Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre de 2018
https://doi.org/10.17979/spudc.9788497497565.0554

Detección de fallos dinámica y descentralizada basada en métodos de regresión

A. Sánchez-Fernández, M.J. Fuente y G.I. Sainz-Palmero
Departamento de Ingeniería de Sistemas y Automática.
Universidad de Valladolid,
{alvsan, mjfuente, gresai}@eii.uva.es)

Resumen

Este artículo propone un método de detección de fallos dinámico y descentralizado. Para hacer la detección de los fallos descentralizada, la planta se divide en bloques de variables que compartan algún tipo de correlación usando métodos de regresión. En cada grupo se incorpora un método de detección de fallos dinámico, en concreto el método DPCA: Análisis de componentes principales dinámico, cuyos resultados son fusionados por un procesador central, utilizando el Criterio de Inferencia Bayesiano (BIC), devolviendo un resultado global. Esta propuesta ha sido aplicada sobre un modelo de planta industrial ampliamente utilizado y comparado con el DPCA centralizado para verificar su efectividad.

Palabras clave: Detección de fallos, Análisis de componentes principales dinámico (DPCA), Descentralización, LASSO, Random Forest.

1 INTRODUCCIÓN

El objetivo del control de procesos incluye hacer que el sistema funcione en las condiciones deseadas, de forma estable y segura, y detectar fallos, esto es, cualquier desviación del comportamiento esperado. Estas desviaciones deben ser detectadas rápidamente, ya que cualquier malfuncionamiento de la instalación puede implicar graves pérdidas de rendimiento, de calidad del producto, paradas no programadas, etc. Además, la aparición de fallos puede llevar a que la planta trabaje en condiciones peligrosas para los empleados, otras instalaciones, etc., por lo que es vital que esta tarea se haga de manera efectiva.

Los métodos de monitorización de procesos basados en datos como el Análisis de Componentes Principales (PCA) [1], Mínimos Cuadrados Parciales (PLS) [2], o el Análisis de Variación Canónica (CVA) [3, 4] han sido ampliamente utilizados en los últimos tiempos; esto se debe a que son capaces de extraer información útil sobre el proceso a partir de datos medidos en la planta y detectar fallos; sin necesidad de poseer ningún conocimiento de la planta aparte de los datos. Por otro lado, muchos de los métodos estadísticos multivariantes, como el PCA, asumen que no existe correlación temporal en los datos, es decir, que el estado actual de la planta no está influido por los estados pasados. Esta situación no se da prácticamente nunca en las plantas industriales, por lo que, para tener en cuenta esta correlación de tiempo y capturar la dinámica del proceso, se han propuesto diversos métodos como el PCA dinámico (DPCA) [5], que aplica el método PCA sobre una matriz de datos aumentada que incluye datos retardados, también [6, 7] han trabajado con CVA, que es un método que incluye datos pasados y futuros para modelar el proceso.

El incremento de la automatización lleva aparejado una ingente cantidad de datos recogidos, por tanto, la capacidad computacional necesaria para procesar los datos se ha incrementado considerablemente. Esto hace que la detección de fallos consuma gran cantidad de tiempo y recursos. En plantas pequeñas es asumible usar un procesador central que recopile todos los datos y realice la detección de fallos. En el caso de plantas grandes y complejas no siempre este procesador central es capaz de realizar esas funciones. Para abordar este problema se puede dividir la planta en bloques de variables [8] e incluir en cada uno de esos bloques un procesador que se ocupe de la detección de fallos sólo en ese bloque. Posteriormente, un procesador central recopila los resultados de la detección de cada bloque y los fusiona para dar una identificación global. Esto es conocido como detección descentralizada.

La detección descentralizada ha sido aplicada por muchos autores [8, 9]. El paso crucial de este método es la división de la planta en bloques de variables, que puede hacerse incluyendo sólo una variable por bloque [10], o varias de ellas por bloque. En este segundo caso, se puede dividir la planta en base a su topología, creando bloques que incluyan los datos de sensores próximos entre sí [11, 12], o, se pueden analizar los datos disponibles para extraer información que permita saber qué variables tienen algún tipo de correlación para agruparlas [8, 13, 9, 14].

La fusión de los resultados locales se puede abordar de distintas formas: [8] propuso el criterio de entropía máxima, también existe la opción de aplicar métodos de toma de decisiones multicriterio como son los operadores Ordered Weighted Average (OWA) propuestos por [15], también en las referencias [13, 9] se usó el Criterio de Inferencia Bayesiano (BIC) aplicado a un método de detección de fallos.

En este trabajo se propone un método de detección de fallos descentralizado y dinámico. Primero se divide la planta en bloques usando métodos de regresión que busquen relaciones entre variables, como son la regresión basada en el Operador de Contracción y Selección Mínima Absoluta (LASSO) [16], y el método Bosques aleatorios (Random Forest) [17]. Esto llevará a que cada variable estará incluida en un grupo con otras variables con las que comparta cierta correlación. Los procesadores locales llevarán a cabo la detección de fallos aplicando DPCA. Después, el procesador central fusionará todos los resultados locales aplicando el Criterio BIC dando lugar a un resultado de detección global.

Este artículo está estructurado de la siguiente forma: la Sección 2 explica la teoría aplicada en este trabajo: métodos descentralizados, LASSO, Random Forests, PCA dinámico, y el criterio BIC de fusión de resultados. En la sección 3 se explica el método de detección descentralizado aplicado en este trabajo. Las pruebas que se han hecho usando la planta Tennessee Eastman Plant (TEP) y los resultados obtenidos se incluyen en la Sección 4. Para finalizar, las conclusiones y las líneas de trabajo futuro se muestran en la Sección 5.

2 PRELIMINARES

2.1 Detección de fallos descentralizada

Al trabajar de forma descentralizada, la planta se divide en bloques de variables y en cada uno de ellos se hace una detección local. Luego, todos los resultados de los bloques se fusionan para obtener un resultado para toda la planta. El paso de dividir la planta en bloques es crucial para el rendimiento posterior del método y puede hacerse de varias formas:

- Descentralización completa [8]: cada variable posee su propio grupo, en el que se incluye dicha variable, y, quizá, datos retardados de esa variable. La información relativa a la correlación entre variables no se tiene en cuenta.
- Descentralización multi-bloque:
  - Un bloque por variable: cada variable tiene un bloque que incluye, aparte de dicha variable, aquellas que tengan algún tipo de correlación con ella. Pueden incluirse aquí datos con retardo de algunas variables.
  - Menos bloques que variables: una vez encontradas las correlaciones entre variables se construyen bloques formados por las variables (y, en su caso, variables con retardo) que tengan algún tipo de correlación. Si algunas variables no están incluidas en ningún bloque, pueden ser descartadas o incluidas en un bloque extra.

Una vez la planta se ha dividido en bloques, se debe implementar un método de detección de fallos en cada bloque, en en este trabajo se propone aplicar el PCA dinámico.

2.2 PCA dinámico

Estando la planta en condiciones normales se obtiene una matriz $\mathbf{X}$ de n observaciones/filas y m variables/columnas. En el caso de querer hacer el método dinámico, se construye una matriz ampliada $\mathbf{X_a}$ incorporando a $\mathbf{X}$ datos con retardo:

$$\mathbf{X_a} = \begin{bmatrix} X_{l+1}^T & X_l^T & \dots & X_1^T \\ X_{l+2}^T & X_{l+1}^T & \dots & X_2^T \\ \vdots & \vdots & \ddots & \vdots \\ X_n^T & X_{n-1}^T & \dots & X_{n-l}^T \end{bmatrix} \quad (1)$$

siendo X_t el vector de medidas tomadas en el instante t, y l el número de retardos que se quieren incluir. El valor óptimo para l se puede determinar mediante el Criterio de Información de Akaike (AIC) según lo indicado en [18, 6]. Aplicando la descomposición SVD a la matriz de covarianza de $\mathbf{X_a}$ se obtiene:

$$\mathbf{S} = \frac{1}{(n-1)}\mathbf{X}_a^T\mathbf{X}_a = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \quad (2)$$

Los valores singulares de $\mathbf{S}$ están en la diagonal de $\mathbf{\Lambda}$, ordenados de forma decreciente y representan la varianza de los datos incluidos en cada componente principal. Las columnas de la matriz $\mathbf{V}$ son los vectores singulares de $\mathbf{S}$.

Detección de fallos La detección de fallos se suele realizar aplicando los estadísticos T^2 y Q [1].

El estadístico T^2 para un nuevo vector de observaciones ampliado $\mathbf{x}$ se obtiene con la fórmula:

$$T^2 = \mathbf{x}^T\mathbf{P}\mathbf{\Lambda_a}\mathbf{P}^T\mathbf{x} \quad (3)$$

donde $\mathbf{\Lambda_a}$ incluye las primeras a filas y columnas de $\mathbf{\Lambda}$ y $\mathbf{P}$ es la matriz de cargas, formada por las a primeras columnas de $\mathbf{V}$. El valor de a es el menor valor posible que recoja la mayor cantidad de información de los datos en el modelo PCA (usando la diagonal de $\mathbf{\Lambda}$). El estadístico detecta un fallo si supera su umbral T^2_α

El estadístico Q (o error de predicción cuadrático) mide la bondad de ajuste del PCA y el ruido de las medidas y se calcula así:

$$Q = [(\mathbf{I} - \mathbf{PP}^T)\mathbf{x}]^T[(\mathbf{I} - \mathbf{PP}^T)\mathbf{x}] \quad (4)$$

donde $\mathbf{I}$ es la matriz identidad. Se detecta un fallo si Q supera su umbral Q_α. Los umbrales de T^2 y Q se calculan según los indicado por [19].

Para evitar falsas alarmas, se impone que un fallo sea detectado si alguno de los dos estadísticos supere su umbral durante un cierto numero de observaciones consecutivas. Dicho número debe ser lo suficientemente alto para evitar falsas alarmas, pero no tan alto que reduzca la capacidad de detectar fallos leves.

2.3 Regresión LASSO

La regresión LASSO (Lest Absolute Shrinkage and Selection Operator) fue introducida en [16] como una mejora del modelo de regresión lineal. Se basa en aplicar una penalización a los coeficientes de la regresión de tal forma que algunos de ellos se hagan cero. Esto consigue un modelo más sencillo y facilita su interpretabilidad. También se puede usar para hacer selección de variables ya que sólo las variables con mayor influencia en la regresión tendrán coeficientes distintos de cero.

Un modelo lineal con m predictores: $X(t) = (x_1(t), x_2(t), \ldots, x_m(t))$ y una variable respuesta $y(t)$ se define con la siguiente ecuación:

$$y(t) = \beta_0 + \beta_1 x_1(t) + \ldots + \beta_m x_m(t) + \varepsilon(t) \quad (5)$$

donde β_i $(i = 1, ..., m)$ son los coeficientes de la regresión y $\varepsilon(t)$ es el error del modelo en el instante t. Si se usa notación matricial:

$$Y = \mathbf{X}\beta + E \quad (6)$$

donde Y es el vector respuesta para todos los instantes, X son las medidas, β es el vector de los coeficientes de regresión para cada variable y, finalmente, E el vector del error predicción.

El método LASSO trata de resolver el problema:

$$\hat{\beta}(\lambda) = \underset{\beta}{\operatorname{argmin}} \left(\frac{\| y - \mathbf{X}\beta \|_2^2}{n} + \lambda \| \beta \|_1 \right) \quad (7)$$

siendo $\lambda \geq 0$. El valor de λ marca el resultado del método, ya que si se incrementa, más coeficientes se harán cero.

2.4 Random Forests

Random Forest es un método basado en los árboles de decisión muy utilizado gracias a su sencillez y robustez frente a cambios en los parámetros [17]. Aunque se suele usar para realizar clasificaciones [20] también es posible aplicarlos para regresión [21, 22]

El método implementa varios árboles de decisión independientes entre sí, creados a partir de conjuntos de datos aleatorios con igual distribución. Si se parte de una matriz de datos $\mathbf{X}$ $(n \times m)$ con n observaciones y m variables, el proceso seguido consiste en ir creando las ramas de un árbol utilizando una muestra aleatoria de p predictores, con $p < m$, escogidos aleatoriamente entre las m variables. Al crear la siguiente rama se tomará otro subconjunto de p predictores seleccionado también de forma aleatoria. El valor p se recomienda, en el caso de hacer una regresión, que sea $p = m/3$, y en el caso de clasificación $p \approx \sqrt{m}$. Esta metodología evita que, cuando se generen varios árboles distintos, si existen varios predictores que tienen mucha influencia, esos árboles estén correlados entre ellos [23, 17].

Se crean varios árboles siguiendo éste método, de modo que cuando se quiera obtener una predicción para una nueva observación, se obtendrá calculando un promedio de las predicciones, para dicha observación, de cada uno de los árboles generados. De cara a aplicar esta técnica para la selección de características se puede medir la importancia de cada variable en el modelo Random Forest creado. Para ello, durante la fase de entrenamiento se va midiendo el error *out-of-bag* [24] de cada observación y haciendo un promedio para todos los árboles. Para medir la importancia de una determinada variable, los valores de dicha variable se permutan para los datos de entrenamiento y se observa como cambia el error *out-of-bag* para estos datos de entrenamiento modificados. De este modo, la importancia de dicha variable es el promedio de la variación de dicho error entre el conjunto de datos original y el modificado. Este proceso se detalla en [17].

2.5 Criterio de Inferencia Bayesiano

Cuando se dispone de resultados en diferentes bloques de un sistema que se quieren fusionar se puede aplicar el criterio BIC (Bayesian Inference Criteria). En el caso de una detección de fallos descentralizada cada bloque dará, en cada instante, un valor para cada estadístico: T^2 y Q. La forma de fusionar cada uno de ellos usando BIC [13, 25] es la siguiente. Para cada bloque i se calcula:

Para el estadístico T^2:

$$P_{(F|x_i)} = P_{(x_i|F)}P_{(F)}/P_{(x_i)} \tag{8}$$

con

$$P_{(x_i)} = P_{(x_i|N)}P_{(N)} + P_{(x_i|F)}P_{(F)} \tag{9}$$

donde N es el estado normal de la planta, y F el estado con fallo. $P_{(N)}$ y $P_{(F)}$ son las probabilidades a priori de que el sistema esté trabajando sin fallo y con fallo, respectivamente. Se toma un valor de α para la primera y de $1-\alpha$ para la segunda. Por otro lado:

$$P_{(x_i|N)} = e^{-T_i^2/T_{i,lim}^2}, P_{(x_i|F)} = e^{-T_{i,lim}^2/T_i^2} \tag{10}$$

siendo $T_{i,lim}^2$ el umbral de T^2 en el bloque i. Finalmente, se fusionan los resultados de cada bloque según la expresión:

$$BIC_{T^2} = \sum_{i=1}^{m}\{\frac{P_{(x_i|F)}P_{(F|x_i)}}{\sum_{i=1}^{m} P_{(x_i|F)}}\} \tag{11}$$

El valor de BIC_Q se calcula de forma similar, sustituyendo T^2 por Q. El criterio para identificar un fallo es que alguno de los BIC supere el valor umbral de $(1-\alpha)$. Valores comunes para α son: 0'8, 0'9, 0'99, etc.

Se usará un determinado valor de α en los datos de entreno y luego se reajustará en los datos de test (sin fallo, en ambos casos) de modo que no aparezcan falsas alarmas (que no haya un determinado número de observaciones anómalas consecutivas).

3 MÉTODO DE DETECCIÓN DE FALLOS DISTRIBUIDO

El método propuesto aquí consiste en aplicar métodos de selección de variables para dividir la planta en bloques y posteriormente implementar un método de detección de fallos dinámico en cada bloque. Los resultados obtenidos en cada bloque serán fusionados para generar un resultado de detección para toda la planta. Esta metodología se detalla a continuación:

Paso 1.- División de la planta: se proponen dos métodos basados en datos para hacer esta división: regresión LASSO y Random Forest.

Regresión LASSO Se parte de una matriz de datos $\mathbf{X}$ y se ajustan varias regresiones LASSO, en cada una de ellas la salida va a ser cada una de las variables $\mathbf{Y} = x_i,\ i = 1, ..., m$ y los predictores serán las columnas de la matriz $\mathbf{X}$ de la que se ha eliminado previamente la columna i-ésima que es la que va a ser modelada. Para cada variable a modelar se harán pruebas con diferentes valores de coeficientes de regresión no nulos y se seleccionará aquel modelo que mejor ajuste los datos. Para ello se calculará la raíz del error cuadrático medio (rMSE) y aquel modelo con menor error será el elegido. Posteriormente, para cada variable i se creará un bloque en el que se incluirá dicha variable y aquellas variables que en el modelo LASSO de la variable i hayan obtenido coeficientes de regresión no nulos. Esto da lugar a una descentralización de la planta con tantos bloques como variables.

Random Forest El proceso es similar al de la regresión LASSO y también se va a crear aquí un bloque por cada variable. La diferencia es que el modelado de cada variable se hará con Random Forest. Para cada variable, una vez obtenido su modelo, se crea un grupo en el que dicha variable estará incluida acompañada de todas aquellas variables cuya importancia en el modelo sea alta. Cuántas variables incluir en cada grupo es algo que debe ser determinado por el usuario en base a pruebas con los datos de entrenamiento con distintos valores, que darán lugar a diferentes configuraciones en la descentralización y, por tanto, a diferentes rendimientos del método de detección de fallos. Finalmente, se elegirá el valor que consiga una detección de fallos más efectiva (más fallos detectados, menor tiempo de detección, menor número de falsas alarmas).

Paso 2.- Entrenamiento del método dinámico de detección de fallos (DDPCA): cada uno de los bloques dispone de datos de sus sensores tomados en condiciones de funcionamiento normales. Para entrenar el método se requiere definir el número de retardos l que se va a usar. Esto puede ser ajustado por el usuario haciendo pruebas con diferentes valores, eligiendo aquel que de mejor resultado en la detección de fallos sobre datos de entrenamiento con fallo. Una vez entrenados los modelos DPCA locales, se obtendrán los umbrales locales teóricos para T^2 y Q. Estos valores se ajus-

tarán con datos de test sin fallo de modo que no haya mas de un 1% de observaciones anómalas.

Hasta aquí se ha trabajo fuera de línea.

Paso 3.- PCA dinámico y descentralizado (DDPCA). Este paso se hace en línea, con la planta en funcionamiento. Cuando se reciba una nueva observación los modelos DPCA locales calcularán unos valores de los estadísticos T^2 y Q, con lo que se tendrán m pares de estadísticos para este instante. Un procesador central recibirá los valores locales de los estadísticos y sus umbrales y los fusionará usando el criterio BIC explicado en el apartado 2.5. De esta forma se obtendrá un valor de BIC para cada estadístico, esta vez, a nivel global. El sistema estará trabajando, en ese instante, en condiciones sin fallo, si ninguno de los dos BIC supera su umbral, para un nivel de confianza α. Por supuesto, la alarma de fallo saltará si algún BIC global supera su umbral un número determinado de instantes consecutivos.

4 APLICACIÓN

El método propuesto se ha probado en el benchmark Tennessee Eastman Process [26]. Proceso ampliamente conocido en el campo del diagnóstico de fallos, que se ha usado para probar diferentes métodos de detección de identificación de fallos [3, 14, 27]. Esta benchmark proporciona datos de 52 variables de una instalación tomados cada 3 minutos. En la figura 1 se ha representado el esquema de esta instalación. Los datos disponibles se dividen en datos de entrenamiento y de test. Cada uno de ellos está formado por 21 conjuntos de datos con fallo y uno sin fallo.

4.1 Metodología experimental

Se han hecho pruebas con diferentes parámetros para los métodos propuestos buscando una combinación de parámetros que de lugar al mejor rendimiento en la detección de fallos: el número de retrasos incluidos en la matriz ampliada, el valor α del índice BIC, el número de observaciones anómalas consecutivas para detectar un fallo, la varianza retenida por DDPCA, α utilizada para obtener los umbrales de los estadísticos y los parámetros utilizados para determinar cuántas variables se incluyen en cada método. Estas combinaciones de parámetros se aplicaron sobre los datos de entreno y luego con los datos de test, ajustando, si es necesario, el valor α para BIC, buscando evitar la aparición de falsas alarmas.

Después de experimentar con diferentes valores, el mejor resultado en términos de ausencia de falsas alarmas, número de fallos detectadas y menores tiempos de detección ha sido, para el método LASSO, 2 retrasos para la matriz aumentada, $0,8$ para el parámetro α del criterios BIC, 3 observaciones anómalas consecutivas para detectar un fallo, 80% de la varianza incluida en los DPCA locales, los umbrales locales se ajustaron para un nivel de confianza de 99% y el parámetro λ se ha ajustado de forma individual para cada variable buscando el menor rMSE. El nivel de significación del BIC se reajustó, como se dijo anteriormente, utilizando datos de prueba sin fallo, obteniendo los valores finales: $0,78$ para T^2 y $0,71$ para Q.

En el caso de Random Forest los parámetros utilizados fueron 2 retardos para la matriz aumentada, $0,9$ para el α del BIC, 3 observaciones anómalas consecutivas para detectar un fallo, 90% de varianza incluidas en los DPCA locales, los umbrales de los DPCA locales se obtuvieron para un nivel de confianza de 99% y se han incluido 20 variables en cada bloque. El BIC α se reajustó con los datos de test sin fallo a los valores $0,87$ para T^2 y $0,88$ para Q.

4.2 Resultados

El método propuesto se comparó con un PCA dinámico centralizado aplicado sobre la planta TEP, cuyos resultados aparecen [6]. Para ver el desempeño de cada método se obtuvieron los valores de tasa de detección de fallos no detectados (MDR), que denota los datos con fallo que no son detectados por el método; el tiempo de retraso en la detección, la tasa de falsas alarmas (FAR), que mide el número de observaciones normales identificadas como anómalas; y el número de fallos detectados.

En primer lugar, la tasa MDR se muestra en las Tablas 1, y 2. En T^2 LASSO da el mejor resultado en 12 fallos, Random Forest en 16, y DPCA solo es el mejor en 1 fallo. Para Q, LASSO es el mejor en 11 fallos, Random Forest en 16 y DPCA en 4. Con lo que el mejor desempeño , en cuanto a MDR, lo da el método Random Forest.

En las Tablas 3 y 4 se muestran los resultados del tiempo de detección para estos métodos. En las tablas se muestra el momento en el que el método detecta una observación anómala, pero, como es necesario un cierto número de observaciones anómalas consecutivas para avisar del fallo, el tiempo real de detección es más alto que los resultados mostrados en estas tablas para los tres métodos. En cuanto a tiempos de detección, en T^2, LASSO es el más rápido en 12 fallos, Random Forest en 19 fallos y DPCA en ninguna. Para el estadístico Q, LASSO es el mejor en 14 fallos, Random Forest en 19, y DPCA en ninguno. Claramente, el mejor resultado de tiempos de de-

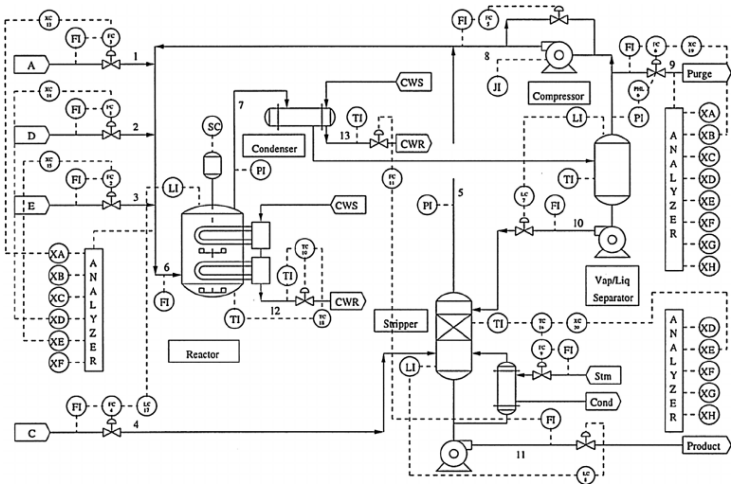

Figura 1: Tennessee Eastman Process diagram

Tabla 1: Missed Detection Rate (MDR), en %. T^2

Fallo	LASSO	Random Forest	DPCA
1	0,25	**0,13**	0,6
2	**1,25**	**1,25**	1,9
3	99,50	**99,1**	**99,1**
4	**92,23**	96,87	93,9
5	**74,56**	75,44	75,8
6	**0**	**0**	1,3
7	**0**	**0**	15,9
8	2,38	**2,01**	2,8
9	99,5	**99,37**	99,5
10	**52,26**	52,38	58
11	**65,16**	**65,16**	80,1
12	**0,63**	**0,63**	1
13	**4,51**	**4,51**	4,9
14	**0**	**0**	6,1
15	97,24	**95,74**	96,4
16	75,31	**74,19**	78,3
17	13,41	**10,90**	24
18	**10,40**	10,53	11,1
19	**97,12**	99,25	99,3
20	52,13	**47,74**	64,4
21	60,15	**57,52**	64,4

Tabla 2: Missed Detection Rate (MDR), en %. Q

Fallo	LASSO	Random Forest	DPCA
1	**0**	**0**	0,5
2	1,25	**0,88**	1,5
3	98,37	**97,37**	99
4	**0**	**0**	**0**
5	**21,30**	74,31	74,8
6	**0**	**0**	**0**
7	**0**	**0**	**0**
8	**2,01**	2,38	2,5
9	98,12	**97,62**	99,4
10	61,53	**40,73**	66,5
11	11,53	**8,02**	19,3
12	**0,88**	2,01	2,4
13	**4,51**	**4,51**	4,9
14	**0**	**0**	**0**
15	98,37	**95,74**	97,6
16	67,54	**41,98**	70,8
17	2,76	**2,13**	5,3
18	**9,40**	9,77	10
19	63,28	**43,23**	73,5
20	35,96	**34,71**	49
21	**53,88**	55,26	55,8

tección lo da el Random Forest, aunque el LASSO también da un buen resultado.

Por último, en la tabla 5 se muestran que los métodos propuestos fueron capaces de detectar más fallos que el DPCA centralizado, siendo LASSO el método con las cifras más altas, 21 fallos detectados con T^2 y Q. Además, Random Forest fue capaz de detectar 20 fallos con ambos estadísticos, mientras que DPCA centralizado solo llega a 18 en el mejor de los casos. La tasa FAR para los datos de prueba tiene un valor 0, que es más baja que DPCA con T^2 y claramente inferior con Q.

5 CONCLUSIONES

En este artículo se han presentado dos métodos para hacer una detección de fallos descentralizada y dinámica. En ambos se usan métodos de regresión para obtener las correlaciones entre las variables que se usan posteriormente para agrupar las variable en bloques.

Al ser comparados con un método dinámico y centralizado, los métodos propuestos demuestran ser más efectivos en cuanto a rapidez en la detección, sensibilidad frente a observaciones anómalas y número de fallos detectados. Además, al ser distribuidos, permiten trabajar en plantas grandes sin necesidad de disponer de equipamiento con gran potencia de procesado de datos. Como líneas futuras de trabajo, se propone ahondar en los

Tabla 3: Retardo en la detección, en instantes. T^2

Fallo	LASSO	Random Forest	DPCA
1	2	**1**	6
2	**10**	**10**	16
3	86	**85**	-
4	74	**72**	151
5	**0**	**0**	2
6	**0**	**0**	11
7	**0**	**0**	1
8	19	**17**	23
9	**3**	-	-
10	47	**38**	101
11	9	**8**	195
12	**0**	**0**	3
13	**36**	**36**	45
14	**0**	**0**	6
15	676	**573**	-
16	195	**33**	199
17	**21**	**21**	28
18	**84**	**84**	93
19	**75**	280	-
20	**80**	**80**	87
21	283	**281**	522

Tabla 4: Retado en la detección, en instantes. Q

Fallo	LASSO	Random Forest	DPCA
1	**0**	**0**	5
2	10	**7**	13
3	**566**	**566**	-
4	**0**	**0**	2
5	**0**	**0**	2
6	**0**	**0**	1
7	**0**	**0**	1
8	**16**	**16**	21
9	**728**	-	-
10	45	**31**	50
11	**3**	**3**	7
12	**0**	**0**	8
13	**36**	**36**	40
14	**0**	**0**	1
15	742	**727**	-
16	13	**11**	196
17	18	**16**	24
18	**15**	78	84
19	43	**11**	82
20	**78**	**78**	84
21	283	**264**	286

Tabla 5: False alarms rates and faults detected (in %)

	LASSO	Random Forest	DPCA
FAR T^2	0	0	0,6
FAR Q	0	0	28,1
Detected faults T^2	**21**	20	17
Detected faults Q	**21**	20	18

métodos de creación de los bloques para encontrar técnicas más efectivas, que sean más eficaces al extraer la correlación entre variables. También, se buscará aplicar otros métodos de detección de fallos, de tipo no lineal.

Agradecimientos

Los autores desean agradecer a la Comisión Europea y al Ministerio de Economía y Competitividad del Gobierno español por su apoyo a través del proyecto: DPI2015-67341-C2-2-R, y a la Universidad de Valladolid a través del Proyecto de Innovación Docente: PID1718-95.

English summary

DYNAMIC AND DECENTRALIZED FAULT DETECTION BASED ON REGRESSION METHODS

Abstract

This article proposes a dynamic and decentralized fault detection method. The plant is divided in blocks using the existing relationships between variables found by regression methods. Each group of variables has a dynamic fault detection method, DPCA, which sends its results to a central processor that fuses the local results using the Bayesian Inference Criterion (BIC), resulting in a global diagnosis. This proposal has been tested on an industrial plant model and compared with the DPCA method to verify its effectiveness.

Keywords: DPCA, Decentralized, Fault detection, LASSO, Random Forest.

Referencias

[1] T. Kourti and J.F. MacGregor. Multivariate SPC methods for process and product monitoring. *Journal of Quality Technology*, 28:409–428, 1996.

[2] R. Muradore and P. Fiorini. A PLS-based statistical approach for fault detection and isolation of robotic manipulators. *IEEE Transactions on Industrial Electronics*, 59(8):3167–3175, 2012.

[3] B. Jiang, D. Huang, X. Zhu, F. Yang, and R. D. Braatz. Canonical variate analysis-based contributions for fault identification. *Journal of Process Control*, 26:17–25, 2015.

[4] P.P. Odiowei and Y. Cao. State-space independent component analysis for nonlinear dynamic process monitoring. *Chemometrics and Intelligent Laboratory Systems*, 103:59–65, 2010.

[5] W. Ku, R.H. Storer, and C. Georgakis. Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and intelligent laboratory systems*, 30:179–196, 1995.

[6] E. L. Russell, L. H. Chiang, and R. D. Braatz. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2000.

[7] Z. Chen, K. Zhang, H. Hao, S. X. Ding, M. Krueger, and Z. He. A canonical variate analysis based process monitoring scheme and benchmark study. *IFAC Proceedings Volumes*, 47(3):10634–10639, 2014. 19th IFAC World Congress.

[8] M. Grbovic, W. Li, P. Xu, A.K. Usadi, L. Somg, and S. Vucetic. Decentralized fault detection and diagnosis via sparse PCA based decomposition and maximum entropy decision fusion. *Journal of Process Control*, 22:738–750, 2012.

[9] C. Tong, T. Lan, and X. Shi. Fault detection and diagnosis of dynamic processes using weighted dynamic decentralized PCA approach. *Chemometrics and Intelligent Laboratory Systems*, 161(Supplement C):34 – 42, 2017.

[10] Q. Cheng, P.K. Varshney, J. Michels, and C.M. Belcastro. Distributed fault detection via particle filtering and decision fusion. In *2005 7th International Conference on Information Fusion*, volume 2. IEEE, 2005.

[11] Y. Zhang, H. Zhou, S.J. Qin, and T. Chai. Decentralized fault diagnosis of large-scale processes using multiblock kernel partial least squares. *IEEE Transactions on Industrial Informatics*, 6(1):3–10, 2010.

[12] W. Li, W. H. Gui, Y. F. Xie, and S. X. Ding. Decentralised fault detection of large-scale systems with limited network communications [brief paper]. *IET Control Theory Applications*, 4(9):1867–1876, September 2010.

[13] C. Tong and X. Shi. Decentralized monitoring of dynamic processes based on dynamic feature selection and informative fault pattern dissimilarity. *IEEE Transactions on Industrial Electronics*, 63(6):3804–3814, June 2016.

[14] Z. Ge and Z. Song. Distributed PCA model for plant-wide process monitoring. *Industrial and Engineering Chemistry Research*, 52:1947–1957, 2013.

[15] R.R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18:183–190, 1988.

[16] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society*, 58:267–288, 1996.

[17] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[18] W. E. Larimore. *Statistical methods in control and signal processing*. Marcel Dekker, 1997.

[19] J. E. Jackson. *A user's guide to principal components*. Wiley, 1991.

[20] Yangming Zhou and Guoping Qiu. Random forest for label ranking. *Expert Systems with Applications*, pages –, 2018.

[21] Dimosthenis Tsagkrasoulis and Giovanni Montana. Random forest regression for manifold-valued responses. *Pattern Recognition Letters*, 101:6 – 13, 2018.

[22] Amod Jog, Aaron Carass, Snehashis Roy, Dzung L. Pham, and Jerry L. Prince. Random forest regression for magnetic resonance image synthesis. *Medical Image Analysis*, 35:475–488, 2017.

[23] Tin Kam Ho. A data complexity analysis of comparative advantages of decision forest constructors. *Pattern Analysis & Applications*, 5(2):102–112, jun 2002.

[24] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*. Springer New York, 2013.

[25] Christopher Bishop. *Pattern recognition and machine learning*. Springer-Verlag New York, 2006.

[26] J. J. Downs and E. F. Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17:245–255, 1993.

[27] Funa Zhou, Ju H. Park, and Yajuan Liu. Differential feature based hierarchical PCA fault detection method for dynamic fault. *Neurocomputing*, 202:27–35, aug 2016.