FFT-2PCA: A New Feature Extraction Method for Data-Based Fault Detection

Matheus Maia de Souza, João Cesar Netto, and Renata Galante(✉)

Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil
{mmsouza,netto,galante}@inf.ufrgs.br

Abstract. The industrial environment requires constant attention for faults on processes. This concern has central importance both for workers safety and process efficiency. Modern Process Automation Systems are capable of produce a large amount of data; upon this data, machine learning algorithms can be trained to detect faults. However, this data high complexity and dimensionality causes a decrease in these algorithms quality metrics. In this work, we introduce a new feature extraction method to improve the quality metrics of data-based fault detection. Our method uses a Fast Fourier Transform (FFT) to extract a temporal signature from the input data, to reduce the feature dimensionality generated by signature extraction, we apply a sequence of Principal Component Analysis (PCA). Then, the feature extraction output feeds a classification algorithm. We achieve an overall improvement of 17.4% on F1 metric for the ANN classifier. Also, due to intrinsic FFT characteristics, we verified a meaningful reduction in development time for data-based fault detection solution.

Keywords: Feature extraction · Fault detection · Tennessee Eastman process

1 Introduction

The industrial automation level achieved in the last decades allow improvements in production and safety. Increase in the sophistication of process control systems has not eliminated abnormal situations. Consequently, faults in a complex environment, like the factory floor, are inevitable. These faults may cause from a reduction in the process performance to injuries in workers [2]. A single fault can cause many process variables to exceed their limits. According to [7], faults occur in the industrial processes that produce undesired or unacceptable system behavior. For instance, in hazardous processes as, e.g., chemical plants, the consequences of a fault can be disastrous.

In this work, we discuss only data-based techniques. Data-based techniques are more generic, as there is no need for in-depth knowledge of the target system. Two groups divide data-based techniques, those that use statistical methods and those that use machine learning methods. This paper presents a new feature

This work is supported by the BNDES under the FUNTEC-SDCD project.

S. Hartmann et al. (Eds.): DEXA 2019, LNCS 11706, pp. 215–224, 2019.
https://doi.org/10.1007/978-3-030-27615-7_16

extraction method for machine learning fault detection. The new method is based on the frequency spectrum extraction of input signals, using a discrete Fourier Transform. Input signals will later be classified according to their states, fault or normal. Given the natural cyclical behavior of industrial environments, the frequency spectrum functions as a signature of behavior that can be classified as normal behavior or fault behavior. To verify if the use of these behavioral signatures produces a performance gain, compared to the use of data in the time domain, we performed experiments using datasets generated by the *Tennessee Eastman* industrial control simulation [5]. The simulation *Tennessee Eastman* was chosen due to the control of the environment that the simulation offers. The measure used to compare the techniques will be the precision with which the algorithm, or set of algorithms, can detect a failure.

The rest of the paper is structured as follows. Section 3 provides a review of the literature. Section 2 describes the background. In Sect. 4, we discuss the method FFT-2PCA, showing the functional steps and components. Section 5 presents the experiments to implement the proposed method along with the acquired results, while discusses these results and their application to actual setups. Section 7 concludes the paper, providing directions for future work.

2 Background

We begin with our background section with data source; the Tennessee Eastman process (TEP), an industrial process simulation which has an academic use in the industrial control field. The TEP is a suitable data source due to its capability to provide a structured data output of a complex system.

The Fast Fourier Transform (FFT) is used to determine the frequencies and phase content of local sections of a signal as it changes over time [3]. Its uses extend from data compression to image processing. In our work, we apply the FFT to obtain a frequency signature that intends to extract the time series features created by the TEP.

Besides our method, FFT-2PCA, we use two other machine learning algorithms to extract features. The first one, Principal Component Analysis (PCA), is a well-known method for dimensionality reduction, it does that by extracting a new set of orthogonal variables from interdependent ones. To do so, it applies singular value decomposition to find these orthogonal variables, also called principal components [1]. The PCs are the new dimensional space in which the original variables will be represented. The second one, Independent Component Analysis (ICA), aim to maximize independence. It finds a linear transformation, which applied to the original set of variables, results in a new space where the independence between variables is maximized [4].

3 Related Work

In [12], authors present a comparison between a set of statistical tools to perform process monitoring and fault detection. The tools chosen were: PCA, Partial Least Squares (PLS), independent component analysis (ICA), and variations of

each technique, and a method based on the model of the process to be monitored. The statistical tests T^2 and squared prediction error (SPE) are used to classify a sample as a normal state or fault state. The results show that the techniques that use PLS obtained higher accuracy and lowered false positive rate. However, these statistical measures, with thresholds manually selected, makes it challenging to use on a large scale since each class of failure requires individual analysis.

In [8], the authors compare two techniques for fault diagnosis, a process to identify a fault class. One based on PCA and another one based on Support Vector Machines (SVM). Like in [12], PCA is a failure detection method having as a deciding factor a threshold for a T^2 test that defines if a sample represents a failure situation and its class. The results show the PCA as a better option since it has less computational effort to obtain better results. To evaluate performance, the authors chose to measure just accuracy. Since accuracy only computes the ratio of correct predictions over total predictions, the results of false negatives are neglected.

In [11], authors use a combination of ANN and PCA as an approach to fault detection and fault diagnosis. The paper hypothesis is that a dimensionality reduction improves classification performance. They apply fault detection for each fault class getting the best overall detection rate of 84.32%. Authors also tested the approach in fault diagnosis obtained the best overall diagnosis rate of 48.29%. In this work again, authors use a fragile methodology to evaluate performance by using only the missed detection rate as a metric. Due to a substantial similarity with our work and being state of the art in our domain, we define this work as our baseline.

The raw input data that comes from multiple sensors in an industrial environment are complex and noisy. All cited works face this problem and make an effort to extract useful information from the input data to perform fault detection with better results.

4 FFT-2PCA

With the rise of industrial automation, processes that have been pure analog now are part of the digital world. This transformation brought the challenge to keep the environment safe while dealing with a sheer amount of data. Our contribution to face this challenge is a feature extraction method designed for data-based fault detection. The data collected in the industrial environment, used as input for the classification, carry some inherent problems, such as noisy and redundant information. Our method aims to mitigate these problems by extracting relevant features in the time domain and taking them to a representation space where the classifiers can more easily distinguish the states of operation.

The Algorithm 1 aims to extract proper signatures from all-time series, providing the time-related knowledge to the classifiers. Our input data comprise of time series, organized in a matrix form. As the name time series implies, each data point is directly dependent on the previous ones. Simple classification algorithms do not take into account this connection, assessing only one data point at a time. The classifier receives the extracted signatures as a matrix that now expresses a representation space created by our algorithm.

Algorithm 1: FFT-2PCA algorithm

Input : $A[m, n]$, k, pcs1, pcs2
Output: $E[(\frac{m}{k}), n \cdot pcs2]$

1 $B[n(\frac{m}{k}), \frac{k}{2}] \leftarrow$ `Dff(`$A[m, n]$`,k)`;
2 $Bnorm[n(\frac{m}{k}), \frac{k}{2}] \leftarrow$ `Normalize(`$B[(\frac{m}{k})n, \frac{k}{2}]$`)`;
3 $C[(\frac{m}{k}), n(\frac{k}{2})] \leftarrow$ `Pivoting(`$B[n(\frac{m}{k}), \frac{k}{2}]$`)`;
4 pca = `setPCA(`$pcs1$`)`;
5 `pca.fit(`$C[(\frac{m}{k}), n(\frac{k}{2})]$`)`;
6 $D[(\frac{m}{k}), n \cdot pcs1] \leftarrow$ `pca.transform(`$C[(\frac{m}{k}), n(\frac{k}{2})]$`)`;
7 pca = `setPCA(`$pcs2$`)`;
8 `pca.fit(`$D[(\frac{m}{k}), n \cdot pcs1]$`)`;
9 $E[(\frac{m}{k}), pcs2] \leftarrow$ `pca.transform(`$D[(\frac{m}{k}), n \cdot pcs1]$`)`;

The first step is data entry; the FFT-2PCA receives a matrix $A_{m,n}$ with m being the number of observations and n the number of system features. To apply the FFT to the matrix $A_{m,n}$, we have to use the transform in its discrete form. This transform requires a parameter k that defines how many observations are going to be used in one transformation. Then we can define arrays with length k for each of n features in the input matrix, these arrays are, separately, transformed by the discrete FFT to the frequency domain. The matrix $B_{\left(\frac{m}{k}\right)\cdot n, \frac{k}{2}}$ shows the result of the transformation on the previous step, the second matrix has $\frac{k}{2}$ columns due characteristics of the transformation, and each observation became n rows in the second matrix. This goes on for all block of k rows in the matrix $A_{m,n}$ resulting in a matrix $B_{\left(\frac{m}{k}\right)\cdot n, \frac{k}{2}}$.

To get again a matrix with rows as observations, we rearrange the matrix $B_{\left(\frac{m}{k}\right)\cdot n, \frac{k}{2}}$, each block of n rows is concatenated in one row, resulting in matrix $C_{(\frac{m}{k}), n(\frac{k}{2})}$. To reduce the dimensionality we apply the first PCA, selecting the more representative FFT weights. The result is the matrix $D_{(\frac{m}{k}), n \cdot pcs1}$ that has its columns as the principal components. So the number of columns will be given by pc1 multiplied by n system features. The last step selects relevant Feature-PC values applying a second PCA. The parameter pcs2 controls the number of PCs; consequently, the number of columns in the last matrix. Finally, we obtain a matrix $E_{(\frac{m}{K}), pcs2}$ that is the input for the classification algorithms.

5 Experiments

Our experiments aim to evaluate how each feature extractors impact classification performance. Our main assumption is that the FFT-2PCA method generates a better input for the classifiers, by extracting the frequency signatures and latter creating a more efficient representation space. Beyond the quality performance, we assess how the feature extractors impact the classifiers regarding training time.

Datasets. We create seven datasets were each one represents a TEP configuration. The structure that keeps this data is a matrix where the columns are the 53 simulation parameters, for instance, internal reactor temperature or the apertures that control the elements mixture. The rows are an observation, also called data point. All seven datasets have the data points count balanced between the two states: faulty or normal. The datapoints count in each dataset respectively is: 261 k; 261 K; 522 K; 783 K; 1.02 G; 1.3 G; 1.56 G; 1.82 G.

Baseline Model. We choose the work "Integration of principal component analysis and neural classifier for fault detection and diagnosis of Tennessee Eastman process" [11], present in Sect. 3, as our baseline. This choice is based on the similarity and relevance to our work. The primary goal of our baseline is to test the impact of the PCA algorithm as the feature extractor for fault classification and diagnosis using ANN. For our work, we choose to focus only on the fault detection problem, to test with two more classifiers (Key-Nearest Neighbor (KNN) and SVM) and bring another algorithm for feature extraction (ICA).

Metrics. To evaluate performance, we use the F1 score that is a harmonic mean of precision and recall. The positive concept, in our case, is related to a fault. For instance, we called a false positive when the classification algorithm gives us a fault indication for a given instance of data, and in the real world, that same instance represents a normal behavior.

Performance Experiments. To evaluate the performance we combine all three feature extractors (PCA, ICA, and FFT-2PCA) with all three classifiers. Also, to test if the use of a given feature extractor results in a significant improvement, we test the classifier without any feature extractor. We set up the FFT-2PCA with parameters as follows: $K = 200$, $pc1 = 100$. The $pc2$ parameter assumes the values: 1000, 2000, 3000, 4000, 5200. These values were chosen to maintain the number of pcs consistently spaced, as in our baseline that configures the PCA with pc values: 10, 20, 30, 40, 52. The ICA and PCA components follow our baseline setup. Each feature extractor configuration paired with a classifier execute all seven datasets 30 times. Then, the F1 results are summarized by dataset calculating the mean for all configurations. To test the statistical significance, we use the Kruskal-Wallis test with a 99% confidence interval.

The ANN was built using a similar architecture of our baseline, with an input layer that matches the dimensionality of PCA or FFT-2PCA output, two hidden layers with 40 and 15 neurons, respectively, and one neuron as the output layer. Except for the output layer that uses a sigmoid, all layers use Relu [10] as the activation function. For the loss function, we use binary cross-entropy. For the KNN classifier, we chose a k-value as five, uniform weights and Minkowski [9] as the distance metric. The SVM classifier is used in Linear [6] form, to handle the amount of data, with the l2 penalty and squared hinge as the loss function. We use two libraries as resources for our method, Keras to build and train the Neural Networks, and Scikit-learn for FFT, PCA, ICA, SVM, and KNN.

Timing Experiments. The experiments regarding the timing characteristics for the pairs feature extractors/classifiers follow the same configuration used in the performance setup. We begin measuring the required time for the feature extraction phase. Also, we perform a time testing for the loading phase, to isolate the weight of the feature extraction in this first phase. Following the timing experiments, we capture the training time spent by each pair feature extractor/classifier. Then, we calculate the mean for each dataset, to verify the impact of the size of the different datasets on the timing results. The statistical test used was the Wilcoxon test, with a 99% confidence interval. The machine used in all timing tests has a processor Intel I5-6500 and 16 GB of DDR3 memory.

6 Results

Performance. Figure 1 shows the results of the experiments for each pair of feature extractor and classifier. Each box-plot inform the distribution, median, and mean of experiments using one of our seven datasets. The KNN classifier had the best performance between our three classifiers. Being a parametric algorithm, the KNN benefits from the dataset size and balance. It is worth notice that in a real fault detection application, these conditions are hardly achieved, but in this work, we use these artificial datasets to establish a parameter to comparison to our baseline and future works. Regarding feature extraction, the PCA had a small impact on the KNN performance; improving only 0.93% the overall performance. Our method, FFT-2PCA, did not improve the KNN classifier.

For the SVM and ANN, the FFT-2PCA improve the overall performance. As we can see on Fig. 1, only when applied to the dataset one the FFT-2PCA, in conjunction with the respective classifier, do not improve the F1 performance. More precisely, we achieve the best improvements when applying the FFT-2PCA with the ANN classifier. The increase in the F1 metric was 17.4%; in comparison with our baseline, the improvement was 48.7%.

Table 1. Overall F1 values for each pair Feature Extractor and Classifier.

ANN	F1	KNN	F1	SVM	F1
FFT-2PCA	**0.75**	PCA	**0.86**	FFT-2PCA	**0.73**
Plain	0.64	Plain	0.85	Plain	0.70
ICA	0.51	ICA	0.84	PCA	0.65
PCA	0.50	FFT-2PCA	0.82	ICA	0.62

Table 1 shows the overall results for each pair feature extractor and classifier. The overall F1 score is the mean of all experiments results across the seven datasets. Each column represents a classifier with rows populated by the feature extractor method; they are ordered by the F1 overall score presented in the right side of each classifier column.

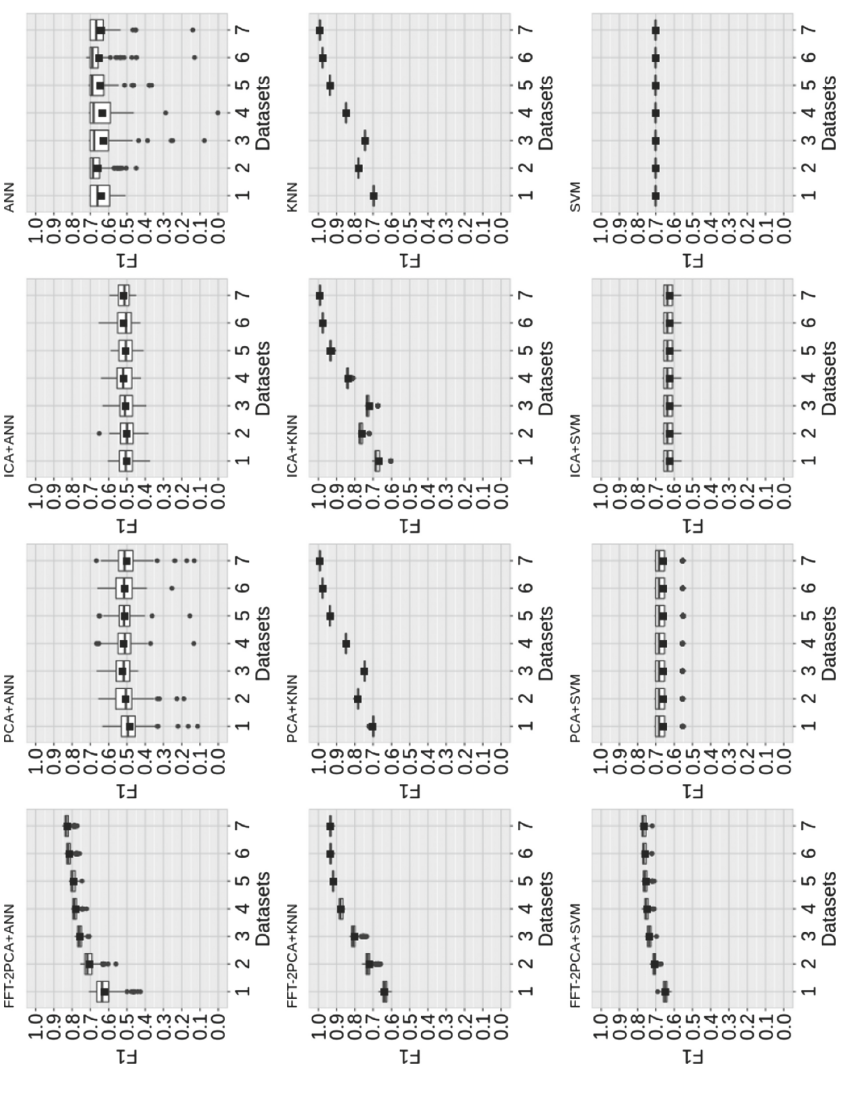
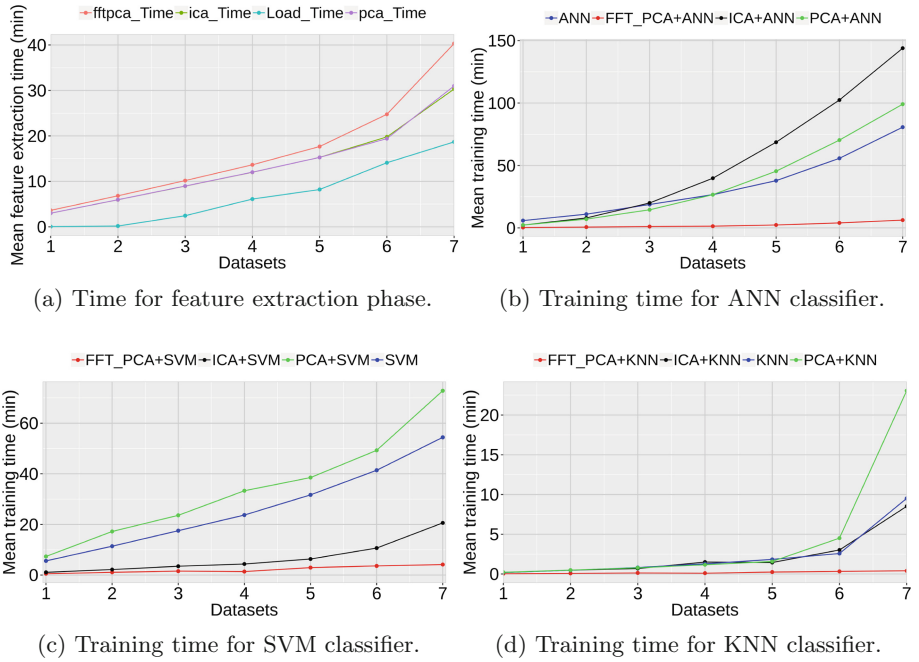


Fig. 1. Box plot graph of F1 measurement for all combinations of feature extraction setup and classifier.

Timing. Figure 2a shows the results for the data load time and feature extraction time applied on all seven datasets. The result for features extraction includes the data loading time too. The graph shows close time results for FFT-2PCA and the other two feature extractors, but, as the data size grows, they drifted. Figure 2a also show that the timing for PCA and ICA feature extractors are close. The PCA and ICA are the only pair that did not achieve statistical relevance.



(a) Time for feature extraction phase.

(b) Training time for ANN classifier.

(c) Training time for SVM classifier.

(d) Training time for KNN classifier.

Fig. 2. Time measurements.

The next time aspect to investigate is the training time for our classifiers. Figure 2b shows the time results for all feature extractor and the plain ANN classifier. The FFT-2PCA was the only feature extractor that showed significant improvement in training time for the ANN classifier. The ICA and PCA did not present a significant difference in comparison with the plain ANN.

Figure 2c show the same configuration than before but applied to SVM classifier. In this case, all results present a significant difference between each other. It means that the ICA and FFT-2PCA can improve the training time for the SVM classifier.

Last training time graph, showed in Fig. 2d, shows the values for the KNN classifier. Again the only feature extractor that achieves a statistical improvement was the FFT-2PCA.

Table 2 shows the average time (in minutes) for the entire process, feature extraction, and training, for all seven datasets. In this table, we can see how

Table 2. Average feature extraction time + average training time (in minutes).

Datasets	FFT-2PCA			ICA			PCA			Plain		
	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN	SVM
1	3.90	3.66	4.15	5.20	3.19	4.08	5.27	3.21	10.35	5.88	0.26	5.66
2	7.45	6.90	7.91	13.88	6.43	8.13	12.94	6.45	23.18	11.13	0.65	11.58
3	11.24	10.31	11.73	28.98	9.64	12.44	23.53	9.76	32.58	21.23	3.25	19.97
4	15.00	13.75	15.02	51.79	13.50	16.37	38.68	13.18	45.33	32.76	7.39	29.82
5	19.99	17.91	20.61	83.93	16.72	21.62	60.71	16.85	53.79	46.09	10.05	39.89
6	28.72	25.07	28.36	122.22	22.80	30.45	89.73	23.91	68.67	69.85	16.67	55.49
7	**46.51**	40.71	44.46	174.23	38.82	50.92	130.10	54.05	103.83	**99.38**	28.20	73.11

FFT-2PCA improve the development time of fault detection solutions. Again, our method, in conjunction with KNN, does not result in an improvement. However, for the SVM and ANN, the FTT-2PCA improve the total time. For the ANN case, the improvement is particularly interesting. As the dataset size increases, the impact of the training times becomes more relevant in the total time, causing the total time reductions brought by the FFT-2PCA to be more prominent. In the best case, a 53% reduction in the total development time of the solution.

Discussion. We begin our discussion reasoning about our classifiers nature. Two of our classifier, KNN, and SVM, are non-parametric classifiers. This trait has a few implications; One benefit is that the classifier can be more flexible, fitting a large number of functions. However, to achieve good performance, a large amount of data must be provided. We can see this behavior in Fig. 1, which is showing a major increase in F1 for the KNN classifier as the dataset size increases. With the fifth dataset, the gain reaches a plateau, and there is not a major increase. The SVM reach its limits right on the first dataset, except when using the FFT-2PCA as a feature extractor that provides a small gain.

Our third classifier, ANN, is a parametric algorithm. This property imposes less dependency on large quantities of training data but also brings more difficulties to fit the hidden function. We can see this caveat in Fig. 1, by looking to results for the ANN classifier. In this case, the F1 metric has more dispersion and more outliers when compared with the SVM and KNN.

With all these concepts in mind, we can think about the applicability of a given fault detection solution beyond the artificial datasets. In real applications, well-annotated fault behavior can be scarce, imposing a limit in size for the training datasets. This factor can limit the use of classifiers, such as KNN and SVM.

As our results showed, the FFT-2PCA brought the most critical improvement were achieved when combined with the ANN classifier. Not address only the problem of the better fitting, but also bring a large improvement in training times. These results show us a promising path for the development of a solution for fault detection based in ANN with our FFT-2PCA as a feature extractor.

7 Conclusion

This work introduces a feature extraction method for fault detection. This method called FFT-2PCA applies Fast Fourier Transform to a range of data points, after, a sequence of PCA reduces the dimensionality, then the resulting data entered in the classification algorithm. Our method was tested against other two feature extractors, ICA and PCA, and the classifiers without any preprocessing. The FFT-2PCA showed the best results when applied with the ANN classifier, a 17.4% improvement in performance and a 54% reduction in training time when compared with the plain ANN classifier. With these results, we can consider the FFT-2PCA a good option in the development of a fault detection solution based in ANN.

In our future work, we plan to expand our tests on the fault diagnosis problem. A fault diagnosis solution consists in not only detect a fault but identify the class that fault belongs. We intend to explore other aspects that we do not address in this work, such as the fault classes balance in our datasets.

References

1. Abdi, H., Williams, L.J.: Principal component analysis. Wiley Interdiscip. Rev. Comput. Stat. **2**(4), 433–459 (2010)
2. Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., Schröder, J.: Diagnosis and Fault-Tolerant Control, vol. 691. Springer, Heidelberg (2006). https://doi.org/10.1007/978-3-540-35653-0
3. Brigham, E.O., Brigham, E.: The Fast Fourier Transform and Its Applications, vol. 1. Prentice Hall, Englewood Cliffs (1988)
4. Comon, P.: Independent component analysis, a new concept? Signal Process. **36**(3), 287–314 (1994)
5. Downs, J.J., Vogel, E.F.: A plant-wide industrial process control problem. Comput. Chem. Eng. **17**(3), 245–255 (1993)
6. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
7. Frank, P.M., Blanke, M.: Fault diagnosis and fault-tolerant control. In: Control Systems, Robotics and Automation XVI (2007)
8. Jing, C., Hou, J.: SVM and PCA based fault classification approaches for complicated industrial process. Neurocomputing **167**, 636–642 (2015)
9. Kruskal, J.B.: Nonmetric multidimensional scaling: a numerical method. Psychometrika **29**(2), 115–129 (1964)
10. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pp. 807–814 (2010)
11. Nashalji, M.N., Arvand, S., Norouzifard, M.: Integration of principal component analysis and neural classifier for fault detection and diagnosis of Tennessee Eastman process. In: 2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T), pp. 166–170. IEEE (2014)
12. Yin, S., Ding, S.X., Haghani, A., Hao, H., Zhang, P.: A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process. J. Process Control **22**(9), 1567–1581 (2012)