Aircraft Engine Fleet Monitoring Using Self-Organizing Maps and Edit Distance

Etienne Côme[1], Marie Cottrell[2], Michel Verleysen[3], and Jérôme Lacaille[4]

[1] IFSTTAR - Bâtiment Descartes 2,
2, Rue de la Butte verte, 93166 Noisy le Grand Cedex, France
etienne.come@ifsttar.fr
[2] SAMM - Université Paris 1 Panthéon-Sorbonne
90, rue de Tolbiac, 75013 Paris, France
marie.cottrell@univ-paris1.fr
[3] Université Catholique de Louvain, Machine Learning Group
Place du Levant 3, 1348 Louvain-La-Neuve, Belgium
michel.verleysen@uclouvain.be
[4] Snecma, Rond-Point René Ravaud-Réau,
77550 Moissy-Cramayel CEDEX, France
jerome.lacaille@snecma.fr

Abstract. Aircraft engines are designed to be used during several tens of years. Ensuring a proper operation of engines over their lifetime is therefore an important and difficult task. The maintenance can be improved if efficient procedures for the understanding of data flows produced by sensors for monitoring purposes are implemented. This paper details such a procedure aiming at visualizing in a meaningful way successive data measured on aircraft engines and finding for every possible request sequence of data measurement similar behaviour already observed in the past which may help to anticipate failures. The core of the procedure is based on Self-Organizing Maps (SOM) which are used to visualize the evolution of the data measured on the engines. Rough measurements can not be directly used as inputs, because they are influenced by external conditions. A preprocessing procedure is set up to extract meaningful information and remove uninteresting variations due to change of environmental conditions. The proposed procedure contains four main modules to tackle these difficulties: environmental conditions normalization (ECN), change detection and adaptive signal modeling (CD), visualization with Self-Organizing Maps (SOM) and finally minimal Edit Distance search (SEARCH). The architecture of the procedure and of its modules is described in this paper and results on real data are also supplied.

1 Introduction

During the flights, some on-board sensors measure many parameters related to the behavior (and therefore the health) of aircraft engines. These parameters are recorded and used at short and long terms for immediate action and alarm

J. Laaksonen and T. Honkela (Eds.): WSOM 2011, LNCS 6731, pp. 298–307, 2011.

generation, respectively. In this work, we are interested in the long-term monitoring of aircraft engines and we want to use these measurements to detect any deviations from a "normal" behavior, to anticipate possible faults and to facilitate the maintenance of aircraft engines. This work presents a tool that can help experts, in addition to their traditional tools based on quantitative inspection of some relevant variables, to easily visualize the evolution of the engine health. This evolution will be characterized by a trajectory on a two-dimensional Self-Organizing Map. Abnormal aging and fault will result in deviations with respect to normal conditions. The choice of Self-Organizing Maps is motivated by several points:

- SOMs are useful tools for visualizing high-dimensional data onto a low-dimensional grid;
- SOMs have already been applied with success for fault detection and prediction in plants and machines (see [8] for example).

This article follows another WSOM paper [3] but contains necessary material (and possibly redundant) to be self-contained. It is organized as follows : first, in Section 2, the data and the notations used throughout the paper are presented. The methodology and the global architecture of the proposed procedure are described in Section 3. Each step is defined and results on real data are given in Section 4.

2 Data

Measurements are collected on a set of I engines. On each engine $i \in \{1, \ldots, I\}$, n_i measurements are performed successively flight after flight; there is thus no guarantee that the time intervals between two measures are approximately equal. Each observation is denoted by Z_{ij}, where $i \in \{1, \ldots, I\}$ is the engine number and $j \in \{1, \ldots, n_i\}$ is the flight number.

Each vector Z_{ij} contains two kinds of variables: those which are strictly related to the behavior of the engine (fuel consumption, static pressure, ...), and those which are related to the environment (temperature, altitude, ...). Let the p engine variables be denoted by $Y_{ij}^1, \ldots, Y_{ij}^p$ and the q environmental variables by $X_{ij}^1, \ldots, X_{ij}^q$. Each observation is therefore a $(p+q)$-vector Z_{ij}, where $Z_{ij} = [Y_{ij}, X_{ij}] = \left[Y_{ij}^1, \ldots, Y_{ij}^p, X_{ij}^1, \ldots, X_{ij}^q\right]$. The variables at disposal are listed in Table 1. There are $p = 5$ engine variables and $q = 15$ environmental variables. The dataset contains measurements for approximately one year of flights and $I = 91$ engines, that leads to a global dataset with $\sum_{i=1}^{91} n_i = 59407$ $(p+q)$-dimensional observations.

3 Methodology

The goal is to build the trajectories of all the engines, that is to project the successive observations of each engine on a Self-Organizing Map, in order to follow the evolution and to eventually detect some "abnormal" deviation. It is

Table 1. Variables names, descriptions and type

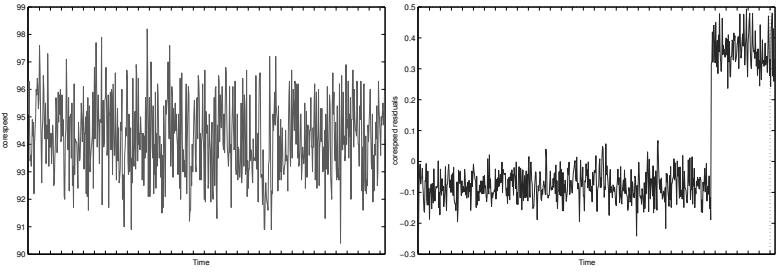
	Name	Description	Type	Binary
	aid	aircraft id		
	eid	engine id		
	fdt	flight date		
X_{ij}^1	temp	temperature	environment	
X_{ij}^2	nacelletemp	nacelle temperature	environment	
X_{ij}^3	altitude	aircraft altitude	environment	
X_{ij}^4	wingaice	wings anti-ice	environment	✓
X_{ij}^5	nacelleaice	nacelle anti-ice	environment	✓
X_{ij}^6	bleedvalve	bleed valve position	environment	✓
X_{ij}^7	isolationleft	valve position	environment	✓
X_{ij}^8	vbv	variable bleed valve position	environment	
X_{ij}^9	vsv	variable stator valve position	environment	
X_{ij}^{10}	hptclear	high pressure turbine setpoint	environment	
X_{ij}^{11}	lptclear	low pressure turbine setpoint	environment	
X_{ij}^{12}	rotorclear	rotor setpoint	environment	
X_{ij}^{13}	ecs	air cooling system	environment	
X_{ij}^{14}	fanspeedi	N1	environment	
X_{ij}^{15}	mach	aircraft speed	environment	
Y_{ij}^1	corespeed	N2	engine	
Y_{ij}^2	fuelflow	fuel consumption	engine	
Y_{ij}^3	ps3	static pressure	engine	
Y_{ij}^4	t3	temperature plan 3	engine	
Y_{ij}^5	egt	exhaust gas temperature	engine	

not valuable to use the rough engine measurements: they are inappropriate for direct analysis by Self-Organizing Maps, because they are strongly dependent on environment conditions and also on the characteristics of the engine (its past, its age, ...). The first idea is to use a linear regression for each engine variable: the environmental variables (real-valued variables) and the number of the engines (categorical variable) are the predictors and the residuals of these regressions can be used as standardized variables (see [3] for details). For each engine variable $r = 1, \ldots, p$, the regression model can be written as:

$$Y_{ij}^r = \mu^r + \alpha_i^r + \lambda_1^r X_{ij}^1 + \ldots + \lambda_q^r X_{ij}^q + \epsilon_{ij}^r \tag{1}$$

where α_i^r is the engine effect on the r^{th} variable, $\lambda_1^r, \ldots, \lambda_q^r$ are the regression coefficients for the r^{th} variable, μ^r is the intercept and the error term ϵ_{ij}^r is the residual.

Figure 1 presents for example the rough measurements of the *corespeed* feature as a function of time (for engine 6) and the residuals computed by model (1). The rough measurements seem almost time-independent on this figure, whereas the residuals exhibit an abrupt change which is linked to a specific event in the life of this engine. This simple model is therefore sufficient to bring to light interesting aspects of the evolution of this engine. However, the signals may contain



(a) (b)

Fig. 1. (a) Rough measurements of the **corespeed** variable as a function of time for engine 6, (b) residuals of the same variable and for the same engine using a simple linear model with the environmental variables and the engine indicator as predictors (see Table 1).

ruptures, making the use of a single regression model hazardous. The main idea of this work is to replace model (1) by a new procedure which deals with the temporal behavior of the signals. The goal is therefore to detect the ruptures and to use different models after each rupture. This new procedure is composed of two modules. The first module (Environmental Conditions Normalization, ECN) aims at removing the effects of the environmental variables to provide standardized variables, independent of the flight conditions. It is described in section 4.1. The second module uses an on-line change detection algorithm to find the above mentioned abrupt changes, and introduces a piecewise regression model. The detection of the change points is done in a multi-dimensional setting taking as input all the normalized engine variables supplied by the ECN module. The Change Detection (CD) module is presented in Section 4.2. As a result of these first two steps, the "cleaned" database can be used as input to a Self-Organizing Map with a "proper" distance for trajectories visualization. The third module (SOM) provides the "map" on which the trajectories will be drawn. Finally, engine trajectories on the map are gathered in a trajectory database which can be accessed through a SEARCH module, which use a dedicated Edit Distance to find similar trajectories. This four-steps procedure is summarized in Figure 2.

4 Description of the Four Modules

4.1 Environmental Conditions Normalization - ECN

The first module aims at removing the effects of the environmental variables. For that purpose, one regression model has to be fitted for each of the p engine variables. As the relationship between environmental and engine variables is complex and definitively not linear, the environmental variables can be supplemented by some non-linear transformations of the latter, increasing the number of explanatory variables. Interactions (all the possible products between two environmental variables), squares, cubes and fourth powers of the non binary environmental variables are considered. The number q of predictors in the model

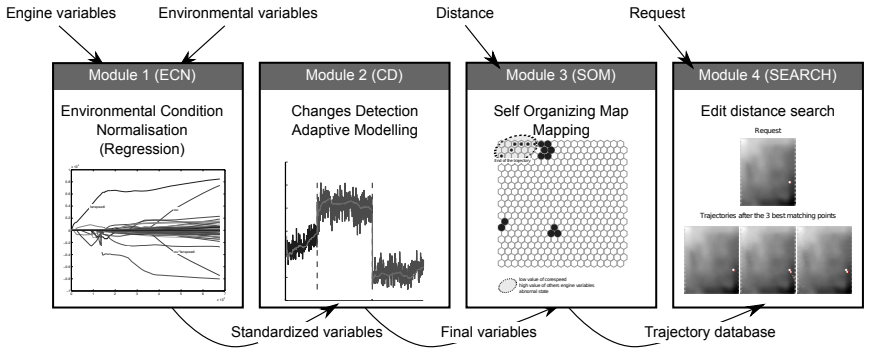


Fig. 2. Global architecture of the health monitoring tools

is therefore a priori equal to $(11+4)*(11+4-1)/2 = 105$ for the interactions variables and $11*4+4 = 48$ for the power of the continuous variables and the binary variables leading to a total of $q = 153$ predictors. This number is certainly too large and some of them are clearly irrelevant due to the systematic procedure used to build the non-linear transforms of environmental variables. A LASSO criterion [4] is therefore used to estimate the regression parameters and to select a subset of significant predictors. This criterion can be written using the notations from Section 2 for one engine variable Y^r, $r \in \{1, \ldots, p\}$ as :

$$\boldsymbol{\beta}^r = \arg\min_{\boldsymbol{\beta}^r \in \mathbb{R}^q} \sum_{i,j=1}^{I,n_i} \left(Y_{ij}^r - \sum_{l=1}^{q} \beta_l^r \, X_{ij}^l \right)^2 , \; \sum_{l=1}^{q} |\beta_l^r| < C^r \tag{2}$$

The regression coefficients are penalized by a condition which forces some of them to be null for a well chosen value of C^r. The LARS algorithm [4] is used to estimate all the solutions of the LASSO criterion (2) for all possible values of C^r. The optimal value of C^r with respect to the prediction error estimated by cross-validation (with 20 blocs) is finally selected. Another possibility could be to use BIC criterion instead of cross validation procedure to pick up the best model. The number of selected predictors and the coefficient of determination R^2 are listed in Table 2 for all engine variables. Engine variables are well explained by the proposed models as attested by the high value of the coefficients of determination.

A qualitative inspection of the model results was also carried out with the help of engine experts. The regularization path plot (as shown in Figure 3) is very

Table 2. Number of selected predictors and coefficients of determination for all engine variables

	corespeed	*fuelflow*	*ps3*	*t3*	*egt*
nb vars	25	43	31	30	41
R^2_{obs}	0.9875	0.9881	0.9773	0.9636	0.8755

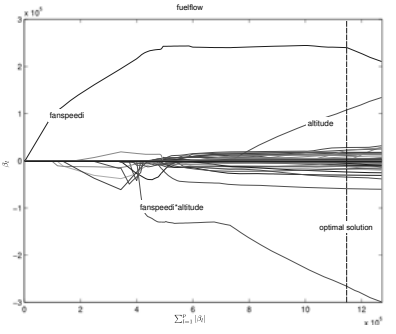


Fig. 3. Regularization path for the *fuelflow* variable: regression coefficients evolution with respect to C^r. The more significant explanatory variables are given and the best solution with respect to cross-validation is depicted by a vertical line.

interesting from the point of view of the experts, because it can be compared with their previous knowledge. Such a curve clearly highlights which are the more relevant predictors and they appear to be in very good adequateness with the physical knowledge on the system.

In summary, the first preprocessing module (ECN) provides $p = 5$ standardized engine variables denoted by $S^r = [S^r_{ij}, i \in \{1, \ldots, I\}, j \in \{1, \ldots, n_i\}]$, with $r \in \{1, \ldots, p\}$, which are the residuals of the selected regressions. They are independent of environmental conditions but still contain some significant aspects such as linear trends and abrupt changes at specific dates. We therefore propose to use an on-line Change Detection algorithm (CD) together with an adaptive linear model to fit the data.

4.2 Change Detection - CD

To take into account the two types of variation (linear trend and abrupt changes), we implement an algorithm based on the ideas from [5] and [7]. The solution is based on the joint use of an on-line change detection algorithm to detect abrupt changes and of a bank of recursive least squares (RLS) algorithms to estimate the slow variations of the signals. The algorithm works on-line in order to allows projecting new measurements on the map as soon as new data are available.The method can be described as follows:

1) One RLS algorithm is used for each one of the p standardized engine variables to recursively fit a linear model. For each $r \in \{1, \ldots, p\}$, for each engine $i \in \{1, \ldots, I\}$ and at each date l, one has to solve the following equation:

$$(\alpha^r_{il}, \beta^r_{il}) = \arg \min_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}} \sum_{j=1}^{l} \lambda^{(l-i)} (S^r_{ij} - (\beta j + \alpha))^2, \tag{3}$$

where λ is a forgetting factor. The estimates α^r_{il} and β^r_{il} are respectively the intercept and the slope of the linear relationship. These estimates are then used

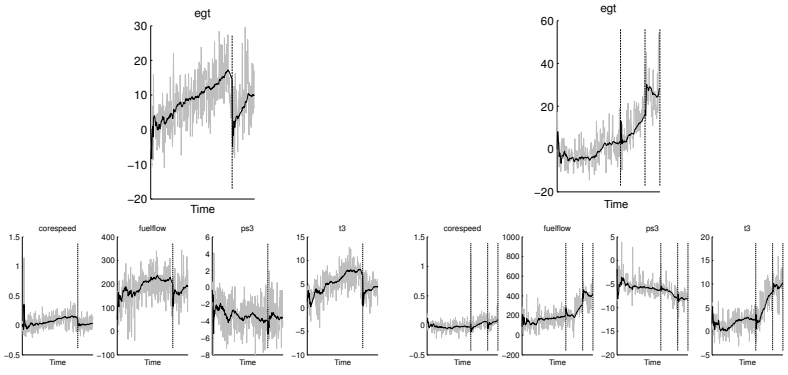


Fig. 4. Change detection results for two engines, (a) engine 2, (b) engine 41. Alarms are depicted by vertical lines, input signals are shown in light gray and signal estimates F^r using RLS are depicted by a black line. One figure (with respect to *egt*) is bigger than the others to present more clearly the RLS estimate of the signal.

to define the variables $\varepsilon_{il}^r = S_{il}^r - (\beta_{il}^r l + \alpha_{il}^r)$, which do not contain anymore the slow variations of the signals.

2) These values are concatenated in a vector $\boldsymbol{\varepsilon}_l = [\varepsilon_l^1, \ldots, \varepsilon_l^p]$, which is then used in a multi-dimensional Generalized Likelihood Ratio (GLR) algorithm [1] to detect the abrupt changes of the signals. The GLR algorithm is a sequential test procedure based on the following model:

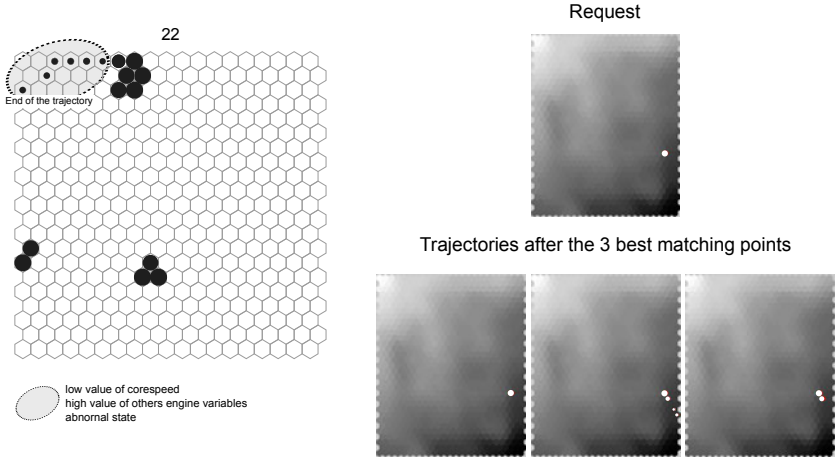
$$\boldsymbol{\varepsilon}_k \sim \mathcal{N}_p(\theta(k), \Sigma), \forall k > 0,$$

where $\mathcal{N}_p(\theta(k), \Sigma)$ is the multivariate normal distribution with variance Σ and mean $\theta(k) = \begin{cases} \theta \in \Theta_0 = \{||\theta|| < r_o\} & \text{if } k < t_0, \\ \theta \in \Theta_1 = \{||\theta|| > r_1\} & \text{if } k \geq t_0. \end{cases}$, ($r_0 < r_1$ are given constants).

3) Finally, when an alarm is sent by the GLR algorithm, all the RLS algorithms are re-initialized. The results supplied by this algorithm are the following:

- the alarm dates supplied by the multi-dimensional GLR algorithm;
- cleaned signals estimated by the RLS algorithm;
- slopes and intercepts estimated by the RLS algorithm.

Figure 4 presents the obtained results for two engines. One abrupt change was found for the first engine and 3 for the second one; all of them seem to be reasonable and a comparison between estimated alarm dates and recorded real events of the engine life have confirmed this fact. The estimated signals are also shown on these two figures. For more information on this aspect of the analysis process see [2]. From now, the observations corresponding to each flight are $F_{il} = [F_{il}^1, \ldots, F_{il}^p]$, where $F_{il}^r = \beta_{il}^r l + \alpha_{il}^r$ are the results of the transformations of the raw data performed by the first two modules (ECN and CD).



(a) (b)

Fig. 5. (a) Trajectories of engine 22 on the map. The sizes of the dots are proportional to the measurement date: smallest dots correspond to recent measurements, larger dots to older measurements. (b) Pieces of similar trajectories found using the edit distance (details are given in section 4.4)

4.3 Self-Organizing Maps - SOM

The cleaned signals F_{il} provided by the previous two modules are then used as input to a SOM for visualization purpose. To project the observations we use a $[20 \times 20]$ SOM implemented with the Matlab toolbox [9] and with defaults settings for the learning rate. Since the variables F_{il}^r are correlated, a Mahalanobis distance is used to whiten the data. A classical learning scheme is used to train the map. Figure 5 (a) presents one example of engine trajectories on the map, which clearly have different shapes. For the studied engine, available maintenance reports inform us that this engine suffers from an deterioration of its high pressure core. This fault is visible on the map at the end of the trajectory: the engine which was projected on the middle north of the map during a large part of its trajectory, suddenly moves towards the north-west corner of the map. This area of the map furthermore corresponds to abnormal values of the engine variables.

4.4 Similar Trajectory Matching - SEARCH

One of the final goal of the proposed tool concerns clustering and prediction of engine trajectories or pieces of engine trajectories. For this end, we have to define a proper distance between pieces of trajectories, which can be of different lengths. Before projection on the map, pieces of trajectories were sequences of $\mathbb{R}^p$-vectors, but as soon as measurements are projected, they can be described by sequences of integers corresponding to the units where measurements are projected. Such sequences will be denoted by $T = [k_1, \ldots, k_L]$ and as they take

their values in a finite set $\{1, \ldots, U\}$ (where U is the number of units), we will call them *strings*. The difficulty comes from the fact that the strings can have different lengths. Such a problem has been already investigated in other fields and one classical solution is to use Edit Distance, which is commonly used for approximate string matching [6].

To compare two strings, Edit Distance uses a *cost function*. This function gives individual cost for each unitary operation such as: suppression, addition or substitution. The cost of a sequence of operations $O = [o_1, o_2, \ldots]$ is simply equal to the sum of all the unitary costs, so: $cost(O) = \sum_t cost(o_t)$. Then, the Edit Distance between two strings $de(T, T')$ is defined as the minimal cost among all sequences of operations that fulfill the constraint $O(T) = T'$. Such a distance can be tuned to our end by carefully choosing unitary costs. We may in particular use the map topology to define meaningful substitution costs, by setting the cost of the substitution $k \leftrightarrow k'$ to the distance between unit k and unit k' on the map. With such a choice, we will take benefit of the fact that close units on the map can be exchanged with a small cost. Suppression and insertion costs are equal to the average of all the pairwise distances between units of the map.

With such a distance one can build classes of pieces of trajectories using hierarchical clustering. But this distance can also be used to supply clues on the possible future evolution of one trajectory. To perform such a task, the following method is proposed. Let T be a piece of engine trajectory:

1. compute the Edit Distance between T and all the pieces of engine trajectories recorded in the fleet database $[T_1, T_2, \ldots]$ (all these distances can be computed efficiently using dynamic programming [6]);
2. search for matching pieces T_x such that $de(T_x, T) < \eta$, where η is a given threshold;
   Note that these pieces are parts of already observed engine trajectories, which were recorded in the fleet database so that their evolutions after the matching points are therefore known, the third step uses this property.
3. look at the pieces that are just after the matching pieces. That gives an idea about the possible futures of T and enables the computation of probabilities of different types of maintenance events if the fleet database is connected to the maintenance database which recorded all the failures and maintenance operations performed on the fleet. We hope that it will be a useful tool to anticipate failures.

Figure 5 (b) presents preliminary results obtained using such an approach, T was built using the last 50 points of an engine trajectory. During this time period, this engine stays in the same unit. We show in Figure 5 (b) the pieces of trajectories that occurred after the 3 best matching points found in the fleet database using the proposed Edit Distance. These possible futures for T seem to be reasonable. Further works concern the connection with the maintenance database to perform a quantitative analysis of the results.

5 Conclusion

The method proposed in this paper is an nice tool to summarize and represent the temporal evolution of an aircraft engine health flight after flight. The regression approach used to deal with the problem of environmental condition normalization (ECN) seems to be effective, even if other model selection methods such as the BIC criterion could be investigated in further works to reduce the number of selected variables. The joint use of an adaptive algorithm to estimate signal evolution (RLS) and of a change points detection method (GLR) is also an interesting solution to deal with the non-stationary of the signals and to clean them (GLR module). Finally, Self-Organizing Maps (SOM) can be used to show the engine health evolution in a synthetic manner and to provide codes for synthetic representation of trajectories, that enables the development of predictive analysis tools (SEARCH module).

References

1. Basseville, M., Nikiforov, I.: Detection of Abrupt Changes: Theory and Application. Prentice-Hall, Englewood Cliffs (1993)
2. Côme, E., Cottrell, M., Verleysen, M., Lacaille, J.: Aircraft engine health monitoring using self-organizing maps. In: Springer (ed.) Proceedings of the Industrial Conference on Data-Mining (2010)
3. Cottrell, M., Gaubert, P., Eloy, C., François, D., Hallaux, G., Lacaille, J., Verleysen, M.: Fault prediction in aircraft engines using self-organizing maps. In: Príncipe, J.C., Miikkulainen, R. (eds.) WSOM 2009. LNCS, vol. 5629, pp. 37–44. Springer, Heidelberg (2009)
4. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.J.: Least angle regression. Annals of Statistics 32(2), 407–499 (2004)
5. Gustafsson, F.: Adaptive filtering and change detection. John Wiley & Sons, Chichester (2000)
6. Navarro, G.: A guided tour to approximate string matching. ACM Computing Surveys 33, 2001 (1999)
7. Ross, G., Tasoulis, D., Adams, N.: Online annotation and prediction for regime switching data streams. In: Proceedings of ACM Symposium on Applied Computing, pp. 1501–1505 (March 2009)
8. Svensson, M., Byttner, S., Rognvaldsson, T.: Self-organizing maps for automatic fault detection in a vehicle cooling system. In: 4th International IEEE Conference on Intelligent Systems, vol. 3, pp. 8–12 (2008)
9. Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J.: Som toolbox for matlab 5. Tech. Rep. A57, Helsinki University of Technology (April 2000)