OPTIMAL EXPERIMENTAL DESIGN FOR TRAINING OF A FAULT DETECTION ALGORITHM

Shi Jin Lou, Thomas Duever[1], Hector Budman

Department of Chemical Engineering
University of Waterloo, Waterloo, Ontario, Canada, N2L 2G1

Abstract: This paper focuses on Optimal Experimental Design to train a Projection Pursuit Regression (PPR) model used for fault detection. A novel experimental design method, referred to as Gaussian Probability Design, is proposed and compared with the conventional Factorial Design. The Gaussian Probability Design automatically searches for the sparseness of the data, and adds pairs of training data on both sides of a class boundary in areas where the data density is the lowest. This design method outperforms the Factorial Design in reducing the fault misclassification more effectively with the same amount of new training data.

Keywords: Optimal Experimental Design, fault diagnosis, Projection Pursuit Regression

1. INTRODUCTION

In most model learning problems, the learner has the ability to act on its environment and gather specific experimental data that will minimize the model errors. A special case of a model learning problem is the training of a fault detection algorithm. In most of the literature dealing with fault detection, the learner has been treated as a passive recipient of data and its active role in determining optimal data for training has been ignored. More specifically, the fault detection algorithms were generally trained on specific faults on the assumption that only these faults will occurred in the future. Thus, research work on optimal experimental design (OED) for training of fault detection techniques has not been extensively considered and reported in the literature. This work addresses the generalization problem where future faults may occur that did not occur during training. Therefore, optimal experimental design may be a venue to minimize the misclassification of faults that were not observed during training of the fault detection algorithm.

In this work, the Projection Pursuit Regression (PPR) technique has been chosen as the basis for the design of the fault detection algorithm. PPR is a multivariate statistical technique [1,3,4], ideally suited for nonlinear systems and has been applied to fault detection [2]. Similar to other multivariate methods it is based on the decomposition of the inputs along principal components. However the basis functions referred to as hidden functions are not fixed a priori but determined by the training data and the output calculation is based on a nearest neighbourhood approach applied in the hidden functions space. In a previous work by Lou et al [5], PPR has been found to be a good trade-off as compared to other techniques from the point of view of extrapolation errors due to insufficient training data and noise rejection. This paper deals with a novel method to design optimal experiments for the training of a PPR-based fault detection algorithm. The objective is to design experimental data in some predetermined window of operating conditions to minimize the fault misclassifications during testing. The incentive is to minimize the number of experiments for the sake of economy. Three different sets of data will be considered: training set #1: obtained using the conventional Factorial Design to obtain a first model, training set #2: added based on the knowledge of the first data set, and a testing set, to assess the classification accuracy of the algorithm trained with the two training data sets. The current work presents a methodology to design the second training data set mentioned above. Thus, this study follows a sequential approach where the optimal design of training set #2 is based on a priori knowledge of training set #1.

Cohn (1996) has applied techniques from OED to guide the query selection of a neural network learner [6]. Cohn demonstrated that these techniques allow the learner to minimize the generalization error by minimizing its variance.

[1] Corresponding author. Tel: +1-519-888 4567 ext. 2540; Fax: +1-519-746-4979. E-mail: tduever@cape.uwaterloo.ca

His OED approach is claimed to be applicable to any network architecture whose output is differentiable with respect to its parameters and may be used for both regression and pattern classification problems. However, for PPR models, the derivatives of the output with respect to the parameters are actually the outputs of the hidden functions. Since for PPR models, each class corresponds to the same hidden output value in each hidden function, all data points belonging to the same class will have the same derivative values with respect to the parameters. Thus, Cohn's method will not be useful for distinguishing between better or worse experimental points belonging to the same class.

In this work a method, referred as to Gaussian Probability Design, is proposed for designing the second set of training data mentioned above. In order to test the efficiency of the proposed design, the design of training set #2 will be compared to a conventional factorial design of this set. Finally, the methods are illustrated and compared for a CSTR case study. This case study, previously used [7] for testing fault detection algorithms, considers faults as extreme values in the inlet temperature and concentration conditions. These conditions are inferred from the measurements of reaction temperature and concentration.

2. CONCEPTION OF EXPERIMENTAL DESIGN METHODS

In a pattern classification problem, Optimal Experimental Design depends on both the design problem and the property of the modelling tool, such as the generalisation/localization ability. Since PPR is based on a nearest neighbourhood idea, if the problem consists of a single straight class boundary, a symmetric pair of training data with one datum on each side of this boundary is sufficient to determine the boundary in the middle of the two points. For example, for the simple detection problem: $y = x + n$, where x is the input, y is the output(measurement), n is a noise term and class I:$x<0.5$,class II:$x>0.5$. In a simple Factorial Design, the designer can easily produce a pair of data symmetric with respect to the boundary, x+-Delta. A pair of data points symmetric to the boundary will be referred as a conjugate pair. On the other hand when the class boundary is curvilinear, many conjugate pairs of training data may be needed, in order to identify the true boundary. An example for this situation is shown in Figure 1 where a curvilinear boundary, i.e. composed of different straight portions, separates two fault classes to be identified. The output of class I given by the rhomboids is equal to 1 whereas the output corresponding to class II given by the triangles is equal to 2. Thus, the problem consists of inferring the class from the output values. At least 8 training data points are needed, as shown in Figure 2, in order to accurately determine the class boundary between the two classes using a PPR algorithm.

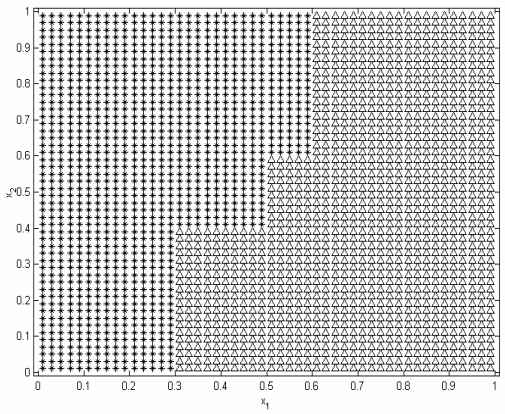


Fig. 1. True classification of a data pattern

To illustrate this point, let us assume in Figure 2, an initial training set #1 is given by points *a* and *b*. Then training set #2 will be selected from either one of the conjugate pairs defined by coordinates (*c, d*), or (*e, f*), or (*g, h*) to test which pair result in the best model, in terms of the smallest class misclassification. The test set consists of all the grid points shown in Figure 1. If *c* and *d* are used as new training data, the PPR method produces a pattern classification as presented in Figure 3. The misclassification rate is as high as 26%. If *e* and *f* are used as the new training data, the corresponding data classification is shown in Figure 4. The misclassification rate is then reduced to only 4.5%. Finally, if the conjugate pair *g* and *h* are applied as the new training data, the misclassification rate will be further reduced to 3.9%.

The above example shows that the design of training data set #2 will result in the best model, if the new data is added to the area with the lowest data density. A major problem with Factorial Design is that it does not automatically search the sparseness of training data in the data domain. Therefore, an algorithm, which can automatically search the data domain and exploit the sparseness of training data, is expected to give better results with the same amount of data. This observation inspired a novel experimental

design method for fault detection, which will be introduced in the Section 4.2.

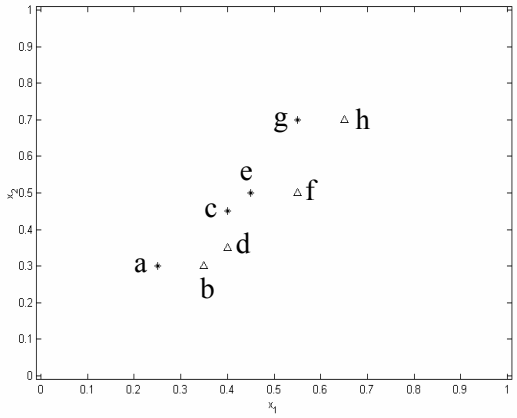


Fig. 2 Minimum training data needed to identify class boundary accurately

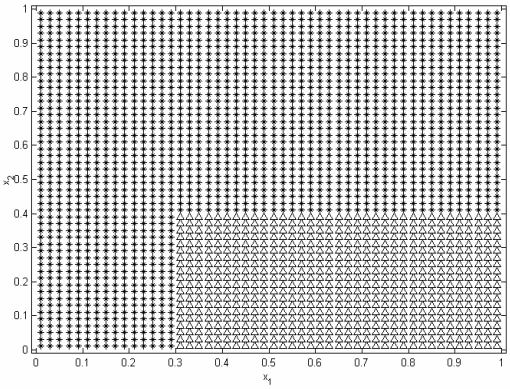


Fig. 3. Classification results with new training data *c* and *d*

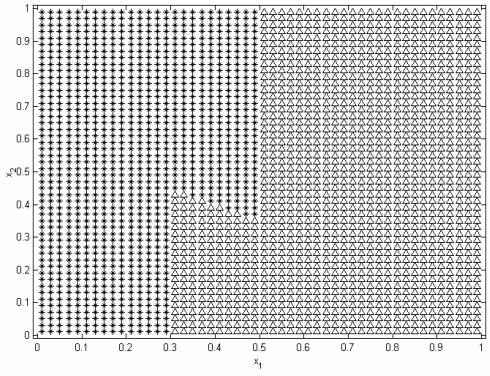


Fig. 4 Classification results with new training data *e* and *f*

3. TRAINING DATA SET #1

The CSTR model used in this case study has been originally used by Venkatasubramanian to study a fault detection algorithm based on a Radial Basis Functions Neural Network [7] and it is based on conventional component and energy balances. Two outputs are measured: the outlet temperature and the outlet concentration. These variables are then used to identify the faults in two process inputs: the inlet temperature and the inlet concentration. The magnitude of the inlet temperature, T_o, is allowed to change from the normal steady state value to 3.0 times of that value. The magnitude of another process input, the inlet concentration C_{Ao}, is allowed to change from the normal steady state value to 1.6 times of that value.

As mentioned before, the experimental design methods discussed in this case study are based on an initial set of training data which is generated by a Factorial Design. The design is carried out at 3 levels: 0, 5, 10. There are two factors: the inlet temperature, and the inlet concentration. For a full Factorial Design with 2 factors and 3 factorial levels, there are 9 possible experimental conditions, as shown in Table 1.

Table 1. Factorial Design for training data set #1

Experiment run	Factorial level 1	Factorial level 2
1	0	0
2	0	5
3	0	10
4	5	0
5	5	5
6	5	10
7	10	0
8	10	5
9	10	10

The magnitude of inlet temperature, with respect to its normal steady state value, is decided according to Equation (1):

$$Magnitude\ 1 = factorial\ level\ 1 \times 0.2 + 1.0 \quad (1)$$

The magnitude of inlet concentration, with respect to its normal steady state value, is decided by Equation (2):

$$Magnitude\ 2 = factorial\ level\ 2 \times 0.06 + 1.0 \quad (2)$$

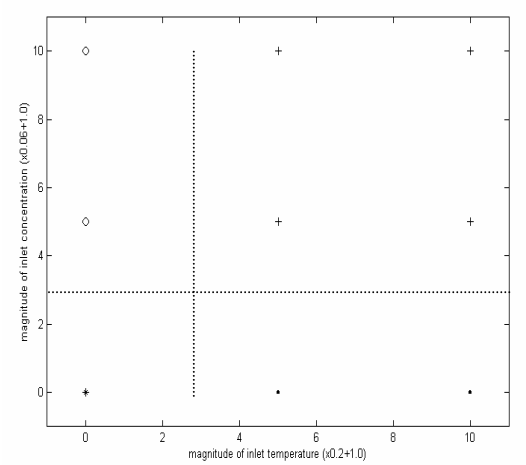
Based on these definitions the faults are defined as shown in Table 2. The generated training data is plotted in the process input space in Figure 5.

Corresponding to the data pattern in the process inputs in Figure 5, the process outputs, i.e., the reactor temperature and concentration, are generated using the CSTR model, to train a PPR model. In order to test the performance of the PPR model, a group of testing data has been designed. The testing data are also generated by a Factorial Design on 2 factors: the inlet temperature and the inlet concentration, but at 20 factorial levels: 0.25, 0.75, 1.25,…, 8.75, 9.25, 9.75; i.e., from 0.25 to 9.75, with an interval of

Table 2. Definitions of Faults for the CSTR example

Normal operation (Class 1):	T_o < 1.6 times its normal steady state value; C_{Ao} < 1.18 times its normal steady state value;
high inlet concentration fault (Class 2):	T_o < 1.6 times its normal steady state value; C_{Ao} ≥ 1.18 times its normal steady state value;
High inlet temperature fault (Class 3):	T_o ≥ 1.6 times its normal steady state value; C_{Ao} < 1.18 times its normal steady state value;
Concurrent faults (Class 4):	T_o ≥ 1.6 times its normal steady state value; C_{Ao} ≥ 1.18 times its normal steady state value;

0.5. A full Factorial Design on two factors at 20 factorial levels results in 400 experimental conditions. The percentage of misclassification is calculated to be 38.5%. The misclassifications are mainly due to the insufficiency of training data.



✱Class 1; ○Class 2; •Class 3; +Class 4; — true class boundary

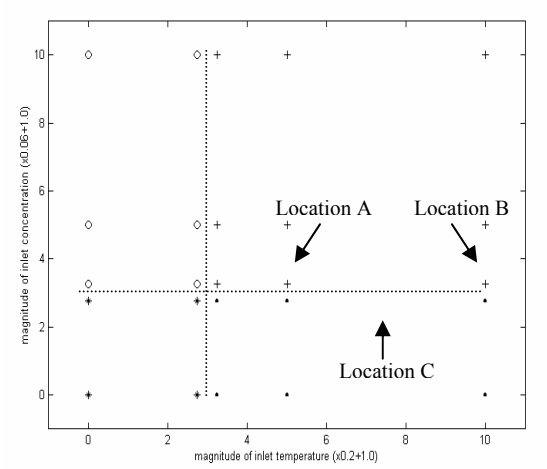
Fig. 5. Original training data in process input space,

4. COMPARISON OF EXPERIMENTAL DESIGN METHODS FOR TRAINING DATA SET #2

4.1 Factorial Design

To generate the new training data #2 using Factorial Design, two factorial levels are added on each of the two factors. Now each factor has a total of five factorial levels: 0, 2.75, 3.25, 5, 10. After the addition of the new data, the training data in the process input space has a pattern as shown in Figure 6. Compared to the original training data, 16 new data have been added.

The PPR model trained with these new data results in 23.25% misclassification, when tested on the testing data set defined above. Checking the locations of the misclassification, it is found that the new training data generated by the Factorial Design does not reduce the misclassification in location *C* in Figure 6. The figure shows that the Factorial Design adds new training data to location *A* and *B*, because sufficient data already exist there. Location *C* has a much lower density of training data than location *A* and *B*. But the Factorial Design does not add any new training data in location *C*, because it does not search for the sparseness of the training data. A combed effect of the sparseness of training data and the curvilinear shape of the class boundary in the process output space results in high misclassification in location *C*.



✱Class 1; ○Class 2; •Class 3; +Class 4; — true class boundary

Fig. 6. Training data in the process input space, after Factorial Design

4.2 Gaussian Probability Design

As for the Factorial Design, the Gaussian Probability Design is also carried out, based on the original training data in the process input space, which is shown in Figure 5. The design is implemented in the following steps.

Step 1: The Gaussian probability is calculated on a set of experimental data, with respect to their process inputs. The experimental data to be

considered are the grid points next to the class boundary. The sampling rates in all dimensions of this grid are decided based on a trade-off between the design accuracy and the computational expense. The Gaussian probability of a datum is calculated by the following equation.

$$p_j = \sum_{i=1}^{N_T} e^{-(\frac{x_{j,1}-x_{i,1}}{s_1})^2-(\frac{x_{j,2}-x_{i,2}}{s_2})^2\cdots-(\frac{x_{j,d}-x_{i,d}}{s_d})^2} \quad (3)$$

Where, p_j represents the Gaussian probability of the j^{th} experimental datum, with respect to all the training data in data set #1; N_T is the total number of the original training data in data set #1. For the i^{th} training datum $X_i = \begin{bmatrix} x_{i,1} & x_{i,2} \cdots x_{i,d} \end{bmatrix}^T$, $x_{i,d}$ represents the process input of the i^{th} training datum, in the d^{th} dimension; s_d is the sampling rate of the process input, in the d^{th} dimension.

Step 2: An experimental datum is accepted as a new training datum, if it has the lowest probability value, p_{min}.

$$p_{\min} = \min_{1\le j\le N_E} \{ p_j \} \quad (4)$$

Where, N_E is the total number of experimental data.

Step 3: Once a new training datum is selected, its conjugate point on the other side of a class boundary is also selected as a new training datum. A conjugate point can further generate a new one with respect to another individual class boundary. This procedure is carried on, until each new training datum has a conjugate point on the other side of a class boundary, to form a conjugate pair. Conjugate pairs of training data, across the class boundary, are sought, because they obviously provide equal amount of class information, on the two sides of the boundary. Unequal amount of training data on the two sides of the class boundary will cause the PPR model to identify a class boundary that is biased towards the side with less training data.

This proposed design method can be carried out in a sequential approach manner. If the design objective, such as the misclassification rate in testing, is not met in the previous run of design, additional training sets can be added according to steps 1-3 above.

For the CSTR example the experimental data to be considered in step #1 are shown in Figure 7. The Gaussian probability is calculated on each experimental datum, according to Equation (3). New training data are then selected from the experimental data following Equation (4). The following three data points have been found to locate in the areas with the (equally) lowest density: [2.75 2.75], [7.75 2.75], [2.75 7.75]. The conjugate points of these new training data are also selected, according to Step 3 above. For example, the data point '*a*' in Figure 8 is a new training datum selected in the previous design steps. The data point '*b*' is accepted as a new training datum, because it is the conjugate point of '*a*', with respect to class boundary *1*. The data point '*d*' is another conjugate point of '*a*', with respect to class boundary *4*. Since data point '*c*' is a conjugate point of both '*b*' and '*d*', with respect to the class boundary *2* and *3* respectively, it is also taken as a new training datum, following Step 3. Finally, the new training data, together with the original ones, are presented in Figure 8.

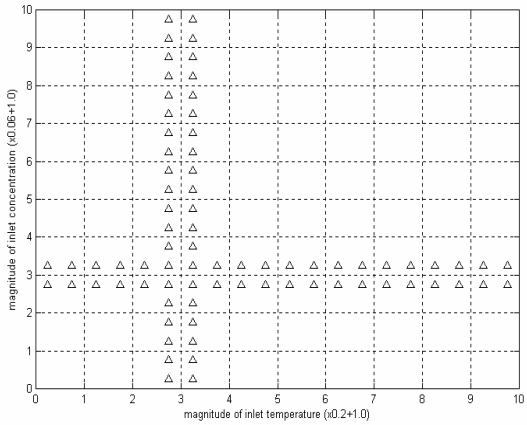


Fig. 7. Experimental data in Gaussian Probability Design,

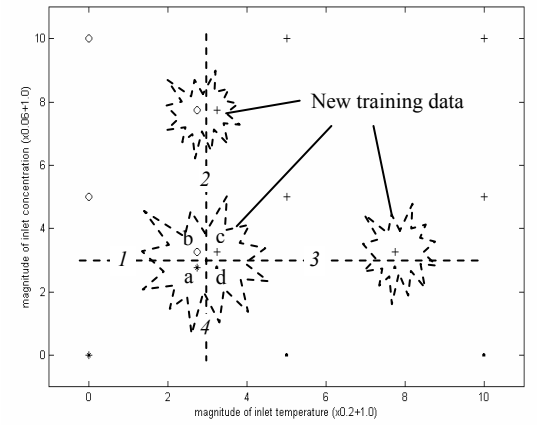


Fig. 8. New training data, together with original ones, in process input space,

Based on the training data set in Figure 8, a PPR model is obtained that gives only 16.25% misclassification in the testing. This result is better than 23.25% misclassification by the Factorial Design. Furthermore, the Gaussian

Probability Design achieves this higher accuracy with only 8 new training data, significantly less than 16 new training data calculated by the Factorial Design. In summary, the Factorial Design adds training data to locations *A* and *B*, which are already dense with training data, but leave location *C* blank. On the other hand, the Gaussian Probability Design automatically searches for the data void, and adds new training data to location *C*, where the data density is the lowest. The difference in the location of new training data shows the advantage of the Gaussian Probability Design, in automatically searching for data voids.

5. CONCLUSION

A novel experimental-design method, referred to as Gaussian Probability Design, is proposed and compared to the conventional Factorial Design. The comparison is carried out on a CSTR process. The simulation results are summarised in Table 3. The results show that, for PPR, the Gaussian Probability Design provides a more accurate classification of faults as compared to the Factorial Design.

Table 3. Comparison results of experimental design methods

	Misclassification
Original training data	*38.50%*
Factorial Design *(16 new data)*	*23.25%*
Gaussian Probability Design *(8 new data)*	*16.25%*

The key disadvantage of Factorial Design for training a fault detection algorithm is that it can not take into account the sparseness of the data. The location of new training data is decided by a combination of the factorial levels. The design adds training data not only on the class boundary, but also inside a class as well. This is often unnecessary for a model, which can make reasonable generalisation, such as PPR. For these models, a training datum near the class boundary is more important than one inside the class. The classification of a datum inside a class can be correctly determined by the PPR technique through interpolation, based on the data located near the class boundary. When the class boundary is of complicated shape, or the dimension of the problem is high, i.e. it has a large number of inputs and outputs, Factorial Design may lead to a prohibitively large amount of training data.

Our simulation results show that using more training data does not necessarily mean less misclassification. This further illustrates that a ***random design*** of training data may lead to poor generalization results.

The Gaussian Probability Design investigates the sparseness of the training data, and adds training data to the class boundary area, where the data density is the lowest. Since low density indicates insufficiency of training data, this method is *efficient* in filling out the data void. The rationale for using the Gaussian probability distribution based on assessing the probability of points in the neighbourhood of the boundary to belong to a specific class based on an initial set of training data. Finally, the Gaussian Probability design results in smaller number of experiments as compared to Factorial Design, since it adds points only in the neighbourhood of the class boundary. It does not add unnecessary data points away from the class boundary that can be easily interpolated by the PPR algorithm based on the points located at the boundary.

REFERENCE

[1] Friedman, J.H. and W. Stuetzle, (1981) Projection Pursuit Regression, *Journal of the American Statistical Association*, no.376, p:817

[2] Flick, T.e., L.K. Jones, R.G. Pries, and C. Herman, (1990) Pattern Classification using Projection Pursuit, *Pattern Recognition*, Vol. 23, No. 12, pp. 1367-1376

[3] Utojo, U., B. R. Bakshi, (1995) Connections between Neural Networks and multivariate Statistical Methods: An Overview, *Neural Networks in Bioprocessing and Chemical Engineering*, Academic Press

[4] Hwang, J.N., S. R. Lay, M. Maechler, R. D. Martin, J. Schimert, (1994) Regression Modeling in Back-propagation and Projection Pursuit Learning, *IEEE Transactions on Neural Network*, vol.5, no.3

[5] Lou, S. J., H. Budman, and T. A. Duever, (accepted in August 2002), Comparison of fault detection techniques: Haar Wave-Net versus Projection Pursuit Regression, *Journal of Process Control*

[6] Cohn, David A., (1996) Neural network exploration using optimal experimental design, *Neural Networks*, vol.9, no.6, pp1071-1083

[7] Baughman, D. R.; Liu, Y. A, (1995) *Neural Networks in Bioprocessing and Chemical Engineering*, Academic Press