can be used to classify the data and analyse abnormal behavior [LS].

2

Theoretical Overview

Our approach consists mainly of the following steps, namely, Data Preprocessing for Artificial Neural Network, performing AutoEncoder - ANN for data compression, and using Clustering algorithm to extract crucial information. The parameters used are discussed in the section 3 technical details.

2.1 Data Preprocessing for ANN

2.1.1 Feature Selection

There are 44 spectral & temporal features, which are extracted from recordings. Some redundant information and noise will exist in the data. Then we need to reduce the data dimensionality of the rows and columns.
For each row, data can be randomly selected in each recording. Since the time interval of the data is relatively small, the selected data can express the information of the original measurement.
For each column, there may be a high degree of correlation between features, which can lead to data redundancy. Therefore, it needs to be judged by correlation. The specific method is: calculate the correlation between each column; select the feature according to the set threshold (the specific threshold is as follows); loop through all the columns, if the correlation between the two columns is greater than the set Threshold, then one of the columns will be deleted.

- `High correlation`: $0.8 \leq |r| \leq 1$.
- `Middle correlation`: $0.3 \leq |r| < 0.8$.
- `Low correlation`: $0 \leq |r| < 0.3$.

2.1.2 PCA - Principal Component Analysis

PCA is the most commonly used linear dimensionality reduction method. Its goal is to map high-dimensional data to a low-dimensional space through a certain linear projection, and expect the largest amount of information in the projected dimension (the largest variance) , in order to use less data dimensions while retaining the characteristics of more original data points. After PCA, the data can be further reduced in dimension [HA10].

2.2 AutoEncoder - ANN

AutoEncoder is composed of Encoder and Decoder. It's self-Supervised Learning and not dependent on labels. The role of AutoEncoder is to compress high-dimensional data into low-dimensional data. It could achieve dimensionality reduction and Anomaly detection. As shown in Figure 2.1, the blue areas represent input and output. The green area is composed of a neural network with many neurons. After neural network training, the compressed area will be obtained, that is pink area - Code. The next steps are reversed. The data will be decompressed. The neural network compares the input and output, finds the prediction error, performs reverse transmission, and gradually improves the accuracy of self-encoding. So it could realize unsupervised dimensionality reduction [FJ16].
The reason I chose this algorithm is that relatively accurate compressed data can be obtained from Code area. At the same time, this algorithm is also different from PCA in that it is nonlinear compression and can better retain the original features. Compared with other unsupervised learning algorithms, its accuracy is also relatively high.

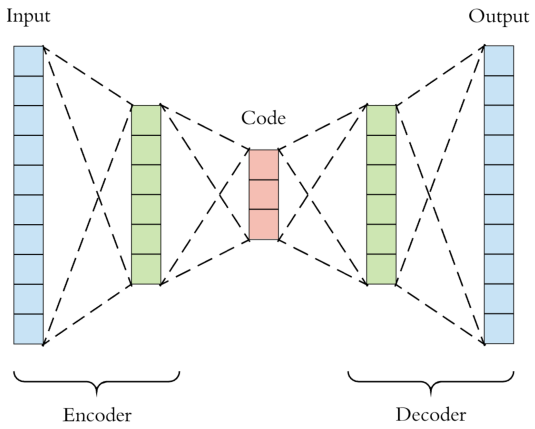


Figure 2.1: AutoEncoder Process[1]

2.3 Clustering Algorithm

There are several clustering algorithms. But some of them perform not very well in this case, like Gaussian Mixture Model. In this work I will mainly introduce DBSCAN and K-Means.

2.3.1 DBSCAN

DBSCAN(Density-Based Spatial Clustering of Applications with Noise), it's density-based clustering algorithm. It divides areas with sufficiently high density into clusters. DBSCAN performs well in noisy spaces. As shown in Figure 2.2, the goal is to classify n objects. The distance between A,B,C and the neighboring point is less than the radius. At the same time they also meet the minimum number. They will be divided into one cluster. The distance between A and N is larger than radius. N will be classified into another cluster. All generated clusters meet the density requirement. In this way, DBSCAN can avoid classification errors due to noise. At the same time, it is more sensitive to abnormal points [AB17].

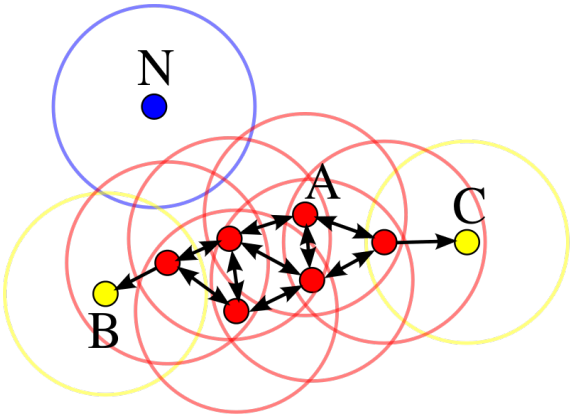


Figure 2.2: DBSCAN Algorithm[2]

2.3.2 K-Means

Kmeans is a distance-based clustering algorithm and has fast calculation speed. As shown in Figure 2.3, it repeatedly divides the data into k clusters according to a certain distance function. The number of clusters needs to be declared in advance. In the k-means algorithm, the similarity of objects in clusters of the same cluster is higher; while the similarity of objects in clusters of different clusters is lower [RA19].

[1]Taken from: https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368 (checked on: 05.02.2021)

[2]Taken from: https://en.wikipedia.org/wiki/DBSCAN (checked on: 05.02.2021)

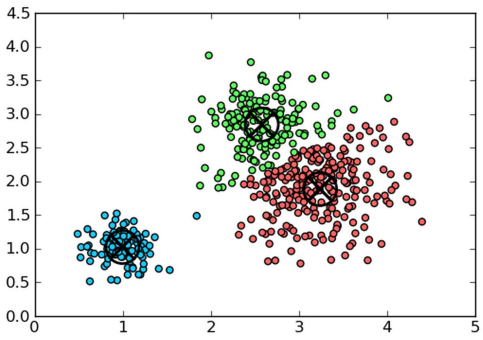


Figure 2.3: K-Means Algorithm[3]

[3]Taken from: https://zhuanlan.zhihu.com/p/37875887 (checked on: 05.02.2021)

3

Technical Details

Here I will specifically introduce the working process and parameter settings. The dimensions of input and output at each step will also be introduced. From the previous explanation, there are currently a total of 559,845 data (rows) and 44 features (columns). The specific process is as follows:

3.1 Data Reduction

Because of redundant information & noise, I select randomly 1 block in each recording file, that is selecting from 319 blocks. Then 1755 data were selected from 559,845 data. The number of features remains unchanged. The Current dimension is (1755, 44).

3.2 Parameters of Features Selection

According to the algorithm mentioned before, we can delete columns with high correlation. In order to retain the appropriate number of columns, here I choose threshold = 0.6. That means, one column will be removed when its correlation with other columns is greater than this threshold(0.6). The correlation heatmap as shown in Figure 3.1. There is no light-colored area, which means that the correlation between each column is weak. And there are not only temporal features, but also spectral features. The current dimension becomes (1755, 15).

According to the data visualization in Figure 3.2, the following two columns will be deleted. Because there is a lot of noise in these two features. We can't get a clear trend from them.

Then remained features are shown in Figure 3.3. It is worth noting that Current dimension is (1755, 13).

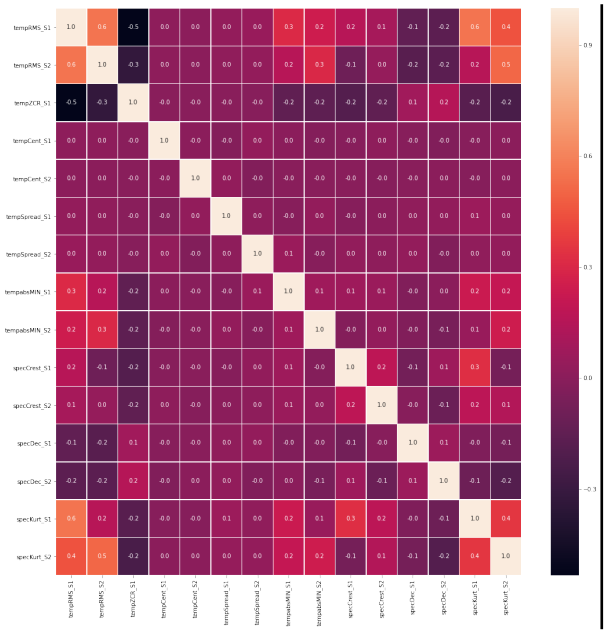


Figure 3.1: Correlation Heatmap

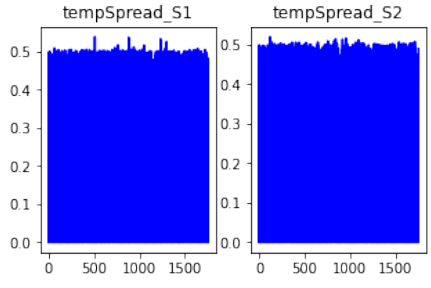


Figure 3.2: Noisy Features

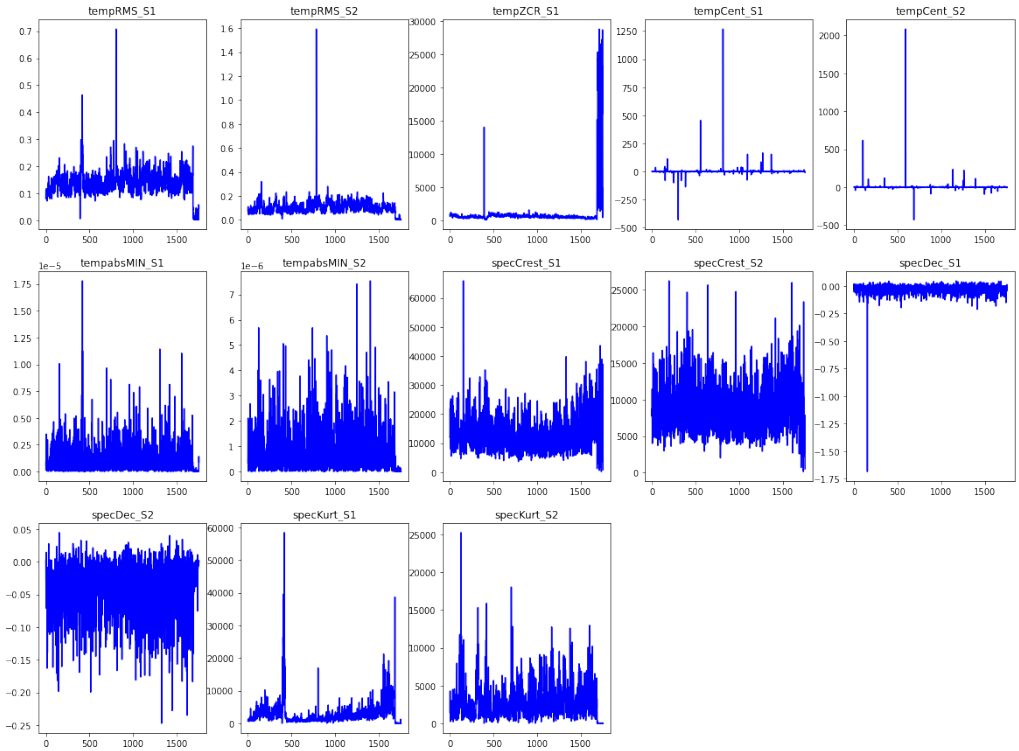


Figure 3.3: Remained Features

3.3 Parameters of PCA

PCA could reduce the dimensions, remove the noise and redundant information. So we need to perform this step. From the parameter of 'pca.explained_variance_ratio_' I can know how much variance the remained components will express. Typically, nearly 95% of the variance can express the entire data. I choose first 10 main components, which can express 94% of the variance. In Figure 3.4, it shows how much variance each component can express.

```
[0.23035686 0.11014015 0.08255235 0.0789062  0.07664501 0.07627441
 0.07062564 0.065439   0.06156088 0.05730247]
```

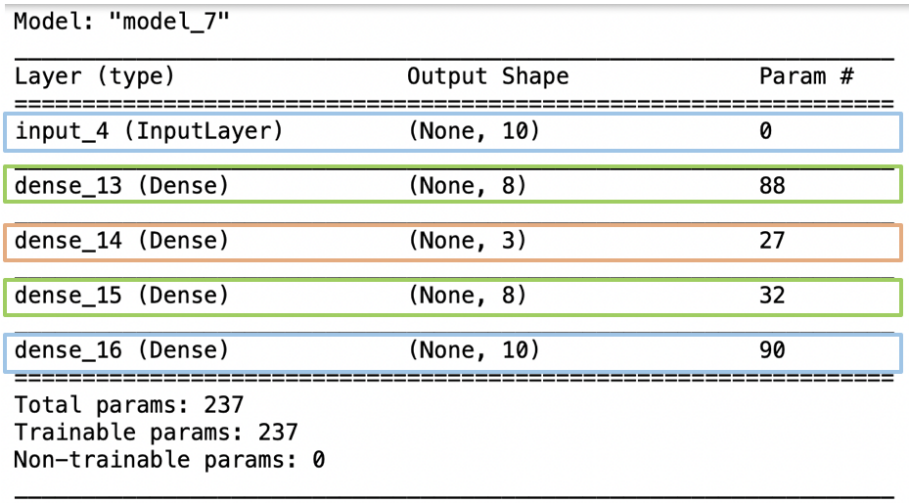
Figure 3.4: Ratio Variance

3.4 Data Standardization

Data standardization could linearly map each dimension feature to the specified interval. It can keep the trend unchanged as the original data. At the same time, it also prepares for the following neural network training. The value range of ReLU activation function is [0, 1]. So I choose this interval, [0, 1]. Now Current dimension is (1755, 10).

3.5 Training AutoEncoder

The structure of the entire model is listed Figure 3.5. Each color box corresponds to a different area in Figure 2.1. The blue boxes correspond to input and output. Here I choose the input dimension equal to (,10). The green box corresponds to the neural network area. Here I choose the dimension of dense layer equal to (,10). The pink box corresponds to the code area. Here I choose the compressed dimension equal to (,3). And finally we only need this part of data. It can be clearly seen that the features are compressed from 10 to 3. The current dimension is (1755, 3).



```
Model: "model_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         (None, 10)                0
_________________________________________________________________
dense_13 (Dense)             (None, 8)                 88
_________________________________________________________________
dense_14 (Dense)             (None, 3)                 27
_________________________________________________________________
dense_15 (Dense)             (None, 8)                 32
_________________________________________________________________
dense_16 (Dense)             (None, 10)                90
=================================================================
Total params: 237
Trainable params: 237
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.5: AutoEncoder Model

4

Results

After training the neural network, we can get the output from AutoEncoder. As shown in Figure 4.1, we can see the 3D graph output from AutoEncoder. This three-dimensional picture represents distribution of reduced data. Because of linear combination the distribution is more clearly than PCA. From the distribution of points, we can see obvious outliers. As for the value of the coordinate axis, it has no specific meaning to us. Now we can analyse the results based on different clustering algorithms.

4.1 Results based on Clustering-DBSCAN

In Clustering-DBSCAN algorithm after several trying, I set 1)radius e=0.3, 2)minimum points MinPts = 5. It's obvious to see the abnormal points as shown in Figure 4.2, which are purple points.

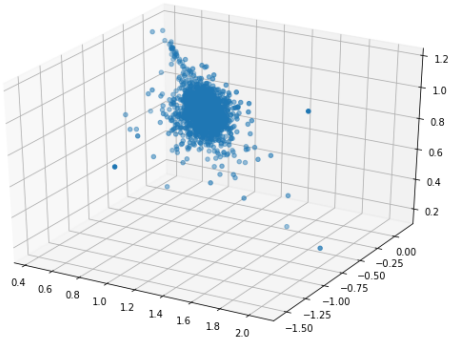


Figure 4.1: Output from AutoEncoder

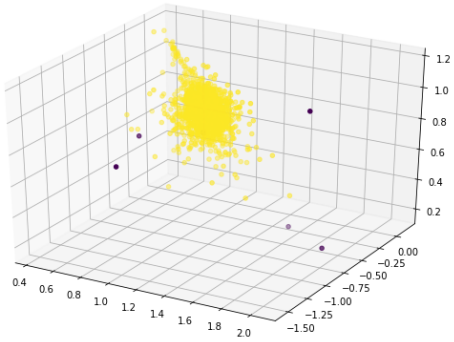


Figure 4.2: Output from DBSCAN

Figure 4.3 shows the result for classification. There are two clusters, label '0' and '-1'. Label '-1' means abnormal behavior. The bottom row corresponds to the

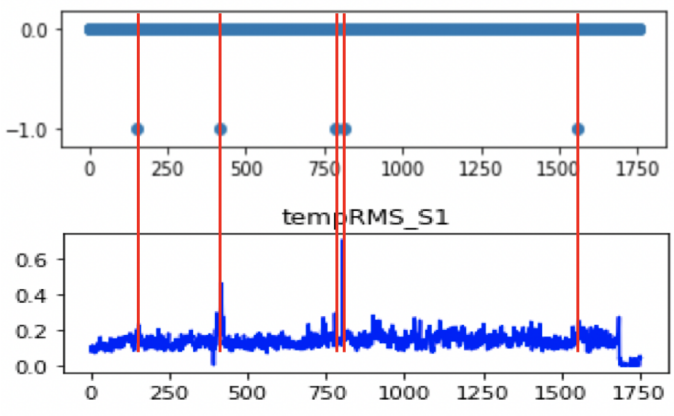


Figure 4.3: Classification Results based on DBSCAN

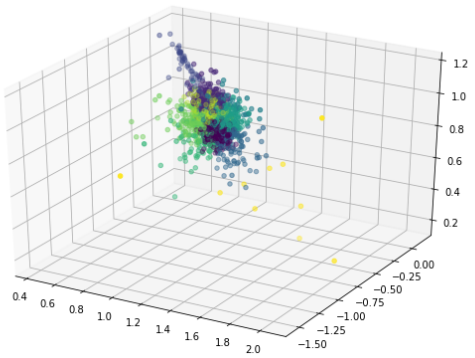


Figure 4.4: Output from KMeans

original data index. Comparing the labels with the original features, we could see each '-1' label corresponds to an abnormal peak. Although we can observe abnormal situations, there is no temporal increasing or decreasing for heath status.

4.2 Results based on Clustering-KMeans

In Clustering-KMeans algorithm, I set number of clusters Num_cluster=10. Figure 4.4 shows the output 3D graph from KMeans. Currently we could only see the distribution of clusters.

As shown in Figure 4.5 it's the result for classification. There are 10 clusters, from label '0' to label '9'. Some labels show abnormal behavior, that is label 3,5 and 9. Others need further analysis. For example, in label 1 and 4, it increases in occurrence over time.

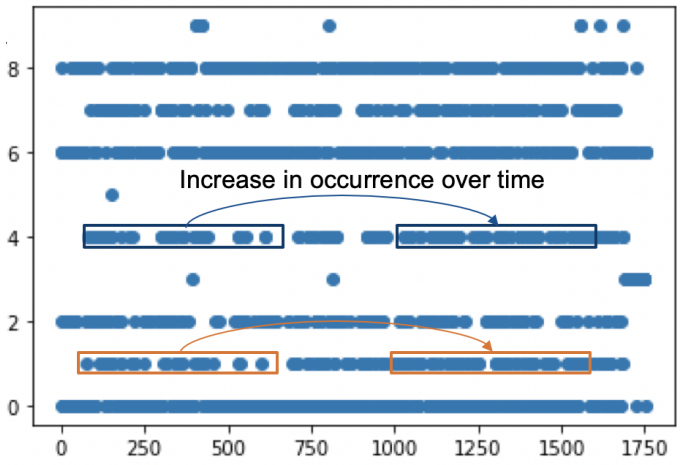


Figure 4.5: Classification Results based on KMeans

For further analysis, appearing frequency of each cluster is shown in Figure 4.6. That is, how often are the clusters chosen over time? In each cluster, I calculate the frequency of occurrence every 39 data, that is 3 days. In 1st picture of Figure 4.6, it shows a decreasing trend, which represents a healthy state. On the contrary, 8th picture of Figure 4.6 shows an increasing trend, which means an unhealthy state. In 2nd picture of Figure 4.6, the red circled part indicates the critical points. In 3rd picture of Figure 4.6 firstly it shows the constant state. Then there are critical points after something happened. And then it shows a healthy state because of decreasing trend. 4th,6th,10th picture of Figure 4.6 indicate abnormal points, which is consistent with the previous DBSCAN.

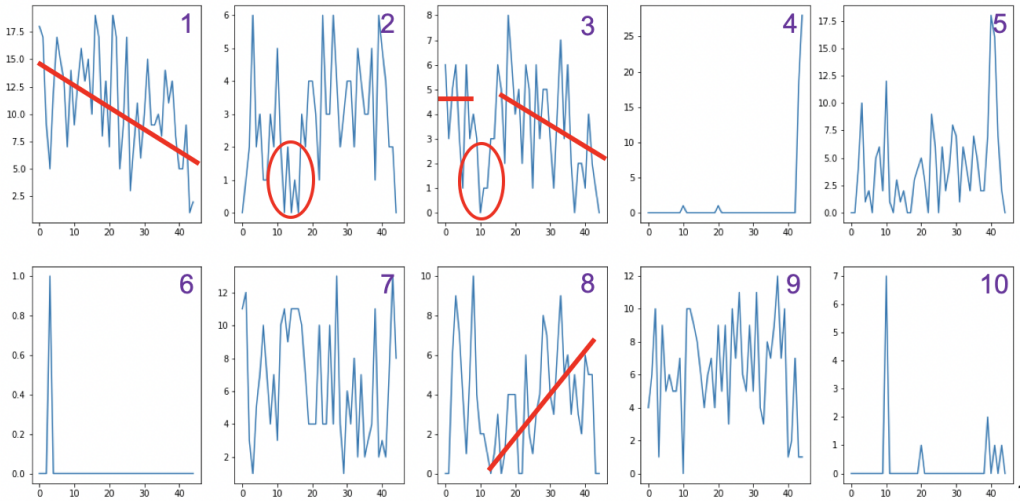


Figure 4.6: Appearing Frequency of each Cluster

5

Conclusion and Outlook

5.1 Conclusion

Here I will summarize what I have achieved till now. Through feature selection, PCA and other methods, the data will be preprocessed. In this way, a preliminary dimensionality reduction is achieved. After that self-supervised learning AutoEncoder can realize nonlinear data compression. Till now I have transformed wealth of features (high dimensionality) to meaningful low dimensionality. And then with clustering algorithms I detect critical / abnormal behavior. After clustering the increasing or decreasing trend could be used as a prediction for health status. Two detection goals can be achieved, that is abnormal behavior and health status.

5.2 Outlook

Since there will be errors in manually marking labels, this method can solve the problem of classifying data without labels. At the same time, AutoEncoder performs well in unsupervised learning. But it also has certain limitations. The clustering algorithm has randomness and uncertainty. Because there is no training set that can be used for comparison, we can't evaluate the clustering algorithm. Therefore, we can roughly get the features of trends and abnormal points. If we want to get more accurate results, we need to add other algorithms to optimize this method.

Bibliography

[AB17] Krzysztof Cios Avory Bryant. RNN-DBSCAN: A Density-Based Clustering Algorithm Using Reverse Nearest Neighbor Density Estimates. *IEEE*, 30:1109–1121, 2017.

[FJ16] Jing Lin Xin Zhou Na Lu Feng Jia, Yaguo Lei. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, 72-73:303–315, 2016.

[HA10] Lynne J. Williams Hervé Abdi. Principal component analysis. *Wires Computational Statistics*, 2:387–515, 2010.

[HL] Kyogu Lee Yoonchang Han Hyungui Lim, Jeongsoo Park. Rare sound event detection using 1D convolutional recurrent neural networks. *Detection and Classification of Acoustic Scenes and Events*.

[LS] Qi Linhai Lin Shan, Wang Hong. A Method for Recognizing and Repairing Power Load Abnormal Data Based on Density Clustering and LSTM. *cnki.net*.

[RA19] Anand Nayyar Rishabh Ahuja, Arun Solanki. Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor. *IEEE*, pages 263–268, 2019.