Reconfiguration process for neuronal classification models: Application to a quality monitoring problem

Melanie Noyel, Philippe Thomas, André Thomas, Patrick Charpentier

Computers In Industry 83 (2016) pp 78-91

Reconfiguration process for neuronal classification models: Application to a quality monitoring problem

Mélanie NOYEL[a,b], Philippe THOMAS[a*], André THOMAS[a], Patrick CHARPENTIER[a]

[a]*Université de Lorraine, CRAN, UMR 7039 CNRS, Campus Sciences, BP 70239, 54506 Vandœuvre-lès-Nancy cedex, France*

[b]*ACTA Mobilier Parc d'activité Macherin Auxerre Nord, 89470 Monéteau, France*

[*]*corresponding author*

Phone: +33 (0)3 83 68 44 30

Fax: +33 (0)3 83 68 44 59

Email: mnoyel@acta-mobilier.fr; philippe.thomas@univ-lorraine.fr; patrick.charpentier@univ-lorraine.fr; andre.thomas@univ-lorraine.fr

Abstract

In the context of smart industries, learning machines currently have various uses such as self-reconfiguration or self-quality improvement, which can be classification forecasting problems. In this case, learning machines are tools that facilitate the modeling of the physical system. Thus, it is obvious that the model must evolve with changes in the physical system, thereby leading to adaptability/reconfigurability problems. Among the various tools reported previously, real-time systems seem to be the best solution because they can evolve autonomously according to the behavior of the physical system. In the present study, we propose a method for using learning machines efficiently in an evolving context. This method is divided into two components: (1) model conception by defining the objective function and influential factors, setting up data collection, and learning using multilayer perceptrons; and (2) monitoring system conception with the aim of tracking the misclassification rate, determining whether the physical system is drifting, and reacting by model adaptation based on the control charts. This paper focuses on the model monitoring procedure because the model conception procedure is quite classical. The proposed method was applied to a benchmark derived from previous research and then to an industrial case of defect prevention on a robotic coating line for which other methods have proved unsuccessful.

Keywords

classification, control chart, learning, multilayer perceptron, neural network, quality, relearning

1 Introduction

In the smart industry domain, the elements of the physical system share information with each other and with a real time decision-making system. These decision systems must evolve and adapt according to the physical state of the environment. For example, in the quality control domain, we may consider the impact of environmental factors (such as temperature and humidity) on a die-casting process (Thomas et al. 2004) or a lacquering process (Noyel et al. 2013a), as well as the impact of tool wear on the finished surface in a machining process (Sick 2002). Other examples of real-world change detection problems include modeling in bio-medicine monitoring and industrial processes, as described by Basseville and Nikiforov (1993).

These different contexts lead to classification or supervised classification problems, which can be resolved using machine learning algorithms, where various techniques can be applied, including logic-based algorithms (such as decision trees and rule-based classifiers), neural networks (NNs such as multilayer perceptrons (MLPs) and radial basis function networks), statistical learning (such as naive Bayes classifiers and Bayesian networks), and support vector machines (SVMs). The present study focuses on classification problems where continuous data are mainly considered. In this case, Kotsiantis (2007) has highlighted that the most suitable approaches are logic-based algorithms, NNs, and SVMs.

To extract a classifier from data using machine learning, the complete learning dataset is provided to the learning machine, which obtains descriptions for the underlying concepts in the dataset (Lazarescu et al. 2003). This type of learning is called batch learning. However, the target concept may be non-stationary and it can change over time, thereby leading to concept modification (Michalski et al. 1986). Gama (2010) separated this conceptual evolution into concept drift when the evolution is gentle and concept shift when the evolution is sudden. For example, a concept drift may result from tool wear or filter clogging, whereas a concept shift may occur after the replacement of parts or a system modification. When a concept shift or concept drift occurs, the model obtained using batch learning may no longer be accurate. Therefore, incremental algorithms, online algorithms, and anytime algorithms have been proposed to respond to this problem. Incremental algorithms consider new data in order to adapt the model without restarting the complete learning process (Salperwick et al. 2009). Online algorithms are used when the stream of data is continuous and the data are used individually and sequentially (Salperwick et al. 2009). In these two approaches, the learning must be faster and the data are generally presented only once to the algorithm. Anytime learning is defined as learning the best model (considering a given criterion) until a break occurs (such as new data arrival) (Dean and Boddy 1988). In these three approaches, the learning and exploitation of the model must be performed simultaneously, so the computational time required may be prohibitive in real-time applications, even if anytime approaches include resource constraints. These learning approaches may exhibit slower convergence than batch approaches, thereby leading to an inaccurate model. Moreover, because the data are used individually, outliers may have a deleterious impact on the model obtained. Finally, incremental learning must address the plasticity-stability dilemma (Bouchachia 2011), which demands a compromise between the stability and reconfigurability of the model.

Due to the limitations of the online approaches, batch learning is useful because it can be performed in real time whereas learning is performed offline. Batch learning can approximate every nonlinear function with the desired accuracy, as well as using a variety of algorithms to avoid bad local minima or the overfitting problem, and cross-validation can be performed based on the validation dataset. None of these processes can be performed with incremental learning, and Sarle (2002) showed that it is generally more difficult and unreliable than batch learning. Considering the drift or shift concepts allows changes to be detected during batch learning, and change detection can be performed with different approaches (Salperwick et al. 2009), as follows:

- Relearning the classifier from scratch,
- Adapting the classifier,
- Adapting the data summary used in the classifier (e.g., the kNN model),
- Using the sequence of classifiers that is learned over time in a classifier ensemble as example (Bifet and Gavalda 2007).

Using of classifier ensemble allows to improve the accuracy of the classification. However, many individual classifiers must be evaluated simultaneously and this fact may be time consuming during the exploitation phase of the model. Adapting the classifier by parameters relearning allows to limit the computational cost by considering that even if a drift occurs, the original model is not so far of the desired one. However, in case of major context change, the model structure itself must be corrected. In this case, the classifier must be relearned from scratch.

We propose an adaption of the classifier in order to limit the computational cost. We focus on the use of a MLP classification model due to its adaptability to change. In fact, the adaptation of a MLP classifier may be performed by a learning process based on the new dataset using the initial model parameters as the parameter set

for initialization. It is assumed that even if a concept shift or concept drift occurs, the initial classifier is no more accurate but its parameter set will remain close to the optimal set.

The following different approaches may also be used for change detection (Gama et al. 2014):

- Using sequential analysis (e.g., sequential probability ratio test (Wald 1947), Page-Hinkley test (PHT) (Page 1954, Hinkley 1971), and cumulative sum (Page 1954));
- Using statistical process control (SPC) (Klinkenberg and Rentz 1998, Gama et al. 2004, Bouchachia 2011);
- Monitoring the distribution based on two different time windows (Dries and Ruckert 2009, Adae and Berthold 2013);
- Contextual approaches (Harries et al. 1998, Bouchachia 2011).

Sequential analysis approaches are very sensitive to the choice of the detection threshold (Ragot et al. 1990). The main limitation of time window-based approaches is the possibility of high memory consumption (Gama et al. 2014). Contextual approaches are used mainly in conjunction with incremental learning. Thus, in the present study, a SPC approach is used to determine when relearning is required. However, even if a change is detected and the need for relearning is evident, a question remains: What dataset should we use for relearning? PHT can estimate the time when a drift or shift concept starts (Ragot et al. 1990); thus, PHT may be used in conjunction with SPC to build the relearning dataset. Finally, industrial datasets are often affected by outliers so a robust learning algorithm is required in order to limit the impact of outliers on the classifier.

The main contribution of this paper is the proposition of a model monitoring procedure which associates SPC and PHT algorithms in order to adapt the model to change (by using relearning procedure). MLP is used as model and its adaptation is performed by using backpropagation algorithm. SPC is used to estimate the need of model's adaptation. PHT is used to determine the dataset which must be used during the relearning procedure.

In the following, after explaining the batch learning approach for designing the monitoring system, we discuss the impact of changing the context. The need for adaptation is highlighted and the proposed method is developed in two steps: determining when the system is finally drifting using control charts and evaluating how much data must be relearned. Finally, the proposed approach is tested on both a benchmark case and an industrial case, i.e., a quality monitoring problem for a lacquering company.

2 Description of the batch learning process

To predict the behavior of a real system, a classical approach involves the design of a forecasting model (Figure 1). The behavior of this forecasting system, which is parallel to the physical one, is as similar as possible to that of the real system. It can measure different parameters in the physical system and compare its forecasts with reality. For classification problems, the considered forecasting system can predict the class of the output data; therefore, it can be used to evaluate the decision taken upstream.

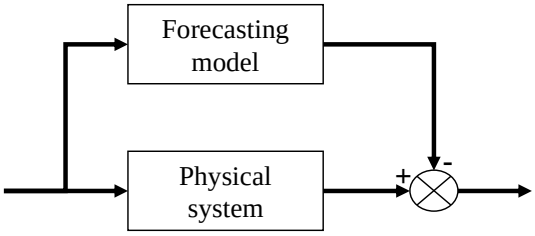

Figure 1 - Relationship between the forecasting model and physical system.

The classical approach to the design of the forecasting model employs a learning machine in order to extract the forecasting model directly from the data using batch learning. This task is performed according to a classical knowledge discovery and data mining (KDD) process, which comprises two main steps: collecting the dataset (including identification and data collection and pre-processing) and the data mining task (which needs to define

the training and validation datasets) (Patel and Panchal 2012). This KDD process is summarized by figure 2 and the two main task will be more detailed in the two following subparts.

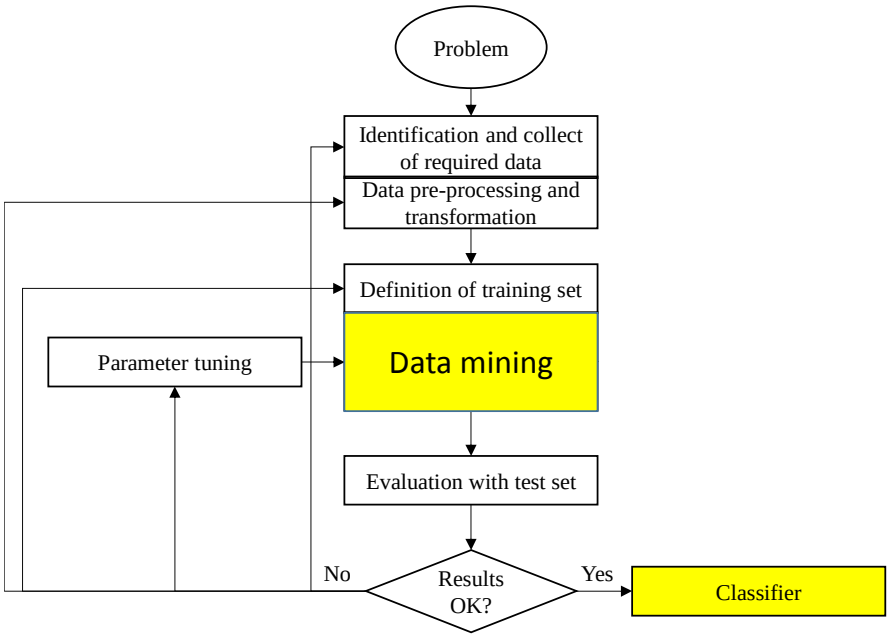

Figure 2: The process of supervised machine learning (Kotsiantis 2007).

2.1 Dataset collection

To design the dataset collection, the first step is to define the objective function, which is the output of the forecasting model and it corresponds to the characteristics of the physical system that we aim to monitor.

Next, the factors that are most likely to affect the objective function must be identified. Different efficient strategies may be used in order to identify these influential factors, such as the Ishikawa method.

After the objective function and influential factors have been identified, the values of these different parameters must be collected. This step should aim to improve the instrumentation by adding some captors in order to collect influential factors that have not yet been collected. However, in some cases, automatic data collection is not possible and manual data collection must be performed. Unfortunately, manual data collection is often viewed as a waste of time by the operators, and the low priority given to this task often leads to corruption of the dataset by outliers. To reduce this risk, it is necessary to ensure that the operators consider the importance of this task and to make the interfaces as intuitive and quick as possible (Noyel et al. 2013b).

The data preparation process required before learning includes selecting, preprocessing, and transforming the data. For example, the data must be preprocessed in order to synchronize the different databases, delete evident outliers, and digitalize qualitative data (e.g., as colors) (Patel and Panchal 2012).

After these steps, a dataset can be used to extract knowledge with learning tools.

2.2 Data mining or learning

Management and quality improvement using data mining methods were discussed by Kusiak (2001). Data mining is the main part of the KDD process, which involves data analysis to summarize the data in the form of useful information. The KDD process may be performed to identify valid, novel, useful, and understandable patterns by exploiting the full volume of data collected.

As shown by Agard and Kusiak (2005), the volume of data that needs to be analyzed is often large. Classically, the collected dataset is divided into two parts, where one is used for learning the model and the other for validation. This procedure allows us to check that our model exhibits the same behavior as the physical system.

Different tools may be used to perform the data mining task, such as Naïve Bayes, decision trees, SVMs, and NNs. Decision trees are faster at classifying the data but they do not work well with noisy data (Patel and Panchal 2012); therefore, this approach is not efficient with industrial data. Naïve Bayes is appropriate for the treatment of discrete data, so we need to discretize the data to apply this approach to continuous data. Both SVMs and NNs employ very similar concepts, and thus they yield very similar results, where SVMs sometimes give better results (Meyer et al. 2003) whereas NNs may give the best at other times (Paliwal and Kumar 2009). These four tools (SVM, MLP, decision trees, and kNN) have been tested and compared based on the real example used in this study, and MLP obtained the best results (Noyel 2015). The proposed approach presented section 3 may be adapted to each type of model. However, in the present study, we focus on the use of a MLP with only one hidden layer, a sigmoidal activation function, and an output neuron. Its structure is given by the following formula (1):

$$z = g_2\left(\sum_{i=1}^{n_1} w_i^2 . g_1\left(\sum_{h=1}^{n_0} w_{ih}^1 . x_h^0 + b_i^1\right) + b\right), \quad (1)$$

where x_h^0 denote the n_0 inputs of the MLP, w_{ih}^1 indicate the weights connecting the input layer to the hidden layer, b_i^1 represent the biases of the hidden neurons, $g_1(.)$ refers to the activation function of the hidden neurons (in this case, the hyperbolic tangent), w_i^2 denote the weights connecting the hidden neurons to the output, b indicates the bias of the output neuron, $g_2(.)$ represents the activation function of the output neuron, and z denotes the network output. This problem involves classification between two classes 0 and 1. So $g_2(.)$ is selected as sigmoidal in order to allow the evolution of the output value between these two bounds 0 and 1.

In order to determine the number of hidden neurons and to discard spurious inputs, the learning phase starts with an overparameterized structure. This structure includes all the variables collected during the preceding step and a number of hidden units clearly greater than necessary (between 2 and 3 times greater than the inputs number). The weights of this overparameterized structure are initialized using classical Nguyen and Widrow algorithm (Nguyen and Widrow 1990, Mathworks 2016), and the learning of these weight is performed by using the Levenberg–Marquard algorithm with a robust criterion (Thomas et al. 1999, Mathworks 2016) in order to avoid the outliers impact on the resulting model. Due to the overparameterized structure, the resulting model presents overfitting. In order to avoid it, a weight elimination algorithm (pruning) is used to discard spurious inputs and hidden nodes (Thomas and Suhner 2015, Mathworks 2016). All this procedure is performed on a part of the available dataset called learning dataset. The accuracy of the obtained model is determined by using the remaining data (not used for the learning) and is called validation dataset.

The completion of this process yields a classification model, which can be used as a forecasting system to monitor the real system. However, this model is static whereas the system or its environment can evolve. Therefore, an adaptation procedure is needed to fit the model to the real system if changes occur.

3 Proposition of classification neuronal model adaptation process

3.1 Limits of batch learning in a changing context

Two different types of changes can occur in the monitored system (Sebastiéão and Gama 2009):

- A “concept shift” refers to an abrupt change;
- A “concept drift” is associated with gradual change.

Concept drift is more difficult to detect and it is often confused with noise. However, this is the type of change that we must identify in order to ensure that the behavior of the monitoring system remains as close to reality as possible.

In our previous study (Noyel et al. 2013a), we highlighted two reasons why the behavior of the forecasting system starts to deviate from reality. The first concerns the evolution of the input parameters. Thus, with a learning model, the learning outcome is valid only in the learned domain. In many cases, the data range which can be determined with the dataset collected (weak bounds) don't correspond to the complete evolution range of the considered variable (strong bounds). As example, if we study the evolution of output temperature in Paris during January 2016, the bounds of the evolution range (weak bounds) are -4.4°C and 13.9 °C. However, by considering all the dataset available since 1873 the temperature has evolved between -14.9°C and 16.1°C and we can imagine that these record values may be outperformed in the future and so, the strong bounds are unknown. For the learning process, the main risk is encountering a situation where one factor is outside the bounds of the learned domain (weak bounds) because the model cannot give a correct answer. For the example shown in Figure 3, the strong bounds are given by dashed lines when the colored sectors correspond to the known range given by the dataset. In this figure, forecasts are possible in cases 1 and 2 but not in cases 3 (factor 5 is out of weak bounds) and 4 (factor 2 is out of weak bounds). These points may be outside the learning domain due to concept drift (e.g., a gradual change) or concept shift (e.g., a pressure drop due to a compressor failure).

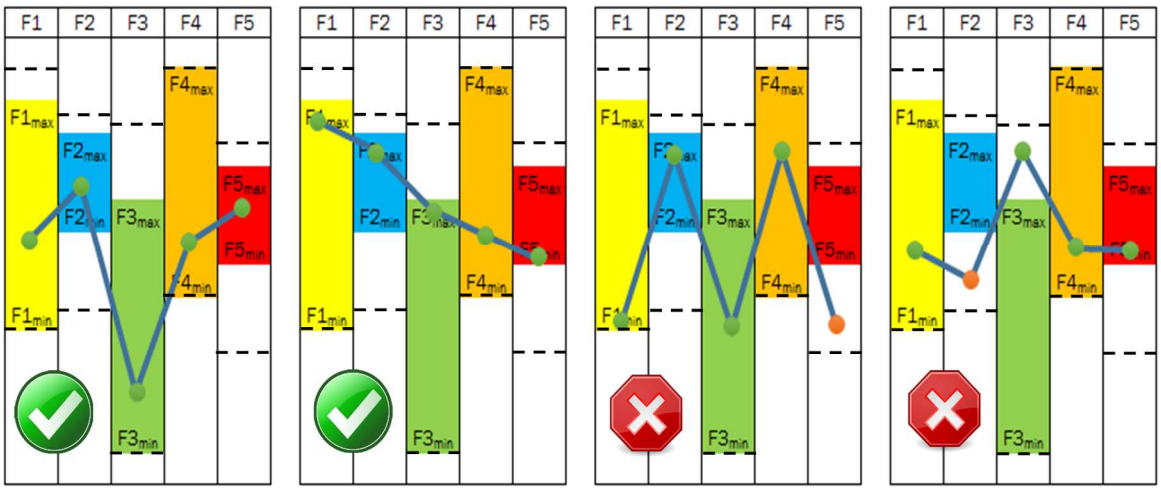

Figure 3 - Situations where a forecast may or may not be possible.

The second reason concerns the uncontrolled modification of the behavior of the real system. In fact, it is possible to affect the behavior of the real system by changing a parameter (voluntarily or not) that is not an input of the forecasting system.

Therefore, the main issue is synchronization with reality, where we have to optimize the synchronization frequency because synchronization is time consuming (a revision of the model can take several minutes to several days). It is better to rely on statistical findings (using SPC tools as example) rather than considering the resynchronization frequency in terms of the response to events (such as the arrival of new information from one of the connected devices (incremental learning) or solicitation by an operator) or over a period (e.g., every hour or week). Among the seven basic tools, control charts, also known as "Shewhart charts" or "process-behavior charts" (Shewhart, 1931), are useful SPC tools in the proposed monitoring system.

First, a prerequisite for detecting changes is the ability to verify hypotheses after being informed about reality. This step requires an improvement to the information technology system in order to collect data as the output from the real system. We can monitor the difference between theory and reality by comparing these data with the system hypotheses, where this value is called the error rate. The evolution of this rate can be disturbed by the

normal production noise. At first glance, it appears to be difficult to determine when the system is actually drifting in reality.

3.2 Principe of classification neuronal model adaptation process

To take account these limitations, a model monitoring approach is proposed for adapting the forecasting model. The principle of model monitoring is illustrated in Figure 4 and is an extension of the classical process presented figure 1. In this approach, a forecasting model is extracted from a dataset in the first step by using a classical KDD process. In the second step, a monitoring procedure is used to detect the occurrence of drift and to react. The main objective of this paper is related to this monitoring process.

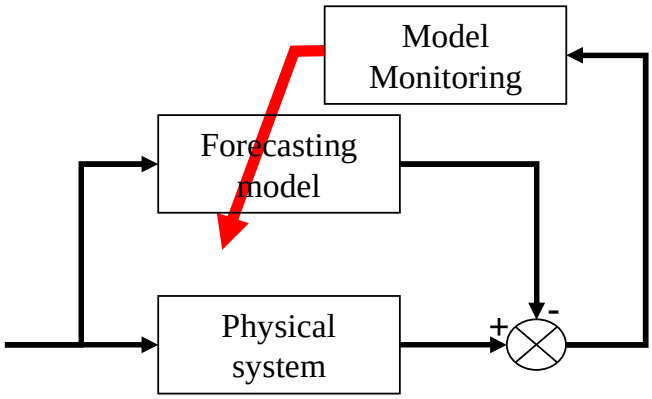

Figure 4 - Monitoring the forecasting model.

Many optimization algorithms are subdivided in different tasks in order to reach their goals. We can cite, as example, pruning algorithms which tend to determine the optimal structure of neural network after learning step (Thomas and Suhner 2015), or METSK-HD algorithm (Gacto et al. 2014) which combines a classical evolutionary learning of fuzzy model in a first step and a post processing step allowing to perform rule selection and tuning of membership function. In this paper, two main tasks must be performed to design the monitoring system, each of which is subdivided into different subtasks, as follows.

1) Design the forecasting model (batch learning process described section 2)
    a. dataset collection (described section 2.1)
    b. data mining or learning (described section 2.2)
2) Design the model monitoring system
    c. Collect the data
    d. Detect the occurrence of drift between the forecasting and physical systems (described later section 3.3)
    e. Construct the relearning database (estimate the start time of the drift) (described later section 3.4)
    f. Adapt the forecasting system

The first main task (1) is a classical KDD process and it is described in section 2. It allows to obtain the initial forecasting model.

The design of the model monitoring system (2) includes the collect of data during the production (c) in order to construct the dataset. It corresponds to the same process as that one described section 2.1. This dataset is used firstly to detect a drift occurrence by using SPC tools (d) and then to estimate the start time of the drift in order to define the relearning dataset by using PHT (e). This start time estimated allows to design the dataset used during the last sub step (f). This dataset includes all the data collected between the estimated start time and the present time. The last step is a relearning step (f). Its goal is to fit the model with the reality by adapting the parameters of the forecasting model to the detected drift.

The process presented in Figure 5 can be followed to adapt the model to changes.

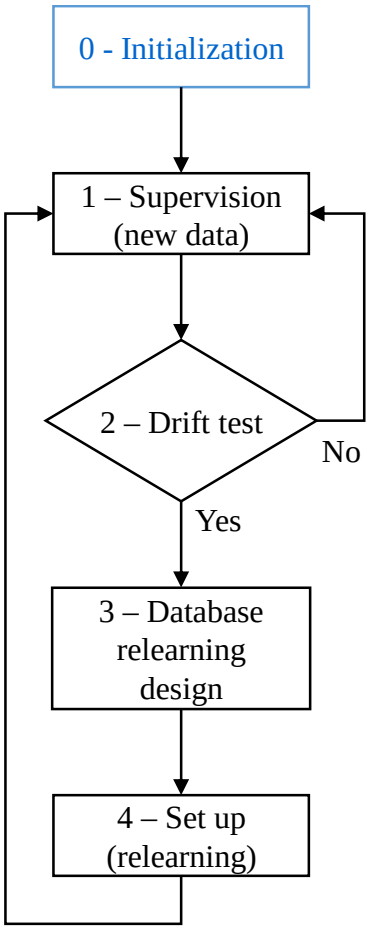

Figure 5 - Adaptation process.

The initialization process was described in section 2 and it corresponds to the design of the forecasting model, which is a classical KDD task, including here the dataset collection (and pre-processing), and the data mining step (including initialization, learning and pruning sub steps). The supervision step is conducted at the same time as model exploitation. During this step, the results obtained by the monitoring model are compared with the data collected from the physical system in order to detect the emergence of drift between both and its correction. The step employed to detect the occurrence of drift is discussed in section 3.3. The relearning database must be constructed when drift is detected. This step is discussed in section 3.4. The setup of the classifier is the relearning step which is performed by using the classical learning algorithm as during the KDD process, here, the Levenberg–Marquard algorithm with a robust criterion (Thomas et al. 1999, Mathworks 2016). The relearning is initiated and performed as soon as a drift is detected, its start time estimated and the relearning dataset constituted.

3.3 Determining whether the system is drifting by using control charts

Control charts are particularly useful for dynamic control based on time-series data (Tague, 2004). This method is useful for statistically determining whether the dimensional variation of parts is no longer under control. Indeed, it is known that even when a process is under control, there is a probability of approximately 0.27% that a point will exceed 3-sigma bilateral control limits (Pareto). These few isolated points should not trigger relearning, but an increase in the number of points will be detected to indicate the presence of a special cause, even if it is not yet known.

We propose the combination of a NN with control charts to exploit the robustness of statistical analysis and the adaptability of the NN. Du et al. (2012) studied the inverse combination of both tools by using a recognition algorithm for control charts and NNs to obtain alerts in the case of quality problems, as well as providing clues to identify the causes.

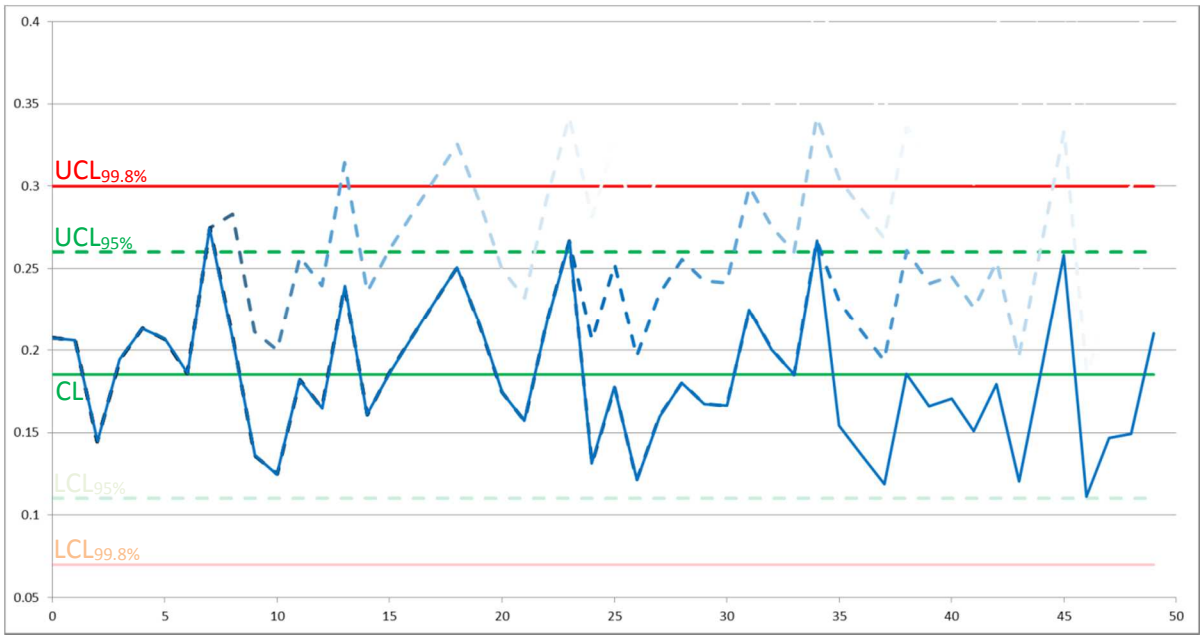

Figure 6 – Control chart for monitoring the real-time forecasting model.

Control charts (p-charts) (NIST/SEMATECH 2012) aim to determine whether the misclassification rate is drifting according to the principle illustrated in Figure 6, where each point in the figure corresponds to the misclassification rate obtained with a sample of *k* data and *k* corresponds to the sample size (*k* is selected according to several criteria such as criticality and frequency). Traditionally, two bounds are determined with control charts in order to define an acceptable zone for the misclassification rate (dashed green and red lines on the graph). The misclassification rate is an average of several values, so we assume that if this rate is outside the green bounds, then many values are outside the bounds, and thus the system is effectively drifting. Therefore, we need to perform relearning when the misclassification rate obtained for one sample (e.g., sample 7 in Figure 6) falls outside these bounds. The dotted lines in Figure 6 represent the evolution of the misclassification rate when no relearning is performed. Three examples are presented which present the evolution of the misclassification rate if the relearning needed at the 7th, 23th and 34th samples (misclassification rate > UCL95%) are not performed.

Two pairs of bounds may be determined using the confidence level. The warning bounds, i.e., the lower ($LCL_{95\%}$) and upper ($UCL_{95\%}$) bounds, are defined with a confidence level of 95%, as follows.

$$\begin{cases} LCL_{95\%} = p - 1.96\sqrt{\dfrac{p(1-p)}{k}} \\ UCL_{95\%} = p + 1.96\sqrt{\dfrac{p(1-p)}{k}} \end{cases} \quad (2)$$

The forbidden bounds, i.e., the lower ($LCL_{99.8\%}$) and upper ($UCL_{99.8\%}$) bounds, are defined with a confidence level of 99.8%, as follows:

$$\begin{cases} LCL_{99.8\%} = p - 3\sqrt{\dfrac{p(1-p)}{k}} \\ UCL_{99.8\%} = p + 3\sqrt{\dfrac{p(1-p)}{k}} \end{cases}, \quad (3)$$

where *k* corresponds to the size of the sample and *p* is the center line, which must be estimated. The center line corresponds to the misclassification rate obtained for the validation dataset (defined section 2.2) during the initial learning process (Noyel et al. 2013b).

The decision about relearning may be made when the misclassification rate of a sample is outside the warning bounds (e.g., for sample 7 in Figure 6). This step highlights when to trigger relearning and the next step determines the amount of data required to perform this action.

3.4 Determining the amount of data required for relearning

Control charts allow us to determine whether the system is actually drifting. To correct our forecasting system, we need to perform relearning based on a specific amount of new data. The relearning is achieved by using a batch second order backpropagation algorithm which includes the determination of hessian and gradient matrix whose size depends on the number of parameters (complexity of the model) and on the number of data (size of the dataset) (Thomas et al. 1999). So the re-learning speed is improved by reducing model size (pruning step, section 2.2) and reducing re-learning dataset (estimation of the start time of the drift). It can be noticed that the re-learning speed is not crucial. In fact, re-learning must be launched only when a new drift is detected by SPC, and so when *k* new data corresponding to the size of one sample in SPC are collected since the last supervision step (figure 5). The re-learning step disposes of the time of collect of these *k* new data to run.

The control charts enhance the data if drifting is detected, but they cannot determine when the drift started, and thus they cannot estimate the precise amount of data required to perform relearning. The following two cases must be considered.

- The time since the last relearning process is not crucial because it is possible to perform relearning based on all of the newly available data. Thus, the task is very simple and fast (similar to classical learning).
- The time since the last relearning process is crucial, and to save time, it is possible to restart the entire learning process based on a defined amount of data, where we can consider "sliding windows." This solution allows the system to forget old behaviors that may no longer be relevant so the behavior of the forecasting system is more flexible. The best data window size must be defined correctly. Thus, the system will learn the noise if it is too short, whereas there will be insufficient flexibility if it is too long. The aim is to determine the point of inflexion based on the error rate, i.e. the estimated start time of the drift.

Different methods may be used to determine the point of inflexion, such as adaptive windowing (ADWIN) (Bifet and Gavalda 2007), SPC (Gama et al. 2004), the fixed cumulative windows model (FCWM) (Sebastião et al. 2010), and PHT (Page, 1954). Sebastião and Gama (2009) tested and compared these different algorithms, where the results suggested that PHT and SPC are less time consuming than ADWIN and FCWM. This is crucial because one of the main objectives is to reduce the calculation time by optimizing the relearning dataset size. SPC cannot estimate the time when drifting begins but it can determine the drift detection time. This is important because the difference between these two times may be significantly large, and thus many data that are useful for relearning might be discarded from the relearning dataset. PHT can detect a drift (not used in the present study) and estimate the time drifting begins.

The goal of PHT is to detect a mean jump in a constant signal polluted by white noise (Page 1954; Hinkley 1971; Basseville 1986). This test can determine whether a jump occurs and estimate the time of this jump. In our case, the signal considered is the absolute value of the error obtained based on different data, where we conduct a search to determine the time drifting occurs between the behavioral model and the real system. This signal may be represented by a sequence of random Gaussian variables: $E = [e_i]$, $i = 1, \ldots, l$, with variance σ^2 and mean m_i. According to the hypothesis that only one jump occurs at an unknown time r with $1 \le r \le l$, detecting this jump corresponds to accepting hypothesis H_1 of a change rather than hypothesis H_0 of no change.

$$\begin{cases} H_0: & e_i \sim N(m_0, \sigma^2), \quad P(e_i) = P_0(e_i) \quad i = 1, \cdots, l \\ H_1: & e_i \sim N(m_0, \sigma^2), \quad P(e_i) = P_0(e_i) \quad i = 1, \cdots, r-1 \\ & e_i \sim N(m_1, \sigma^2), \quad P(e_i) = P_1(e_i) \quad i = r, \cdots, l \end{cases} \qquad 4$$

The use of this test implies that the two mean values m_0 and m_1 are known *a priori*. In our case, the mean m_0 may be estimated based on the mean of the error obtained for the validation dataset during the forecasting system design task. However, the mean m_1 is unknown and the minimal absolute value of the amplitude of the jump δ_m

that needs to be detected is fixed, where the two tests are performed in parallel to detect an increase and a decrease in the mean, respectively. These tests may be calculated recursively to detect the increase in the mean as follows:

$$\begin{cases} U_0 = 0, & U_i = U_{i-1} + e_i - m_0 - \dfrac{\delta_m}{2} \quad ,i \geq 1 \\ \gamma_0 = 0, & \gamma_i = \min(\gamma_{i-1}, U_i) \qquad\qquad ,i \geq 1 \end{cases}, \tag{5}$$

and the time r_i of the last increase in the mean is given as follows.

$$r_i = \max(i \,|\, U_i = \gamma_i) \tag{6}$$

For the decrease in the mean, the test is given by

$$\begin{cases} T_0 = 0, & T_i = T_{i-1} + e_i - m_0 + \dfrac{\delta_m}{2} \quad ,i \geq 1 \\ \eta_0 = 0, & \eta_i = \min(\eta_{i-1}, T_i) \qquad\qquad ,i \geq 1 \end{cases}, \tag{7}$$

and the time r_d of the last decrease in the mean is given as follows.

$$r_d = \max(i \,|\, T_i = \eta_i) \tag{8}$$

Therefore, if the control chart detects that relearning is required at time l, then relearning must be performed using all of the data collected between time r and l, where r given by:

$$r = \min(r_i, r_d)\,. \tag{9}$$

Tests (5) and (7) use the minimal absolute value of the amplitude of the jump δ_m, which may be fixed as a multiple of the standard deviation σ of the error obtained for the validation dataset. In the present study, we set δ_m as:

$$\delta_m = \frac{\sigma}{3}\,. \tag{10}$$

In function of the complexity of the neural model (number of parameters) and in the goal to avoid overfitting the minimal size of the relearning dataset must be limited to a threshold Δ such that $l\text{-}r > \Delta$. It can be noticed that the overfitting risk is limited by the use of a robust learning criterion (Thomas et al. 1999).

4 Application to a benchmark

In order to propose a simple and comprehensive application of the proposed approach, a simulation example is used to illustrate the procedure, which is derived from the example proposed by Lin et al. (2000). The main advantage of this simulation is the possibility to create artificially drifts (concept shift and concept drift) and to evaluate the capacities of the approach to detect the drifts, evaluate their start time and fit the model to the new reality. This example considers a population that comprises two subpopulations. The positive subpopulation follows a bivariate normal distribution with mean $(0, 0)^T$ and covariance matrix diag(1, 1), whereas the negative subpopulation follows two bivariate normal distributions with mean $(2, 2)^T$ and covariance diag(2, 1) for the first subpopulation, and mean $(–2, –2)^T$ with covariance diag(2, 1) for the second subpopulation. The population is unbalanced where the positive and negative subpopulations account for 80% and 20% of the total population, respectively. The negative subpopulation is balanced and follows two different laws in order to ensure that the two classes cannot be linearly separable.

4.1 Initial forecasting model

The first step is to determine the initial forecasting model as presented section 2. A dataset comprising 1000 pieces of data is constructed and divided into two datasets with 500 pieces of data in each: one for learning and the other for validation. A classification model is constructed using these data. The initial structure is constructed with two inputs and 10 hidden neurons. A pruning phase allows the deletion of three of the 10 hidden neurons. The resulting model obtains a misclassification rate of 8.1%.

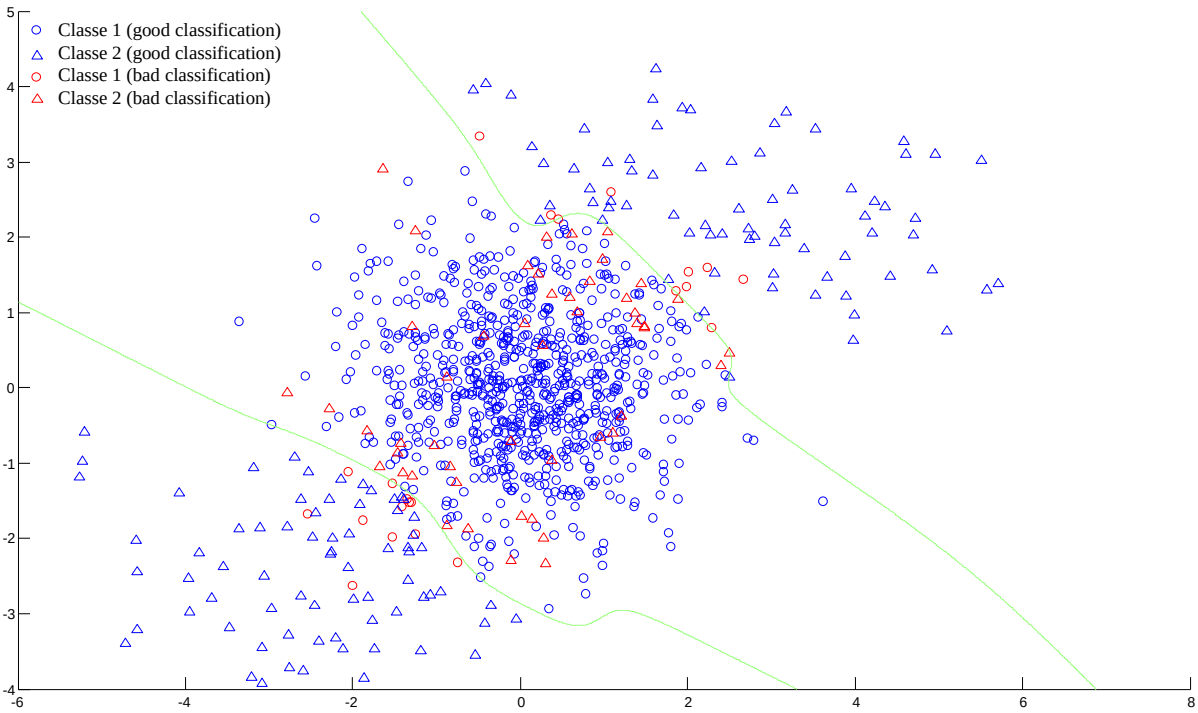

Figure 7 – The validation dataset divided into two classes and the limits between the two classes.

Figure 7 shows the validation dataset and the separation of the input space into the two classes given by the forecasting model.

Two supplementary datasets are constructed in order to illustrate the procedure, i.e., a concept shift and a concept drift.

4.2 Impact of concept shift

A dataset comprising 2000 pieces of supplementary data is constructed for the same simulation example, except the means of the normal distributions change at time 300. At this time, the mean of the positive population becomes $(1.5, 0)^T$ and the means of the two normal distributions for the negative population become $(3.5, 2)^T$ and $(–0.5, –2)^T$. The covariance matrix remains unchanged.

A control chart is constructed to monitor the forecasting model. The sample size for the control chart is fixed to 100. Two relearning procedures are used to adjust the model to reality. The first uses all of the available data based on the last relearning process, whereas the second uses the PHT procedure described in the previous section.

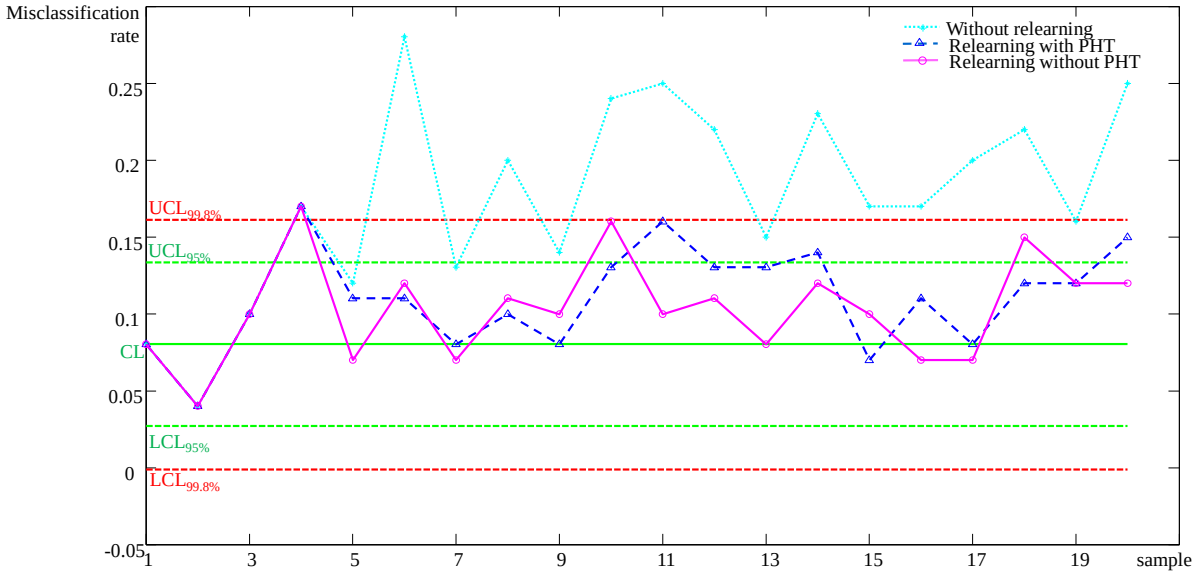

Figure 8 – Monitoring the forecasting model: concept shift.

Figure 8 shows the control chart used to monitor the forecasting model, where $UCL_{99.8\%}$ and $LCL_{99.8\%}$ are shown in the graph, but they are not useful. Each point outside the range [$LCL_{95\%}$ – $UCL_{95\%}$] triggers relearning. The evolution of the control chart without relearning is shown by the dotted cyan line, whereas that with relearning by using PHT to determine the appropriate size of the dataset is represented by the dashed line blue, and that with relearning using all of the available data based on the last relearning process is represented in magenta. The concept shift in the real system occurs at time 300, and the control chart detects this concept shift based on the fourth sample (which corresponds to time 400 because the size of each sample is 100). If relearning is not performed, the forecasting model cannot maintain good accuracy. The two relearning strategies allow us to adapt the model to the new behavior of the system even if other relearning steps are required after time 400 to maintain the accuracy of the forecasting model. The results obtained by the two relearning strategies are equivalent. Using the two strategies, no sample is outside the forbidden bounds.

Table 1 – Number, dataset size, time, and duration of relearning: concept shift.

	number of relearning	time to relearning	size of the dataset	duration
without PHT	3	400	400	0.35s
		1000	600	
		1800	800	
with PHT	4	400	143	0.17s
		1100	141	
		1400	209	
		2000	339	

Table 1 shows the number of relearning cycles, time, dataset size for each relearning cycle, and the duration of the overall procedure. This table shows that even if only three relearning cycles are required when PHT is not used (compared with four using PHT), the duration of the total procedure is 100% greater than with PHT. This is because the relearning procedures are performed based on relatively large datasets when PHT is not used.

4.3 Impact of concept drift

A new dataset comprising 2000 pieces of supplementary data is constructed for the same simulation example, except the drift occurs at time 300. At this time, the mean of the positive population becomes (0.02*(k – 300),

$0)^T$ and the means of the two normal distributions for the negative population become $(2 + 0.02*(k – 300), 2)^T$ and $(–2 + 0.02*(k – 300), –2)^T$, where k denotes the index of the data. The covariance matrix remains unchanged.

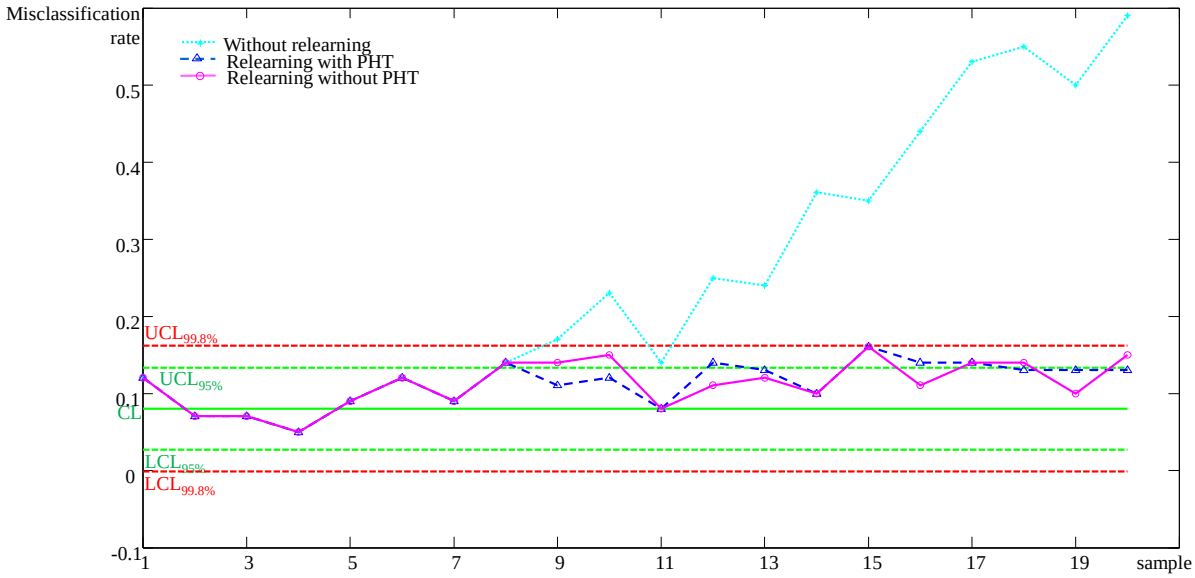

Figure 9 – Monitoring the forecasting model: concept drift.

As mentioned in the previous section, a control chart is constructed to monitor the forecasting model. The sample size for the control chart is fixed to 100. The same two relearning procedures are tested.

Figure 9 shows the control chart used to monitor the forecasting model. The evolution of the control chart when no relearning is performed is shown by the dotted line cyan, relearning using PHT to determine the appropriate size of the dataset is represented by the dashed line blue, and relearning using all of the available data based on the last relearning process is represented in magenta. The concept drift occurs at time 300, and the control chart detects this concept drift in the eighth sample (time 800). When relearning is not performed, the forecasting model cannot maintain good accuracy. The two relearning strategies allow the model to be adapted to the new behavior of the system and new relearning processes are launched periodically in order to maintain the accuracy of the forecasting model. The results obtained for the two relearning strategies are equivalent. Using the two strategies, no sample is outside the forbidden bounds.

Table 2 – Number, dataset size, time, and duration of relearning: concept drift.

	number of relearning	time to relearning	size of the dataset	duration
without PHT	7	800	800	0.35s
		900	100	
		1000	100	
		1500	500	
		1700	200	
		1800	100	
		2000	200	
with PHT	6	800	233	0.19s
		1200	361	
		1500	286	
		1600	91	
		1700	98	
		1800	95	

Table 2 shows the number of relearning cycles, time, dataset size for each relearning cycle, and the duration of the overall procedure. This table shows that using PHT avoids one relearning process. Moreover, as in the previous example, using PHT decreases the time required for the overall procedure by 50%. This is also because the relearning procedures are performed based on relatively large datasets when PHT is not used.

5 Application to an industrial quality monitoring problem

In order to illustrate the applicability of this approach a real industrial case is presented. Acta-Mobilier is a company that produces high-quality lacquered panels made of medium-density fiberboard (MDF) for kitchens, bathrooms, offices, stands, shops, and hotel furniture. According to its certifications (ISO 9001, ISO 14001, and OHSAS 18001), the product quality is a constant concern for this company. The manufacturing processes are implemented on several shop floors. In these workshops, each workstation is likely to generate defects and the company has to include a quality control step in each case. In this study, we focus on a robotic lacquering workstation because it has the highest defect rate. In literature, many industrial optimization problems have been considered and we can cite without to be exhaustive, logistics infrastructure problems (Kazakov and Lempert, 2015, Lempert et al. 2015), or continuous regulation problems (El Sehiemy et al. 2013, David et al. 2014). The main objective of this application is to determine the optimal tuning of the robotic lacquering workstation in order to reduce the defects rate.

The production quality of this workstation is unpredictable (the risk of defect occurrence is unknown) and fluctuating (the percentage of defects may vary from 45% one day to 10% the next day without any changes in the settings). It is very time consuming and difficult to obtain and plan a Taguchi experimental design in order to improve this setting. Thus, a robotic lacquering workstation is considered as a bottleneck workstation, but it is very difficult to reduce the time required for the experiments or to plan throughout the production lots according to the experimental conditions. In addition, the cost of these experiments is very high because they consume semi-finished products (which already have a high added value).

Therefore, a forecasting system is implemented to predict the occurrence of defects and to determine the optimal setting of the controllable factors considering the characteristics of the products and the environmental conditions. This forecasting system should highlight the relationships between process parameters and the quality of the finished products, which may be extracted from the dataset by using a NN (Yu et al. 2008, Xiaoqiao et al. 2015).

5.1 Forecasting system

The forecasting system can be represented as shown in Figure 10.

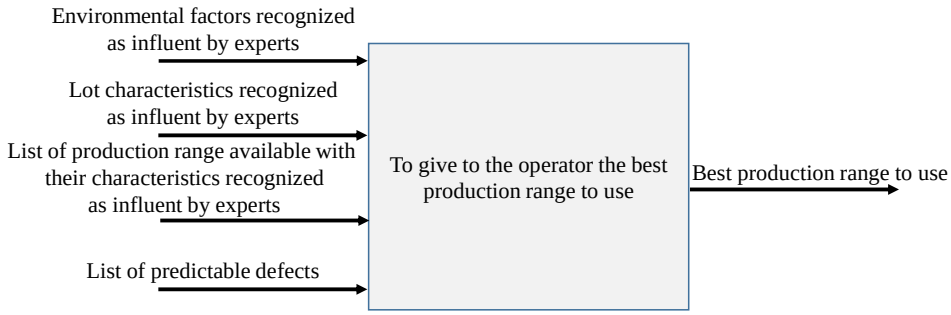

Figure 10 - Forecasting system.

The aim is to give the operator the best production range or the best parameters to set up the machine directly. So, we need a forecasting system able to predict the risk of defect occurrence considering the characteristics of the considered products, the state of the environmental factors and the different available settings. These defect

risk predictions allow to propose to the operator the best setting to use in the present condition. Thus, the system needs three different types of inputs, each of which requires many settings, as follows.

1. Environmental factors and lot characteristics, where there are two prerequisites. The first is the computerized production monitoring system, for which real-time production information must be collected in a semi-automated or automated manner. We do not consider that the operator has to enter data (e.g., temperature) at each production lot. The second is the virtualization of expert knowledge. A database must be implemented to consider the fact that only experts can know whether a factor is important for quality or not.
2. The list of available production range or parameter settings. In most cases, the workstation has one production range/setup according to the type of product. To obtain this high quality level, the production system needs to be flexible and adaptable. In the case where we consider different production ranges, alternative routings should be implemented. Similarly, in the case where we consider the parameter settings, adjustments to the limits of the parameters must be implemented.
3. The list of predictable defects and their criticality value. Experts can list the possible defects that may occur, but they cannot know whether a defect is actually predictable. This is one of the tasks that we discuss later in the learning step. However, experts can determine whether a defect is actually important. They need to attribute a criticality value to each possible defect according to different factors, such as the possibility of repair or the repair cost. Thus, if the system cannot find a solution with a zero defect probability, it will try to find a solution that minimizes the penalizing defects.

The forecasting system can be decomposed into two subsystems, as illustrated in Figure 11.

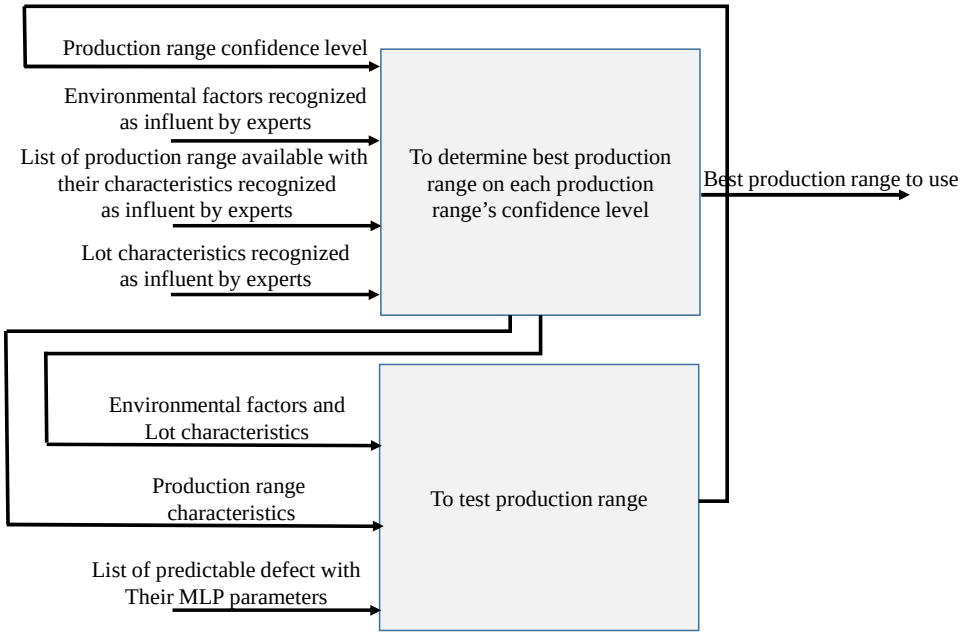

Figure 11 – Decomposition of the forecasting system.

The system needs to compare each production range, so we calculate a confidence level for each production range. This confidence level is obtained as shown in Figure 12 (decomposition of the "To test production range" in figure 11).

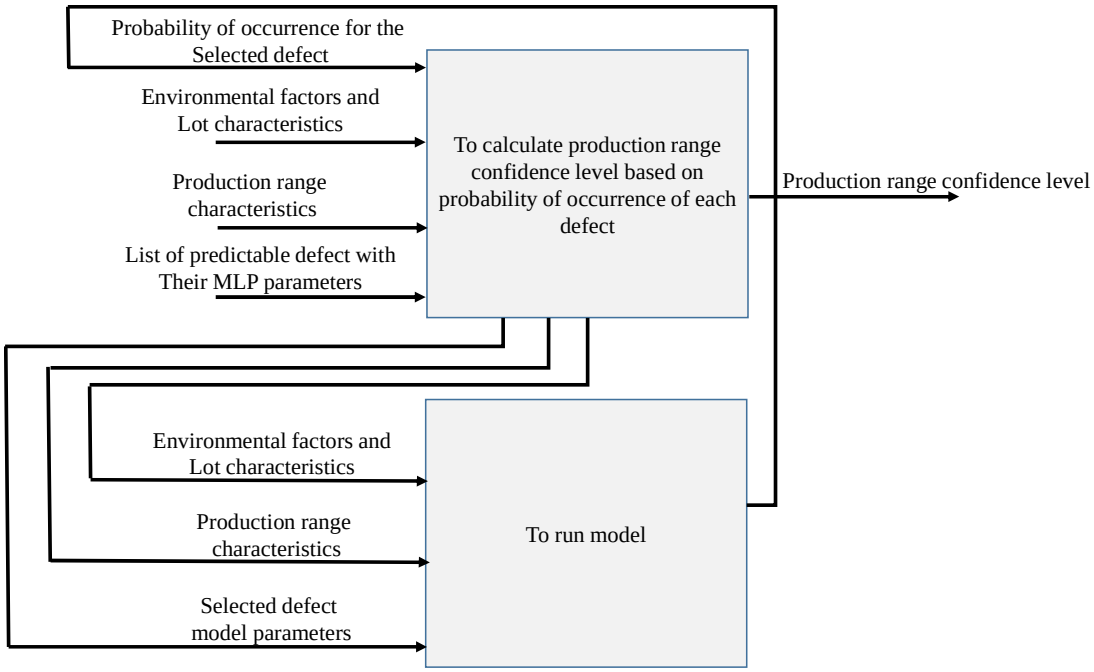

Figure 12 - Details of the production range's confidence level module.

To calculate the production range's confidence level, we first need to calculate the probability of occurrence for each defect. Thus, we use all of the factors validated by experts (such as environmental factors, lot characteristics, and production range characteristics) and a prediction model can be estimate for the probability of occurrence for each defect in this condition. The confidence level of production range P_x is calculated as follows:

$$CL(P_x) = \sum_{i=1}^{N_d} p_i \times \alpha_i, \qquad 11$$

where p_i denotes the probability of occurrence for defect i with $1 \leq i \leq N_d$, N_d indicates the number of different defects identified by experts, and $\propto_i$ represents the criticality level of defect i.

The only missing component is the probability of occurrence p_i. This probability is given by a forecasting model built using the approach proposed in Section 2.

5.2 Implementation of the forecasting system

In this application, 25 different quality defects should be considered, thereby leading to the design of 25 quality prediction neural models. The resulting quality monitoring system (set of 25 neural classifiers) is embedded in the supervision tool of the lacquering workstation for use by the operators. The memory of these NNs is physically remote in an SQL database; therefore, each independent program may access this memory if needed. This tool is a decision support system and it requires a human/machine interface, which is as intuitive as possible. The tool is implemented directly in the setup interface of the robotic lacquering workstation (Figure 13). Using this additional function, after entering the production information (such as the selected production range and the number of units produced), the operator may assess the risk of occurrence for a defect (Figure 14). If the risk appears to be too large, the production parameters can be changed (e.g., choosing another production range) and the program can be run in parallel to compare the evaluation results until a satisfactory result is obtained.

The current version of the quality monitoring system requires an average of 12 seconds to display the result. Thus, within 12 seconds, it can recover the memory from the SQL database, traverse the 25 NNs, and visually synthesize the results to facilitate interpretation by the operator, so less than half a second is required for the calculation by the NN.

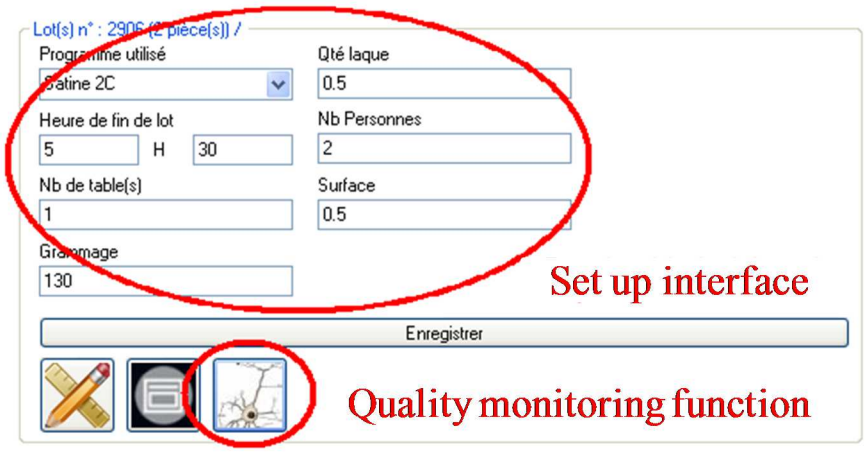

Figure 13 - Interface for collecting production data.

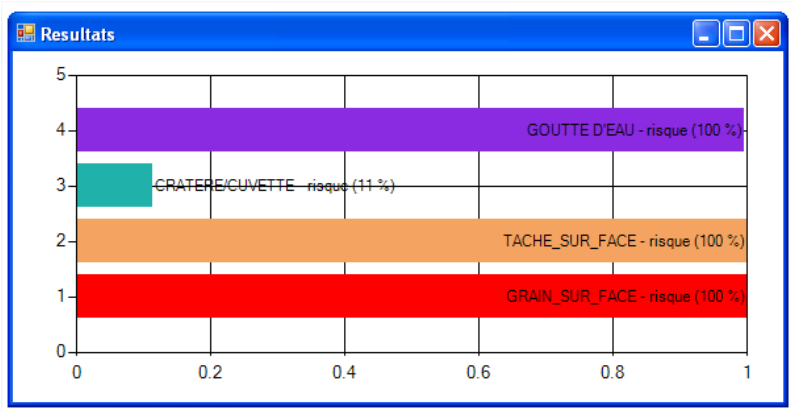

Figure 14 - Forecast example

However, as expected, the answers provided by our system drift away from reality shortly after its implementation.

5.3 Design of the initial forecasting neural model

As explain in section 5.2, 25 different quality defects must be monitored leading to the design of 25 different neural forecasting models. In the sequel, we focus our presentation on one particular defect: “stain on back”. The different factors which may have an impact on this defect occurrence are collected. These factors are technical factors (load factor, number of passes, time per table (lacquering batches), liter per table, basis weight, number of layers, number of products and drying time), environmental ones (as temperature, atmospheric pressure and humidity). Some of the technical factors are imposed by the products (number of passes, time per table, liter per table, number of layers, and number of products). The three last factors (load factor, basis weight, drying time) are the tunable parameters whose optimal setting is seeking. Some of these factors are discrete and are binarized. So the initial structure of the neural model includes 15 inputs (9 continuous and 6 binary) and 25 hidden neurons.

The dataset is constituted of 2270 data and is split into 2 data sets for identification (1202 data) and validation (1068 data). After initialization and learning, pruning phase is able to eliminate spurious inputs and hidden neurons. 6 hidden neurons and 1 input (passes number) are eliminated.

During the validation phase, we therefore compare the results of the NN with the real defects detection. The defect "Stains on back" occurs 127 times on the 1068 data validation set. The NN can detect 112 defects which lead to a non-detection rate of 11.8%. The proportion of false positive is 19.2%, which may be partly explained by the fact that some defects haven’t been identified out of the machine (Noyel et al. 2013a).

5.4 Drift

Two apparent reasons lead our monitoring system to drift away from reality. The first concerns the evolution of the input parameters. Using a learning model, the learning outcome is valid only in the learned domain, so the model can only provide a valid solution in this domain.

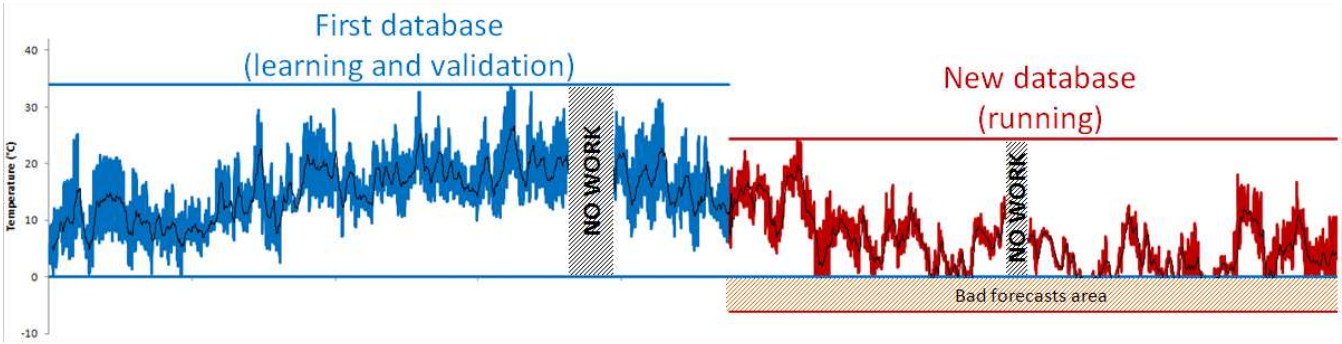

Figure 15 - Difference between the learning and running domains.

For our implementation, the learning and validation databases were collected during spring and summer. For the new database (UCI 2016), 446 items of data were collected during autumn and winter, and the exploitation of the quality monitoring process led to 73% non-detections and 32% false positives for one of the 25 defects monitored. These poor results can be explained by the different process conditions in the two periods. As shown in Figure 15, in the first database, the temperature range varies between $0°C$ and $32°C$, whereas in the new database, the temperature range varies between $-5.2°C$ and $24°C$. These negative temperatures represent 25% of the new database and they correspond to the operating range of the process, which is not learned during the quality monitoring process.

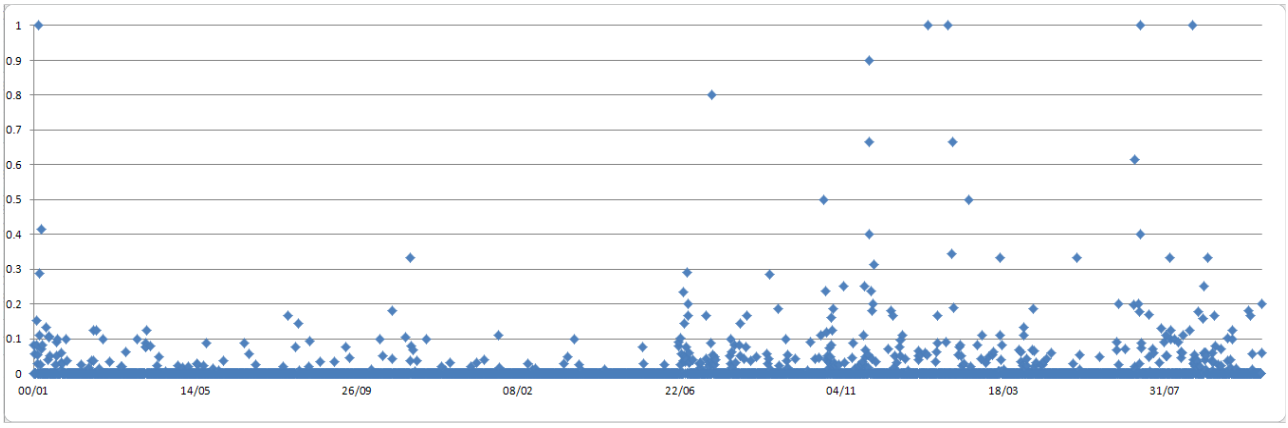

Figure 16 - Historic data on the defect percentage for grains on edges.

The second reason concerns the uncontrolled modification of the machine behavior. Indeed, it is still possible to affect the behavior of the machine by changing a parameter (voluntarily or not) that is not an input of the neural classifier. For example, this parameter may change due to the clogging of a filter or the replacement of a dirty filter. These changes may or may not be known. Thus, in the application considered, we know that the lacquering nozzles are changed during the exploitation phase, but the time of this change is not known. In this case, we can conclude that this parameter should be part of the model inputs, but because it is considered constant for the duration of the learning step, it is not actually retained. The model will produce results that do not agree with reality because of this change, which may be unknown to the operators and managers.

For example, as shown in Figure 16, it is clear that the studied defect rate (grains on edges) increases sharply after June 22, which is due to an unknown change in the real system, so the forecasting system is no longer pertinent.

It is not always possible to control changes in the production parameters (such as uncontrollable parameters, weather, and unanticipated changes made by the operator), so it is necessary to be capable of detecting them. By providing the quality monitoring system with the capacity to verify its hypotheses about reality, it can have the ability to recognize its failure and react accordingly.

Therefore, a control chart is designed to monitor the forecasting system, where the sample size for the chart is fixed to 100 values, which corresponds to slightly less than one week of production.

Only the UCLs are considered because the model is better when the misclassification rate is lower. Therefore, only the UCLs are calculated to represent 95% and 99.8% of the data.

Figure 17 shows the control charts obtained with the new dataset during the exploitation phase. The dotted line corresponds to a control chart without relearning. We can see that the quality process is under control for sample 1 but the second sample shows that the process is no longer under control (the results are between $UCL_{95\%}$ and $UCL_{99.8\%}$). This is due to the new operating range detected in the data, as explained in the previous section. Therefore, the quality monitoring process must be improved by relearning the NN using the data from the first two samples.

The initial structure and weights of the network are those given by the original quality monitoring process, so a pruning phase is note needed. The initial weights are close to the optimal values; therefore, the relearning phase is fast and it requires only a few iterations.

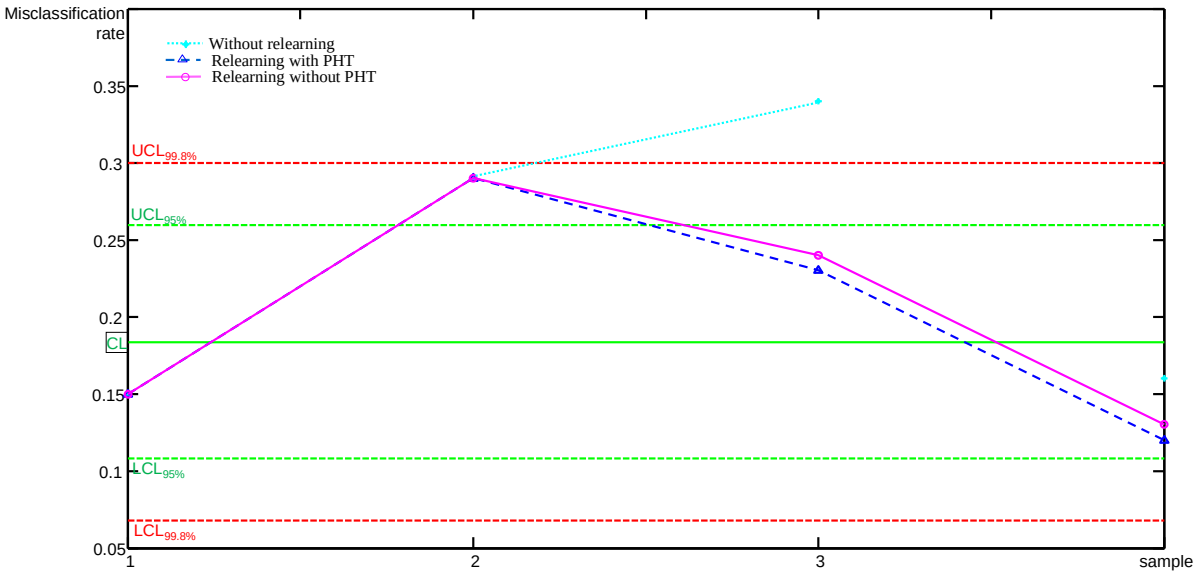

Figure 17 - Industrial application of forecast improvement by relearning.

The evolution of the control chart when no relearning is performed is shown by the dotted cyan line, relearning using PHT to determine the appropriate dataset size is represented by the dashed blue line, and relearning using all of the available data based on the last relearning process is represented by the continuous magenta line. This chart shows that relearning allows the quality monitoring process to be adapted to the new operating range. Thus, the results for sample 3 are greatly improved and the process remains under control until the end. The relearning processes with and without PHT obtain very similar results.

Table 3 – Number, dataset size, time, and duration of relearning: industrial example.

	number of relearning	time to relearning	size of the dataset	duration
without PHT	1	200	200	0.22s
with PHT	1	200	101	0.14s

Table 3 shows the number of relearning cycles, time, dataset size for each relearning cycle, and the duration of the overall procedure. This table shows that using PHT reduces the time required by the overall procedure by 36%. This approach allows us to determine whether the quality monitoring process needs to be adapted without systematic relearning.

All this procedure is applied in order to maintain the accuracy of the defect prediction models on the robotic lacquering workstation. These prediction models are use in a second step in order to find the optimal tuning of parameters which limits the defect risk considering the product characteristics and the environmental condition (Noyel et al. 2013a).

6 Conclusion

Predicting the behavior of a real system requires the use of a forecasting model that behaves in as similar manner as possible to the real system. However, drifts and shifts can rapidly create a difference between the model's behavior and reality.

Therefore, in the present study, we proposed a method for adapting a classification neuronal model to this particular changing context by using control charts and PHT. The main goal is to keep under control the misclassification rate in order to maintain the model close to the reality. This novel hybrid system allows us to reduce the time required because of the following two reasons: the relearning process is not systematic and it is triggered only when a drift is finally detected; and the relearning process is not performed based on all of the available data, but instead it only uses the data that reflect the drift. Thus, we proposed a model monitoring approach, which aims to detect drifts and shifts between reality and the forecasting model. We tested this method based on a benchmark case and the results were promising. The results obtained for the industrial quality monitoring problem demonstrate that the process can be brought under control after adapting the model. These results based on the management of quality indirectly influence the management of flows in the system.

This approach tends to adapt the model when each drift is detected. However, in some cases, a drift may be a symptom of a process failure, so the process (and not the model) must be repaired. In our future research, we will try to determine whether the process or the model should be corrected when a drift occurs.

Bibliography

Adae, I., & Berthold, M., (2013). Eve: a framework for event detection. Evolving systems 4, 61–70.

Agard, B., & Kusiak, A. (2005). Exploration des bases de données industrielles à l'aide du datamining - Perspectives. *9ème colloque national AIP PRIMECA*.

Basseville, M. (1986). Detecting changes in signals and systems. Research Report RR-0658, INRIA, https://hal.inria.fr/file/index/docid/75895/filename/RR-0658.pdf.

Basseville, M., & Nikiforov, I., (1993). *Detection of Abrupt Changes: Theory and Applications*. Prentice-Hall Inc.

Bifet, A., & Gavalda, R. (2007). Learning from Time-Changing Data with Adaptive Windowing. *SDM*.

Bouchachia, A., (2011). Fuzzy classification in dynamic environments. Soft Comput. 15, 5, 1009–1022.

Clifford, M., & Duncan, S. (2002). Benefits of continuous on-line monitoring of mapping for CD control systems. *Pulp & Paper - Canada*, *103*(8), 48-51.

David R.C., Precup, R.E., Petriu, E.M., Radac, M.B., & Preitl, S. (2014). Gravitational search algorithm-based design of fuzzy control systems with a reduced parametric sensitivity. *Information Sciences, 247*, 154-173.

Dean, T., & Boddy, M., (1988). An analysis of time-dependent planning. Proceedings of the seventh national conference on artificial intelligence, pp. 49–54.

Dries, A., & Ruckert, U., (2009). Adaptive concept drift detection. Stat. Anal. Data Min. 2, 5-6, 311–327.

Du, L., Ke, Y., & Su, S. (2012). The embedded Quality Control System of Product Manufacturing. *Advanced Materials Research*, *459*, 510-513.

El Sehiemy, R., El Ela, A.A., & Shaheen, A. (2013). Multi-objective fuzzy-based procedure for enhancing reactive power management. *IET Generation, Transmission & Distribution, 7(12)*, 1453-1460.

Gacto, M.J., Galende, M., Alcal, R., & Herrera, F. (2014). METSK-HD^{e}: A multiobjective evolutionary algorithm to learn accurate TSK-fuzzy systems in high-dimensional and large-scale regression problems. *Information Sciences, 276*, 63-79.

Gama, J., (2010). *Knowledge Discovery from Data Streams*. Chapman and Hall/CRC Press.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. (S. B. Heidelberg, Éd.) *Advances in Artificial Intelligence - SBIA 2004*, pp. 286-295.

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A., (2014). A survey on concept-drift adaptation. ACM Computing Surveys, 46(4).

Harries, M., Sammut, C., & Horn, K., (1998). Extracting hidden context. Machine Learning 32, 101–126.

Hinkley, D.V. (1971). Inference about the change-point from cumulative sum tests. *Biometrika*, 58, 509-523.

Karp, R. (1992, 8). On-line algorithms versus off-line algorithms: How much is it worth to know the future? *IFIP Congress*, *12*, 416-429.

Kazakov, A.L., & Lempert, A.A., (2015). On mathematical models for optimization problem of logistics infrastructure. *International Journal of Artificial Intelligence*, 13, 200–210.

Klinkenberg, R., & Renz, I., (1998). Adaptive information filtering: Learning in the presence of concept drifts. Workshop Notes of the ICML/AAAI-98 Workshop on Learning for Text Categorization. 33–40.

Kotsiantis, S.B. (2007). Supervised machine learning: A review of classification techniques. *Informatica, 31*, 249-268.

Kusiak, A. (2001). Rough set theory: a data mining tool for semiconductor manufacturing. *Electronics Packaging Manufacturing; IEEE Transactions on, 24*, 44-50.

Lazarescu, M.M., Venkatesh, S., & Bui, H.H., (2004). Using multiple windows to track concept drift. *Intelligent Data Analysis*, 8, 29–59.

Lempert, A.A., Kazakov, A.L., & Bukharov, D.S., (2015). Mathematical model and program system for solving a problem of logistic objects placement. *Automation and Remote Control*, 76, 1463–1470.

Lin, Y., Lee, Y., Wahba, G. (2000). Support vector machines for classification in nonstandard situations. *Technical Repport, http://roma.stat.wisc.edu/sites/default/files/tr1016.pdf*.

Mathworks (2016) fileexchanges https://fr.mathworks.com/matlabcentral/fileexchange/58102-mlp-learning

Meyer, D., Leisch, F., & Hornik, K. (2003). The support vector machine under test. *Neurocomputing, 55*, 169-186.

Michalski, R. S., Mozetic, I., Hong, J., & Lavrac, N., (1986). The Multi-Purpose incremental Learning System AQ15 and its Testing Application to Three Medical Domains. Pro*ceedings of the Fifth National Conference on Artificial Intelligence*, 1041–1045.

Nguyen, D., & Widrow, B. (1990, juin 17). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *IJCNN International Joint Conference on Neural Networks 1990*, 21-26.

NIST/SEMATECH, (2012). e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/.

Noyel, M. (2015) C*ontrôle intégré du pilotage d'atelier et de la qualité des produits, application à la société ACTA mobilier*, PhD thesis of université de Lorraine.

Noyel, M., Thomas, P., Charpentier, P., Thomas André, & Beaupretre, B. (2013). Improving production process performance thanks to neuronal analysis. *Intelligent Manufacturing System*.

Noyel, M., Thomas, P., Charpentier, P., Thomas, A., & Brault, T. (2013). Implantation of an on-line quality process monitoring. *International Conference on Industrial Engineering and Systems Management, IESM'13*.

Page, E.S. (1954). Continuous inspection schemes. *Biometrika*, 41, 100-115.

Paliwal, M., & Kumar, U. (2009). Neural Networks and Statistical Techniques: A review of applications. *Expert Systems with Applications, 36*, 2-17.

Patel, M., & Panchal, M. (2012). A review on ensemble of diverse artificial neural networks. *Int. J. of Advanced Research in Computer Engineering and Technology, 1*(10), 63-70.

Ragot, J., Maquin, D., Darouach, M., & Bloch, G., (1990). Validation de données et diagnostic. Hermès, Paris.

Refke, A., Barbezat, G., & Loch, M. (2001). The benefit of an on-line diagnostic system for the optimization of plasma sprays devices and parameters. *Thermal Spray 2001: New Surfaces for a new millennium*, 765-770.

Salperwyck, C., & Lemaire, V., (2009). Classification incrémentale supervisée : un panel introductif. *Proc. of the 9ème conférence francophone Extraction et Gestion des Connaissances EGC'09*, 27 au 30 Janvier, Strasbourg, France.

Sarle, W.S. (2002) (current editor). Comp.ai.neural-nets FAQ, part 2, available on-line at: ftp://ftp.sas.com/pub/neural/FAQ2.html.

Sebastiéão, R., & Gama, J. (2009). A study on Change Detection Methods. Proceedings of the 14th Portuguese Conference on Artifcial Intelligence, 353–364, *http://epia2009.web.ua.pt/onlineEdition/353.pdf*.

Sebastião, R., Gama, J., Rodrigues, P., & Bernades, J. (2010). Monitoring incremental histogram disctribution for change detection in data streams. (S. B. Heidelberg, Éd.) *Knowledge Discovery from Sensor Data*, pp. 25-42.

Shewhart, W. (1931). *Economic control of quality of manufactured product*. New York, 501.

Sick, B. (2002). On line and indirect tool wear monitoring in turning with artificial neural networks: a review of more than a decade of research. *Mechanical Systems and Signal Processing, 16*, pp. 487-546.

Stirl, T., & Skrzypek, R. (2003). Practical experiences and benefits with on-line monitoring systems for power transformers. *Proceedings of the 6th international conference on electrical machines and systems ICEMS 2003, 1*(2), 309-313.

Tague, N. (2004). *The Quality Toolbox, 2nd Edition.* ASQ Quality Press.

Thomas, P., Bloch, G., Sirou, F., & Eustache, V. (1999). Neurall modeling of an induction furnace using robust learning criteria. (I. Press, Éd.) *Integrated Computer-Aided Engineering, 6*(1), 15-26.

Thomas P., Suhner, M.C., Meutelet, B., Brachotte, G. (2004). Quality monitoring of a high pressure die casting process based on bayesian and neural networks. *11th IFAC Symposium on automation in Mining Mineral and Metal processing MMM'04*, Nancy, France September 8-10.

Thomas P., Suhner M.C., (2015). A new multilayer perceptron pruning algorithm for classification and regression applications. *Neural Processing Letters*, 42(2), 437-458

UCI (2016) Machine Learning Repository. http://archive.ics.uci.edu/ml/

Wald, A., 1947. Sequential Analysis. John Wiley and Sons, Inc.

Xiaoqiao W, Mingzhou L., Maogen G., Lin L., Conghu L., (2015). Research on assembly quality adaptive control system for complex mechanical products assembly process under uncertainty. *Computers in Industry*, 74, 43-57.

Yu J., Xi L., Zhou X., (2008). Intelligent monitoring and diagnosis of manufacturing processes using an integrated approach of KBANN and GA. *Computers in Industry*, 59, 489-501.