Fast failure detection in robotics using supervised learning method with convolutional gaussian processes

Yang YOU
y.you@cranfield.ac.uk

***Abstract*— The execution failure is a classical problem in robotics and much progress has been observed in recent years with supervised learning methods. In this paper, a method, which applying gaussian kernel convoluted data then using different machine learning algorithms and neural networks method to perform this classification task, shows a higher accuracy in prediction and less time consuming in training than the existing methods without convolution process.**

Keywords: Convolution process, gaussian kernel, machine leaning, neural networks, failure detection, less time consuming.

I. INTRODUCTION

Robots are complex and need to interact with the environment continuously and quickly. With the progression of machine learning and artificial intelligence, much of them can be deployed into robots' applications. For example, deep reinforcement learning algorithms like DDPG [1] and HER [2] can be used to train robot's manipulation [3] and convolution neural network [4] for robot vision. However, as evolves, the robot system become more complicate to control, which lead to a rise of execution's failure and uncontrollable status. Hence, a more accurate and faster prediction of failure becomes crucial.

At the hardware level, techniques such as sensor placement optimization [5] are used to obtain a fast and robust robot system. Regarding to the failure detection using captured data, the difficulties are the relevance of the data from each sensor and environmental noises. However, due to the limitation of computing resource on the robot, a complex model will require more time and causing further risks. Thus, with the condition of fast training, the key points to improve the accuracy of prediction are extracting features and eliminate the influence of noise. The existing supervised method combined with principal component analysis [6][7] shows a improvement in prediction result. However, due to the relevance between sensor data, it would be better to consider extract the local features and get rid of the influence of noise. This idea has been confirmed by this experiment, where a significant increase of success rate can be observed in the prediction accuracy. Moreover, the convolution process is written in C++ in this experiment which ensured the time efficiency of computing.

The structure of this paper is designed as follows. The next section will introduce the feature extraction for the raw data. The Section 3 will give a brief introduction of classification, related machine learning methods and explanation of the techniques used in this experiment. The performance of purposed method and existing method will be displayed in Section 4, and the results will be analyzed and concluded in the last section.

[1]Source code can be found at https://github.com/yangyou95/Robot-Failure-Detection–A-convolutional-method

[2]Back propagation code was provided in folder Neural_Network_Codes

II. FEATURE EXTRACTION

Feature extraction is a process which can reduce dimensions of data. This process can reduce the number of variables in one data set without losing the accuracy and competency of the original data. Two classical algorithms of feature extraction will be introduced shortly.

A. Principal Component Analysis

This method is used to find the optimal subspace of data distribution [6]. The key idea is to select the first N biggest eigenvealues in the covariance matrix, their eigenvectors are used to have a linear map to the original data. Dimensions corresponding to the smallest eigenvalues are omitted which lead to a dimensional reduction. The mean square error ϵ^2 is given by

$$\epsilon^2 = trace(cov(x)) - \sum_{i=1}^{L} \lambda_i = \sum_{i=L+1}^{M} \lambda_i$$

However, the linear model also limits the relation between the principal component and original data can only be linear, thus methods like Kernel PCA [8] can be used as a nonlinear process extension.

B. Independent components analysis

Unlike the principal component analysis extract the non-relevant variables to reduce the dimension, the independent components analysis method reduce the dimension by extracting and selecting the most independent variables [9]. The main idea is to find a linear transformation

$$z = Wxz = Wx$$

such that the basis vectors in Z have the maximum independence [10].

III. MACHINE LEARNING AND CLASSIFICATION

Once we represent the data by feature descriptor, a classifier is needed to classify those data to labels. Some machine learning methods are suitable for this task. There are numbers of machine learning algorithms which can be divided into three main categories.

A. Supervised learning

Supervised learning is the method which can learn a specific function and perform a mapping task from an input to an output. The learning process is instructed by example input-output pairs, and the function's parameters will update automatically during training. [11]. Examples like classification and regression.

B. Unsupervised learning

Unlike supervised learning, the unsupervised learning doesn't need the priory knowledge which is labeled data, it can classify those data into different categories based on commonalities of data and then let human label those categories. Clustering is a classic method in unsupervised learning [12], the generative adversarial networks [13] in neural networks is also one popular example in this domain.

C. Reinforcement learning

Reinforcement learning is the machine learning task inspired by behaviourist psychology. There is no supervisor but only a reward signal. The reward is given by the environment which autonomous agents take actions to interact with. The foundation of reinforcement learning is reward hypothesis. It can be understood that all goals can be described by the maximisation of expected cumulative reward. The most wield known application of reinforcement learning is the AlphaGo [14]. It can also be used in robot manipulator control [15], autonomous cars etc.

In this experiment, the supervised learning algorithms are used as classifiers. Those methods include SVM [16], decision trees [17] and some models of neural networks [18].

IV. Experiment Methodology

The goal of this experiment is to improve the accuracy of prediction in robot failure execution. The datasets were obtained from UCI repository of ML [19][20]. The data set description is presented in Table 1 and Table 2.

Table 1

Dataset	Instances	Number of classes and distribution
LP1	88	4 (1 = 24%; 2=19%;3=18%;4=39%)
LP2	47	5 (1=43%; 2=13%; 3= 15%; 4=11%; 5= 19%)
LP3	47	4 (1 = 43%; 2 = 19%; 3 = 32%; 4=6%);
LP4	117	3 (1 = 21%; 2=62%;3 = 18%;)
LP5	164	5 (1 = 27%; 2=16%; 3 = 13%; 4=29%; 5=16%)

Table 2

Data Set	Type of failure
LP1	failures when approaching to grasp position
LP2	failures when transferring of a part
LP3	position of part after a transfer failure
LP4	failures when approaching to ungrasp position
LP5	failures in motion with part

The features' values were measured by force and torque sensors after failure detection. Those features are numerical values and each failure instance contains 90 features.

The experiment can be divided into three steps.

A. Data splitting and data cleansing

A set of five failure scenarios was designed based on statistical summary features [21]in order to improve classification accuracy. For the classification training, an input-output pair which features' data(vector form)-label pair was required, however, each class in data set has one string-type label and 90 features' values which were in matrix form. Thus, a 2-dimension data convert to 1-dimension data process was needed. In this experiment, the features' value in each row were connected to the previous row's end in order to converting the data matrix to vector. Source code is provided for this process.

B. Gaussian kernel convolution

To eliminate the influence of noise and disturbance, a gaussian kernel convolution [22] is introduced in this experiment. This process can be also called gaussian smoothing in image processing [23]. The gaussian distribution in 1D and 2D are presented as follows.

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{1}$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

In Equations 1 and 2, σ is the standard deviation of the distribution. The functions are illustrated in Figure 1 and Figure 2.

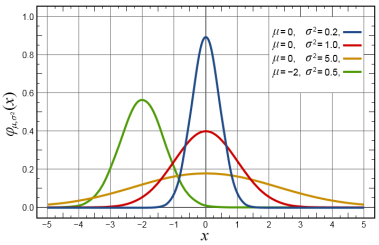

Fig. 1: 1D Gaussian distribution graph

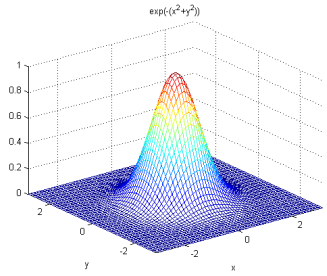

Fig. 2: 2D Gaussian distribution graph

In this experiment, a size $3 * 3$ convolution kernel is used and the σ value is set to 1. According to the equation 2,

the gaussian kernel used in this experiment is determined in table 3.

Table 3
Gaussian Convolution Kernel

0.0751136	0.123841	0.0751136
0.123841	0.20418	0.123841
0.0751136	0.123841	0.0751136

The source code is provided for this process[1].

C. Classification with supervised learning methods

Once we obtained processed data, we can proceed to classification. Classification learner in Matlab [24] is applied for this part to search for the best classification model type which include decision trees, discriminant analysis, support vector machines and some other methods. To prevent overfitting, 5-fold cross validation was used in this experiment.

Moreover, an artificial neural network also used in this experiment for classification. However, it requires changing the label names to numerical values. Because in the back propagation, an error is computed at the output and then distributed backwards throughout the network's layers to obtain a gradient that is used for calculating the weights in the network [25]. A specific source codes file about the back propagation was created and provided[2] to explain this process. The error is the difference between the output of the last hidden layer and the label, thus the labels must be numerical values in vector form.

V. Result and Discussion

The building time for the step 1 and 2 took only 0.3822s on the experimental device (A macbook air 2013 with very low computation resource). Because of the convolution process, the total amount of features decreased from 90 to 52, thus the classification took less time to train.

Experimental results on the performance of existing and purposed methods for handling robot failure execution's prediction are described as follows.

A. Existing method

To access the possible effectiveness, PCA is used to exclude insignificant features.

The best three models were selected according to the accuracy of prediction in each failure scenario as presented in Figure 3. The Bagged Trees and subspace KNN algorithms appeared in every top 3 models for all the robot failure scenario, which indicated the decision trees, nearest neighbors and their extensions are promising to solve this kind of problem.

B. Purposed method

On the other hand, the results of purposed method is presented in Figure 4. A similar trend that both results shows a obvious decrease in prediction's accuracy for data set LP5, the reason could be some features have certain dependency but lack relevance.

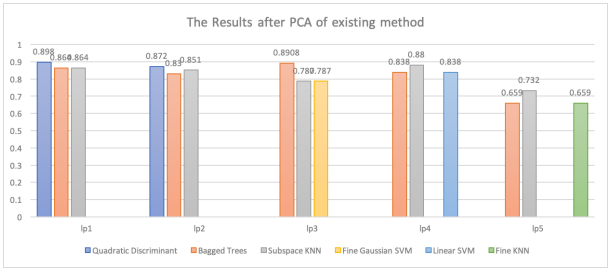

Fig. 3: The Results after PCA of existing method

The comparison of the results between purposed method and existing method is showned in Figure 5, except the LP5 data set, the purposed method outperformed all the rest scenarios than the existing method. Moreover, due to the reason analyzed below, the PCA process was removed to recalculate the prediction for LP5, the result in Figure 6 shows a advantage in accuracy of prediction by the purposed method.

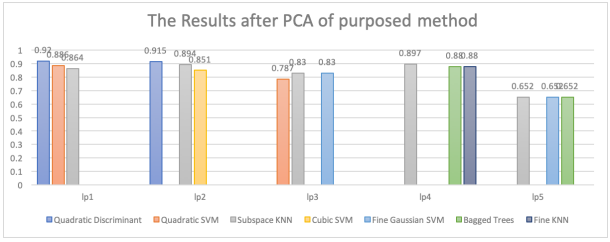

Fig. 4: The Results after PCA of purposed method

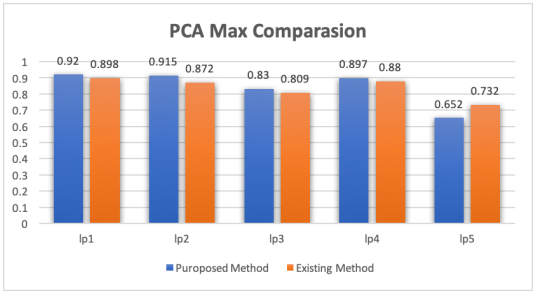

Fig. 5: The Max Successful Rate of Prediction

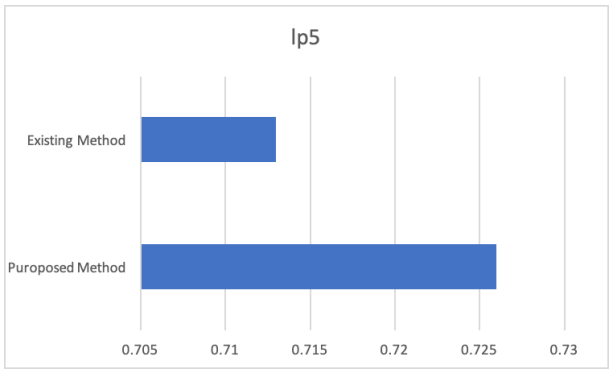

Fig. 6: The Results for LP5 without PCA

C. Result Comparison with Neural Networks

The data set LP5 was also tested with neural network method with a distribution of 70% in training, 15% in validation and 15% of testing to prevent overfitting.

The first experiment in this part run by adding 20 neurons in the hidden layer and its results are presented in Figures 7, 8.

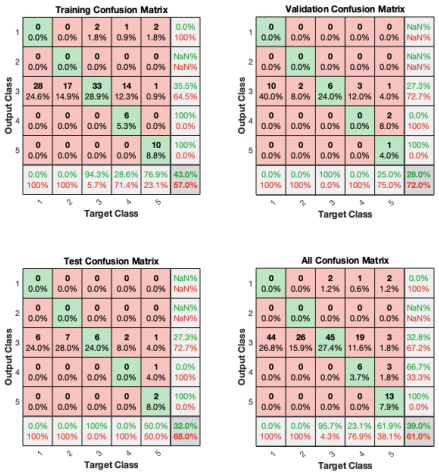

Fig. 7: The Confusion Results with 20 neurons in the hidden layer

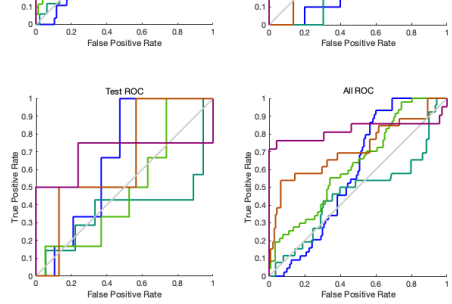

Fig. 8: The ROC Results with 20 neurons in the hidden layer

The results for the first test was not satisfying and the next step was adding more neurons (100 neurons) in the hidden layer. The structure of neural network is showned in Figure 9. The Results in Figures10,11 give a remarkble improvement, which also confirmed the feasibility of purposed method (72.6 %). However, if more neurons were added, a decrease trend in success rate could be observed due to the overfitting in the training process.

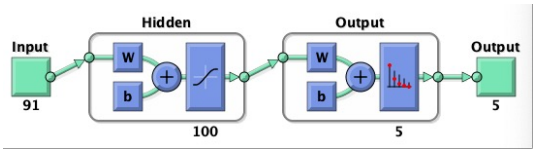

Fig. 9: Neural Network structure

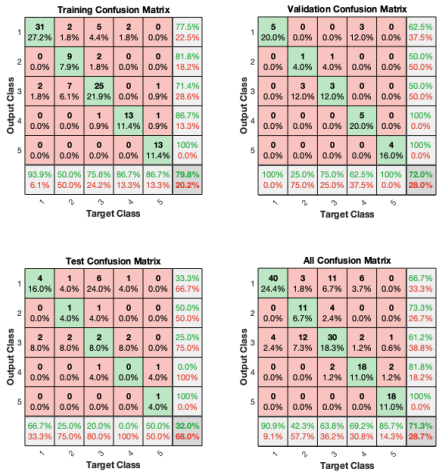

Fig. 10: The Confusion Results with 100 neurons in the hidden layer

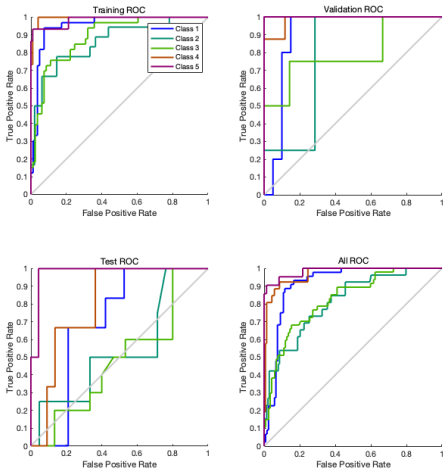

Fig. 11: The ROC Results with 100 neurons in the hidden layer

VI. CONCLUSIONS

In this experiment, several state-of-the art feature extraction and classification techniques, namely PCA, SVM, Decision Trees, kernel descriptors and neural networks has been implemented to address the problem of robot's failure detection. To represent the failure data by a 1-D vector, the data splitting and the concatenation of feature descriptors were used. A multi-class prediction was performed on test data by fitting different supervised learning methods with training and validation data. Some computation complexity reduction techniques are done prior to the classification training, parameters are tuned by cross validation. Moreover, by analyzing the robot execution background and add a convolutional gaussian processes, an improvement in the accuracy of prediction was observed and the hypothesis was confirmed.

REFERENCES

[1] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv e-prints*, page arXiv:1509.02971, September 2015.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. *arXiv e-prints*, page arXiv:1707.01495, July 2017.

[3] Iker Zamora, Nestor Gonzalez Lopez, Victor Mayoral Vilches, and Alejandro Hernandez Cordero. Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo. *arXiv e-prints*, page arXiv:1608.05742, August 2016.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] K. Tada, S. Kawamoto, N. Yamada, T. Kimura, M. Iwahashi, T. Tadaumi, and K. Kato. Sensor placement optimization in on-site photovoltaic module inspection robot for fast and robust failure detection. In *2015 IEEE 42nd Photovoltaic Specialist Conference (PVSC)*, pages 1–3, June 2015.

[6] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[7] Alberto Rodriguez, David Bourne, Margaret Mason, G.F. Rossano, and Jianjun Wang. Failure detection in assembly: Force signature analysis. pages 210 – 215, 09 2010.

[8] Bernhard Schlkopf, Alexander Smola, and Klaus-Robert Mller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[9] Ratnadeep Deshmukh, Niket Borade, and Pukhraj P. Shrishrimal. Independent component analysis and number of independent basis vectors. volume 58, 08 2015.

[10] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[11] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, NJ, third edition, 2010.

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1406.2661, June 2014.

[14] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016.

[15] Shixiang Gu*, Ethan Holly*, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proceedings 2017 IEEE International Conference on Robotics and Automation (ICRA)*, Piscataway, NJ, USA, May 2017. IEEE. *equal contribution.

[16] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-base learning methods*. Cambridge University Press, 2000.

[17] J. R. Quinlan. Induction of decision trees. *MACH. LEARN*, 1:81–106, 1986.

[18] Kevin Gurney. *An Introduction to Neural Networks*. Taylor & Francis, Inc., Bristol, PA, USA, 1997.

[19] Luis Seabra Lopes and Portugal Luis M. Camarinha-Matos Universidade Nova de Lisboa, Monte da Caparica. UCI machine learning repository, 1999.

[20] L. M. Camarinha-Matos, L. S. Lopes, and J. Barata. Integration and learning in supervision of flexible assembly systems. *IEEE Transactions on Robotics and Automation*, 12(2):202–219, April 1996.

[21] Luis Seabra Lopes and Luis M. Camarinha-Matos. *Feature Transformation Strategies for a Robot Learning Problem*, pages 375–391. Springer US, Boston, MA, 1998.

[22] Mark van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional gaussian processes. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2849–2858. Curran Associates, Inc., 2017.

[23] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J., 2008.

[24] Michael Paluszek and Stephanie Thomas. *MATLAB Machine Learning*. Apress, Berkely, CA, USA, 1st edition, 2016.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.